



RESTREINT

RAPPORT D'ANALYSE

PRÉPARÉ POUR

Ostorlab Official

RISQUE GLOBAL - ÉLEVÉ

DATE

2026-10-15 18:03:02

Résumé exécutif

L'application  **PlayStore** (`owasp.sat.agoat:1.0 (1)`) s'est vu attribuer un **ÉLEVÉ | 9,1** niveau de risque et présente actuellement **44** vulnérabilités, **0** dont sont considérées comme risque critique, **12** considérées comme risque élevé, **16** considérées comme risque moyen et **6** considérées comme risque faible. Il y a **8** vulnérabilités potentielles nécessitant une investigation approfondie. Pour renforcer la sécurité de l'application, la mise en place de **2** mesures de durcissement est recommandée.

La surface d'attaque de l'application est **Notable**, car elle utilise **0** permissions, dont **0** permissions dangereuses. L'application expose également **0** activités, **0** service, et **0** récepteurs de diffusion. De plus, elle utilise **0** fonctionnalités. Assurez-vous que votre surface d'attaque est justifiée et supprimez toute fonctionnalité ou exposition inutile.

Globalement, la couverture de l'application est jugée **Limitée**. Pour améliorer votre couverture, il est recommandé de lancer une analyse dynamique complète.

Résumé de l'application

Plateforme	 Android
Paquet	owasp.sat.agoat
Version	1.0 (1)
Empreinte SHA1	<code>ea96f4adc27083696a449b1c6e5cd612a2cf0f5c</code>
Taille	2 MB
Date	Mai 19th 2026, 16:40:34

Résumé des résultats

Vulnérabilités groupées par niveau de risque



Résumé des tickets

Aperçu des statuts des vulnérabilités

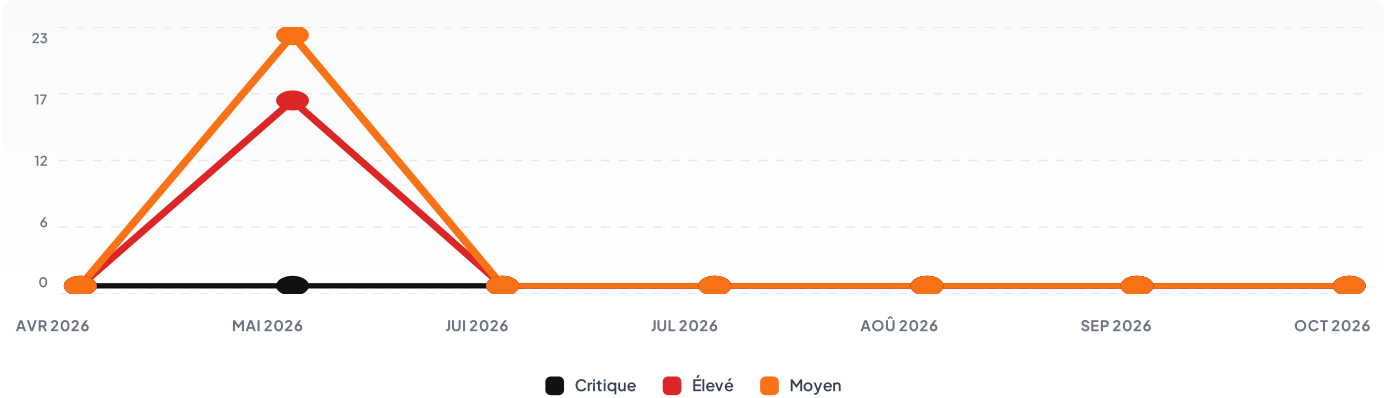


Top 10 des vulnérabilités par sévérité

OTP à faible entropie exposé ...		1
Injection SQL via rawQuery no...		1
Activité protégée impliciteme...		1
Service exporté permettant le...		1
Identifiants sensibles stockés ...		1
Identifiants d'authentification...		1
Clickjacking confirmé sur des ...		1
Fuite d'identifiants d'authenti...		1
Injection SQL via execSQL no...		1
Base de données Firebase exp...		1

Tendances des vulnérabilités de l'application

Tendances historiques des 6 derniers mois



Résumé de l'analyse

Cible	Android: owasp.sat.agoat (1.0 (1))
Risque global	HIGH
Vulnérabilités	34 au total 0 CRITIQUE 12 ÉLEVÉ 16 MOYEN 6 FAIBLE 8 POTENTIEL 2 DURCISSEMENT

Tableau des résultats

Risque CVSS	Statut(s)	Étiquettes	Titre	Description courte
ÉLEVÉ	Ouvert	Aucune étiquette disponible.	OTP à faible entropie exposé via le presse-papiers système et un PRNG faible	<code>ClipboardActivity</code> génère un OTP à quatre chiffres à l'aide d'un PRNG non cryptographique, le copie dans le presse-papiers global et l'affiche dans un Toast, rendant le jeton prévisible et accessible par toute application co-installée ou tout IME.
ÉLEVÉ	Ouvert	Aucune étiquette disponible.	Injection SQL via rawQuery non assaini dans SQLInjectionActivity	SQLInjectionActivity construit une requête SELECT en concaténant directement l'entrée utilisateur d'un EditText dans rawQuery() avec des selectionArgs à null, permettant l'exécution arbitraire de SQL, l'énumération du schéma et le contournement de l'authentification locale.
ÉLEVÉ	Ouvert	Aucune étiquette disponible.	Activité protégée implicitement exportée permettant le contournement de l'authentification (AccessControlViewActivity)	AccessControlViewActivity is implicitly exported via a deep-link intent-filter and contains no internal authentication checks, allowing any third-party application to launch the protected view without a PIN.
ÉLEVÉ	Ouvert	Aucune étiquette disponible.	Service exporté permettant le téléchargement non autorisé de fichiers (DownloadInvoiceService)	<code>DownloadInvoiceService</code> is explicitly exported in the manifest and performs file downloads to public external storage without any caller authorization checks.
ÉLEVÉ	Ouvert	Aucune étiquette disponible.	Identifiants sensibles stockés en clair dans SharedPreferences, SQLite et le stockage externe	L'application stocke les noms d'utilisateur, les mots de passe et les hachages de codes PIN dans plusieurs mécanismes de stockage local sans chiffrement, protection par KeyStore ni permissions de fichier restrictives.
ÉLEVÉ	Ouvert	Aucune étiquette disponible.	Identifiants d'authentification protégés par hachage MD5 non salé (AccessControlIssue1Activity)	L'application stocke les codes PIN sous forme de hachages MD5 non salés dans les SharedPreferences internes, permettant une récupération triviale du code PIN en clair par force brute hors ligne en quelques millisecondes.
ÉLEVÉ	Ouvert	Aucune étiquette disponible.	Clickjacking confirmé sur des activités sensibles dans owasp.sat.agoat	Une application malveillante disposant de SYSTEM_ALERT_WINDOW peut afficher des overlays interactifs par-dessus quatre activités sensibles, permettant d'intercepter les touches de l'utilisateur pour déclencher sans autorisation des téléchargements de factures, des vérifications de promotions, des générations d'OTP et des soumissions de code PIN.
ÉLEVÉ	Ouvert	Aucune étiquette disponible.	Fuite d'identifiants d'authentification en clair et de requêtes SQL dans Android Logcat	Plusieurs activités écrivent sans condition des noms d'utilisateur, des mots de passe et des instructions SQL en clair dans les logs système Android via <code>Log.e</code> , <code>Log.v</code> et <code>System.out.println</code> , permettant l'exfiltration d'identifiants par toute application disposant d'un accès aux logs ou par un attaquant utilisant <code>adb</code> .
ÉLEVÉ	Ouvert	Aucune étiquette disponible.	Injection SQL via execSQL non assaini dans InsecureStorageSQLiteActivity	InsecureStorageSQLiteActivity construit une instruction INSERT en concaténant deux valeurs EditText non assainies dans execSQL(), ce qui permet une injection de requêtes empiilées, une manipulation arbitraire des enregistrements et une compromission complète de l'intégrité de la base de données locale.
ÉLEVÉ	Ouvert	🚫 Sécurité	Base de données Firebase exposée publiquement	exposer publiquement une base de données Firebase, pouvant entraîner des accès non autorisés, l'altération de données, la divulgation d'informations, des attaques malveillantes et une perte de réputation.
ÉLEVÉ	Ouvert	🔒 Confidentialité 🚫 Sécurité	Vulnérabilité LOGJAM – Nombre premier commun	Utilisation de nombres premiers Diffie-Hellman largement partagés, permettant à des acteurs étatiques de mener des attaques par précalcul contre une grande partie du trafic Internet.

Risque CVSS	Statut(s)	Étiquettes	Titre	Description courte
ÉLEVÉ	Ouvert	🚫 Sécurité	Vulnérabilité d'injection SQL mobile	Une vulnérabilité d'injection SQL mobile permet à un attaquant d'exécuter des requêtes SQL malveillantes sur la base de données d'une application mobile, pouvant entraîner un accès non autorisé, une fuite de données ou une manipulation de données.
MOYEN	Ouvert	Aucune étiquette disponible.	Privilèges administrateurs déterminés par un flag booléen côté client sans vérification côté serveur	BinaryPatchingActivity contrôle l'accès aux fonctionnalités de l'interface d'administration via un champ booléen local qui n'est jamais validé par une API backend, un jeton cryptographique ni un challenge de réauthentification.
MOYEN	Ouvert	Aucune étiquette disponible.	Échec silencieux du Certificate Pinning dans TrafficActivity en raison d'une gestion trop large des exceptions	Le CertificatePinner d'OkHttp dans <code>TrafficActivity.doPinning()</code> est neutralisé car <code>SSLPeerUnverifiedException</code> est avalée par un bloc catch trop large, permettant des attaques silencieuses de type Man-in-the-Middle.
MOYEN	Ouvert	Aucune étiquette disponible.	Fuite d'identifiants due à des permissions de stockage externe non sécurisées (InsecureStorageSDCardActivity)	<code>InsecureStorageSDCardActivity</code> écrit les identifiants utilisateur dans un fichier temporaire sur le stockage externe et définit explicitement ce fichier comme lisible et modifiable par tous.
MOYEN	Ouvert	Aucune étiquette disponible.	Les écrans sensibles permettent la capture d'écran non autorisée en l'absence de FLAG_SECURE	Toutes les 25 activités de owasp.sat.agoat ne définissent pas le flag FLAG_SECURE, permettant la capture et l'enregistrement d'écran sans restriction d'écrans sensibles, notamment les écrans de saisie du code PIN, d'identifiants et de génération d'OTP.
MOYEN	Ouvert	Aucune étiquette disponible.	Des flags de backup et de debug non sécurisés exposent la sandbox de l'application	Le manifeste de l'application active <code>android:debuggable="true"</code> et <code>android:allowBackup="true"</code> , permettant à tout utilisateur disposant d'un accès au débogage USB d'extraire, de modifier et de restaurer l'ensemble des données privées de l'application, ou d'attacher un débogueur au processus en cours d'exécution.
MOYEN	Ouvert	Aucune étiquette disponible.	Identifiants d'authentification codés en dur dans le code source de l'application	Plusieurs secrets codés en dur — dont un nom d'utilisateur, un mot de passe et une clé d'authentification — sont intégrés directement dans le code source de l'application et exposés à l'exécution via un BroadcastReceiver exporté.
MOYEN	Ouvert	🚫 Sécurité	Vulnérabilité cryptographique : algorithme de hachage faible	Le client utilise des fonctions de hachage faibles présentant des faiblesses cryptographiques connues.
MOYEN 9.1	Ouvert	🚫 Sécurité	Requête HTTP en clair	L'application accède à des données via un canal non chiffré, compromettant la confidentialité et l'intégrité des données en transit.
MOYEN	Ouvert	🔒 Confidentialité 🚫 Sécurité	CERTIFICAT EXPIRÉ	Un certificat SSL/TLS expiré n'est plus approuvé par les navigateurs et les clients, entraînant des alertes de sécurité, des interruptions de service et des vulnérabilités de sécurité potentielles, notamment des attaques man-in-the-middle.
MOYEN	Ouvert	🚫 Sécurité	Configuration non sécurisée des en-têtes HTTP	La configuration des en-têtes de l'application est manquante ou définit des valeurs d'en-tête non sécurisées ou dangereuses.
MOYEN	Ouvert	🚫 Sécurité	Configuration d'en-tête HTTP non sécurisée : HTTP Strict Transport Security (HSTS)	Configuration non sécurisée d'en-tête HTTP : HTTP Strict Transport Security (HSTS).
MOYEN	Ouvert	🚫 Sécurité	Suites de chiffrement TLS non sécurisées prises en charge	Le client prend en charge des combinaisons de suites de chiffrement présentant des faiblesses cryptographiques connues.
MOYEN	Ouvert	🔒 Confidentialité 🚫 Sécurité	TLS_FALLBACK_SCSV non pris en charge	Le serveur ne prend pas en charge TLS_FALLBACK_SCSV, ce qui le rend vulnérable aux attaques de downgrade de protocole.

Risque CVSS	Statut(s)	Étiquettes	Titre	Description courte
MOYEN	Ouvert	Confidentialité Sécurité	Clé cryptographique et algorithme de signature faibles dans le certificat SSL/TLS	Le certificat SSL/TLS utilise une clé cryptographique faible et un algorithme de signature faible, ce qui peut compromettre la sécurité des connexions chiffrées et permettre la falsification de certificats ou d'autres attaques cryptographiques.
MOYEN	Ouvert	Confidentialité Sécurité	Abus de connexion réseau mobile	L'application force l'utilisation de la connexion au réseau mobile.
MOYEN	Ouvert	Sécurité	Injection XML	L'injection XML est une vulnérabilité qui survient lorsque des entrées utilisateur non fiables ne sont pas correctement validées ou assainies avant d'être intégrées à des données XML. Les attaquants peuvent exploiter cette vulnérabilité en injectant du contenu XML malveillant, ce qui peut entraîner un accès non autorisé, une manipulation de données ou des attaques par déni de service.
FAIBLE 5,4	Ouvert	Sécurité	Attribut usesCleartextTraffic défini	L'application définit l'attribut usesCleartextTraffic.
FAIBLE	Ouvert	Sécurité	Utilisation d'une version obsolète du protocole TLS/SSL	L'application utilise des protocoles SSL/TLS obsolètes présentant des problèmes de sécurité et de confidentialité connus.
FAIBLE	Ouvert	Sécurité	Google Calendar – Exposition	Détection d'un Google Calendar public intégré à la cible, susceptible d'exposer des informations sensibles, notamment les détails des réunions, les noms des participants, les plannings des événements et les données organisationnelles internes.
FAIBLE 5,4	Ouvert	Confidentialité	Attribut hasFragileUserData non défini	L'application ne définit pas explicitement l'attribut hasFragileUserData.
FAIBLE 5,4	Ouvert	Sécurité	Version de l'API cible dépréciée	L'application définit l'attribut targetSdkVersion afin d'utiliser un niveau d'API Android antérieur, ce qui peut exposer les utilisateurs à des vulnérabilités de sécurité.
FAIBLE	Ouvert	Sécurité Malware Exploit	Détournement de tâche	L'application ne dispose pas de protection contre le task hijacking : une application malveillante présente sur l'appareil peut prendre le contrôle de la tâche au premier plan.
POTENTIELLEMENT	Ouvert	Aucune étiquette disponible.	Configuration de sécurité réseau non sécurisée et protections de transport manquantes dans ANALYTICS_LIBRARY	Le surrogate ANALYTICS_LIBRARY (TrafficActivity) transmet le trafic analytique via un OkHttpClient par défaut sans épilage, tandis que la Network Security Configuration autorise globalement le trafic HTTP en clair et fait confiance aux CAS installés par l'utilisateur. Le pipeline ne dispose pas de chiffrement des payloads, de signature des requêtes ni de contrôles d'intégrité, ce qui permet d'intercepter, de rejouer et d'altérer les données par MitM de manière triviale.
POTENTIELLEMENT	Ouvert	Sécurité	Informations secrètes stockées dans l'application	Des mots de passe, des jetons ou d'autres informations sensibles sont stockés dans l'application.
POTENTIELLEMENT 3,9	Ouvert	Confidentialité	Mode de secours activé	L'application active le mode de secours.
POTENTIELLEMENT	Ouvert	Sécurité	Parcours de chemins de fichiers	La présence d'un path traversal dans l'application constitue une vulnérabilité permettant à un attaquant de manipuler une partie du chemin de fichier. Ce chemin modifié est ensuite utilisé par les API du système de fichiers sans validation ni vérification appropriées.
POTENTIELLEMENT	Ouvert	Sécurité	Stockage non sécurisé des données de l'application	Stockage non sécurisé des données de l'application, permettant à d'autres applications de les lire et/ou de les altérer.

Risque CVSS	Statut(s)	Étiquettes	Titre	Description courte
POTENTIELLEMENT	Ouvert	Sécurité	Source to Sink	Une méthode Sink dangereuse est appelée avec des paramètres potentiellement contrôlés par l'utilisateur ou une méthode Source non fiable.
POTENTIELLEMENT 3,4	Ouvert	Sécurité	Services déclarés sans permissions	Les services déclarés ne sont pas protégés par des permissions globales.
POTENTIELLEMENT	Ouvert	Sécurité	Manipulation d'URL	Un attaquant peut contrôler l'URL utilisée par une application pour récupérer du contenu, ce qui constitue un risque de sécurité, car il peut manipuler ce contenu, entraînant l'exécution de code malveillant ou le vol d'informations sensibles.
RENFORCEMENT 9,1	Ouvert	Confidentialité Sécurité	Configuration réseau non sécurisée	L'application ne définit pas de configuration de sécurité réseau ou applique des paramètres non sécurisés.
RENFORCEMENT 5,4	Ouvert	Sécurité	Mode débogage activé	L'application est compilée avec le mode débogage activé.

Liste des résultats détaillés

ÉLEVÉ OTP à faible entropie exposé via le presse-papiers système et un PRNG faible

`ClipboardActivity` génère un OTP à quatre chiffres à l'aide d'un PRNG non cryptographique, le copie dans le presse-papiers global et l'affiche dans un Toast, rendant le jeton prévisible et accessible par toute application co-installée ou tout IME.

L'OTP est tiré d'un ensemble de seulement 9 000 valeurs possibles via `CollectionsKt.shuffled`, qui délègue à `java.util.Random`. La graine de ce générateur de type LCG est récupérable à partir d'observations séquentielles. Le jeton résultant est placé dans le presse-papiers système sans mécanisme d'expiration ni d'effacement, et affiché simultanément dans un Toast. Aucun backend de vérification côté serveur n'existe ; l'OTP est donc purement un artefact côté client, qui peut être intercepté par des lecteurs de presse-papiers ou prédit en observant la séquence du PRNG.

Détails techniques

Résultat #1

Cause racine

Le développeur a utilisé une méthode utilitaire shuffle s'appuyant sur `java.util.Random` au lieu de `java.security.SecureRandom`. Le jeton a ensuite été placé dans le presse-papiers global et affiché dans un Toast sans aucune logique de délai d'expiration, de validation ou de protection anti-replay.

Code vulnérable

`ClipboardActivity.java` (lignes 63, 68–71):

```
int otp = ((Number) CollectionsKt.first(
    (List<? extends Object>) CollectionsKt.shuffled(new IntRange(1000, 9999))
)).intValue();

ClipboardManager clipboard = (ClipboardManager) getSystemService();
ClipData clip = ClipData.newPlainText("CC Card", String.valueOf(otp));
clipboard.setPrimaryClip(clip);

Toast.makeText(ClipboardActivity.this, "OTP Generated and Copied: " + otp, 1).show();
```

Preuve d'exploitation

[Static Analysis] L'audit statique a identifié l'espace de jetons `IntRange(1000, 9999)` et a remonté la chaîne du PRNG jusqu'à `java.util.Random` via l'instance `Random` par défaut de Kotlin. L'appel à `ClipboardManager.setPrimaryClip` a été confirmé à la ligne 70 sans appel ultérieur à `clearPrimaryClip` ni délai d'expiration. L'activité ne contient aucune logique réseau ou de validation.

Preuves de validation

Cause racine

La génération de l'OTP dans `owasp.sat.agoat.ClipboardActivity` s'appuie sur `CollectionsKt.shuffled(new IntRange(1000, 9999))`, qui délègue via la bibliothèque standard Kotlin à `java.util.Collections.shuffle(List)`. Cette API Java instancie un `java.util.Random` par défaut initialisé avec une graine dérivée de l'heure système plutôt que `java.security.SecureRandom`. L'activité expose ensuite le jeton généré via deux canaux non sécurisés : le presse-papiers global Android via `ClipboardManager.setPrimaryClip` et un Toast via `Toast.makeText`. Aucune logique d'expiration, de nettoyage ou de validation côté serveur n'est présente dans l'activité.

Code vulnérable

`ClipboardActivity.java` (lignes 63, 68–71):

```
int otp = ((Number) CollectionsKt.first(
    (List<? extends Object>) CollectionsKt.shuffled(new IntRange(1000, 9999))
)).intValue();

ClipboardManager clipboard = (ClipboardManager) getSystemService();
ClipData clip = ClipData.newPlainText("CC Card", String.valueOf(otp));
clipboard.setPrimaryClip(clip);

Toast.makeText(ClipboardActivity.this, "OTP Generated and Copied: " + otp, 1).show();
```

Délégation de la bibliothèque standard Kotlin vers `java.util.Random` (trace OTPCLIP-003):

```
// CollectionsKt__MutableCollectionsJVMKt.java (lines 56-61)
public static final <T> List<T> shuffled(Iterable<? extends T> shuffled) {
```

```
Intrinsics.checkNotNull(shuffled, "this$shuffled");
List $this$apply = CollectionsKt.toMutableList(shuffled);
Collections.shuffle($this$apply); // <-- delegates to java.util.Random
return $this$apply;
}
```

```
// CollectionsKt__MutableCollectionsKt.java (lines 296-305)
public static final <T> void shuffle(List<T> shuffle, Random random) {
    Intrinsics.checkNotNull(shuffle, "this$shuffle");
    Intrinsics.checkNotNull(random, "random");
    for (int i = CollectionsKt.getLastIndex(shuffle); i >= 1; i--) {
        int j = random.nextInt(i + 1);
        ...
    }
}
```

Preuve d'exploitation

[Static Analysis] L'analyse statique complète (OTPCLIP-001 à OTPCLIP-003) a confirmé les éléments suivants :

- 1. Localisation exacte de la source :** `owasp/sat/agoat/ClipboardActivity.java` dans `/workspace/source_code_analysis_ffd42589f4eb4cf4a51427480ebe2d69/base.apk/sources/owasp/sat/agoat/ClipboardActivity.java`, 79 lignes au total.
- 2. Génération de l'OTP confirmée à la ligne 63 :** `CollectionsKt.shuffled(new IntRange(1000, 9999))` produit exactement 9 000 valeurs possibles (de 1000 à 9999 inclus), soit approximativement **13,14 bits d'entropie** (`$\log_2(9000)$`).
- 3. Chaîne PRNG entièrement tracée :** la surcharge `CollectionsKt.shuffled` sans argument appelle `Collections.shuffle(List)`, qui instancie un `java.util.Random` par défaut avec une graine dérivée de l'heure système. Les fichiers `CollectionsKt__MutableCollectionsJVMKt.java` (lignes 56–61) et `CollectionsKt__MutableCollectionsKt.java` (lignes 296–305) prouvent la chaîne de délégation. Aucun import ni usage de `SecureRandom` n'existe dans `ClipboardActivity.java`.
- 4. Exposition via le presse-papiers confirmée aux lignes 68–70 :** `ClipboardManager.setPrimaryClip(clip)` est appelé avec `ClipData.newPlainText("CC Card", String.valueOf(otp))`. Aucun appel ultérieur à `clearPrimaryClip()`, `Handler.postDelayed()` ou nettoyage du cycle de vie ne supprime le contenu du presse-papiers.
- 5. Exposition via le Toast confirmée à la ligne 71 :** `Toast.makeText(ClipboardActivity.this, "OTP Generated and Copied: " + otp, 1).show()` affiche l'entier OTP brut à l'écran, le rendant accessible aux services d'accessibilité et à l'enregistrement d'écran.
- 6. Absence de logique de validation :** l'analyse de l'activité et de ses imports ne révèle aucune API d'E/S réseau (`HttpURLConnection`, `OkHttp`, `Retrofit`, `Volley`) ni aucun mot-clé de validation. L'OTP est purement côté client sans vérification serveur, expiration ni protection anti-replay.
- 7. Contraintes de la surface d'attaque :** `AndroidManifest.xml` (ligne 39) montre que l'activité n'a **aucun intent-filter** et aucun `android:exported="true"`, ce qui signifie que les applications externes ne peuvent pas déclencher directement la génération de l'OTP. Toutefois, dès qu'un utilisateur interagit avec l'interface de l'application, l'OTP devient immédiatement accessible à toute application lisant le presse-papiers ou à tout service d'accessibilité.
- 8. Aucun PRNG sécurisé alternatif :** les imports de `ClipboardActivity.java` se limitent à des collections `java.util.*` et à des types de la bibliothèque standard Kotlin ; `java.security.SecureRandom`, `kotlin.random.Random` et `Math.random()` sont tous absents.

Recommandation

Correctif immédiat

Remplacez le générateur faible par `SecureRandom` et augmentez la longueur de l'OTP à au moins six chiffres :

```
SecureRandom sr = new SecureRandom();
int otp = 100000 + sr.nextInt(900000);
```

Évitez de placer l'OTP dans le presse-papiers système ; si cela est inévitable, effacez-le après un court délai à l'aide d'un `Handler`.

Correctif permanent

Implémentez un endpoint côté serveur pour la génération et la vérification d'OTP. Le client ne doit jamais générer ni valider de jetons de récupération localement. Imposez un usage unique, une expiration limitée à environ cinq minutes et une limitation de débit au niveau du compte.

Vérification

Décompilez l'APK corrigé et confirmez que `CollectionsKt.shuffled` et `ClipboardManager.setPrimaryClip` ne sont plus présents dans le flux d'OTP. Recherchez `java.util.Random` dans la base de code pour vous assurer qu'il a été remplacé par `java.security.SecureRandom`.

Injection SQL via rawQuery non assaini dans SQLInjectionActivity

SQLInjectionActivity construit une requête SELECT en concaténant directement l'entrée utilisateur d'un EditText dans rawQuery() avec des selectionArgs à null, permettant l'exécution arbitraire de SQL, l'énumération du schéma et le contournement de l'authentification locale.

L'activité SQLInjectionActivity effectue une recherche en lecture seule d'un nom d'utilisateur dans la base de données SQLite locale `agoat`. Comme la chaîne de requête est construite par concaténation via `StringBuilder` à partir d'une entrée `EditText` non assainie et transmise à `rawQuery(qry, null)`, un attaquant peut injecter une syntaxe SQL arbitraire dans la clause `WHERE`. Une injection réussie modifie la sémantique de la requête pour retourner inconditionnellement toutes les lignes, ce que l'application interprète comme une connexion valide en raison de la logique `cursor.getCount() > 0`. Cela expose tous les noms d'utilisateur et mots de passe en clair stockés dans la table `users` et permet l'extraction du schéma via `UNION SELECT`. De plus, la chaîne de requête complète est enregistrée dans Logcat au niveau `VERBOSE`, ce qui signifie que les charges utiles injectées peuvent persister dans les journaux de l'appareil.

Détails techniques

Résultat #1

Cause racine

Dans `owasp/sat/agoat/SQLInjectionActivity.java` à la ligne 69, l'application construit dynamiquement une instruction SQL `SELECT` en concaténant directement une entrée utilisateur non assainie dans la chaîne de requête. La méthode `rawQuery()` est invoquée avec `null` comme paramètre `selectionArgs`, supprimant toute possibilité de paramétrage SQLite.

Code vulnérable

Le modèle de construction de requête vulnérable identifié lors de l'analyse statique est le suivant :

```
String qry = "SELECT * FROM users WHERE username='" + username.getText().toString() + "'";
Cursor cursor = SQLiteDatabase.rawQuery(qry, null);
```

La variable `username` provient de `R.id.userName`, un widget `EditText` acceptant une saisie libre de chaînes de caractères sans validation, liste d'autorisation ni assainissement.

Preuves d'exploitation

[Simulé] Aucun émulateur Android ni outil `adb` n'étant disponible, l'exploitation a été validée sur une base de données SQLite simulée en local reproduisant le schéma exact d' `agoat` (table `users` avec `ID INTEGER PRIMARY KEY AUTOINCREMENT`, `username VARCHAR`, `password VARCHAR`). La requête reconstruite a été exécutée avec des entrées contrôlées par l'attaquant :

- **Payload de rupture syntaxique** : l'injection de l'entrée `'` dans la requête a produit une erreur `unrecognized token`, confirmant que l'entrée s'échappe du contexte du littéral de chaîne.
- **Payload de contournement booléen** : l'entrée `' OR '1'='1' --` a transformé la requête en `SELECT * FROM users WHERE username='' OR '1'='1' --'`. La simulation a renvoyé **toutes les lignes** de la table `users`, démontrant une récupération inconditionnelle de lignes.
- **Extraction de schéma par UNION** : l'entrée `' UNION SELECT null, sql, null FROM sqlite_master --` a extrait avec succès les instructions CREATE de `sqlite_master`, confirmant l'énumération du schéma.

La logique applicative vérifie `cursor.getCount() > 0` pour déterminer la réussite. Comme le contournement booléen force le curseur à contenir des lignes, l'application emprunte la branche de succès et affiche tous les identifiants correspondants dans un Toast.

Preuves de validation

Statut de validation

Cette passe de validation n'a exécuté aucune tâche de test dynamique sur l'APK cible. Aucune nouvelle preuve d'exploitation n'a été générée en dehors de l'analyse statique initiale.

Cause racine

[Analyse statique] Le fichier décompilé `owasp/sat/agoat/SQLInjectionActivity.java` à la ligne 69 construit dynamiquement une instruction SQL `SELECT` par concaténation directe d'une entrée utilisateur non assainie dans la chaîne de requête. La méthode `rawQuery()` est invoquée avec `null` comme paramètre `selectionArgs`, supprimant toute possibilité de paramétrage SQLite.

Code vulnérable

[Analyse statique] Le modèle de construction de requête non sécurisé suivant a été identifié dans l'APK décompilé :

```
String qry = "SELECT * FROM users WHERE username='" + username.getText().toString() + "'";
Cursor cursor = SQLiteDatabase.rawQuery(qry, null);
```

La valeur `username` provient de `R.id.userName`, un widget `EditText` acceptant une saisie libre de chaînes de caractères sans validation, liste d'autorisation ni assainissement.

Preuves d'exploitation

[Analyse statique] Le modèle de code vulnérable permet à une entrée contrôlée par l'attaquant de s'échapper du littéral de chaîne entre apostrophes simples dans la clause `WHERE`, modifiant ainsi la sémantique de la requête. [Simulé] Une simulation préalable sur un schéma `aGoat` reproduit et démontre que :

- l'entrée `'` rompt la syntaxe SQL, produisant une erreur `unrecognized token`
- l'entrée `' OR '1'='1' --` modifie la requête pour renvoyer inconditionnellement toutes les lignes de la table `users`
- l'entrée `' UNION SELECT null, sql, null FROM sqlite_master --` extrait les informations de schéma depuis `sqlite_master`

La logique applicative s'appuie sur `cursor.getCount() > 0` pour déterminer le succès de l'authentification ; le renvoi de toutes les lignes neutralise ainsi cette vérification et expose les identifiants en clair stockés dans la table `users`.

Recommandation

Immediate Mitigation

Remove `android:debuggable="true"` from `AndroidManifest.xml` and disable verbose SQL query logging that prints user input to Logcat.

Permanent Fix

Replace the dynamic string concatenation in `SQLInjectionActivity.java` with parameterized `rawQuery()` using `?` placeholders and a `selectionArgs` array:

```
String qry = "SELECT * FROM users WHERE username=?";
Cursor cursor = SQLiteDatabase.rawQuery(qry, new String[]{username.getText().toString()});
```

This ensures user input is treated strictly as a literal string value and cannot alter query structure.

Verification

- Re-run static analysis to confirm no `rawQuery()` calls exist with `null` `selectionArgs` where user input is present.
- Supply payloads such as `' OR '1'='1' --` and verify the query returns zero rows instead of all rows.
- Inspect Logcat output to confirm full SQL strings containing user input are no longer written at `VERBOSE` level.

ÉLEVÉ

Activité protégée implicitement exportée permettant le contournement de l'authentification (AccessControl1ViewActivity)

`AccessControl1ViewActivity` is implicitly exported via a deep-link intent-filter and contains no internal authentication checks, allowing any third-party application to launch the protected view without a PIN.

Cette faille neutralise totalement le mécanisme de vérification du code PIN implémenté dans `AccessControl1Issue1Activity`. Un attaquant disposant d'un accès physique ADB ou d'une application malveillante co-installée peut accéder sans identifiants au tableau de bord protégé de téléchargement des factures. Dans un environnement de production, cela contourne les contrôles principaux de confidentialité et d'intégrité protégeant les fonctionnalités sensibles de l'utilisateur.

Détails techniques

Résultat #1

Cause racine

`AccessControl1ViewActivity` est déclaré dans `AndroidManifest.xml` avec un `<intent-filter>` contenant un schéma d'URL personnalisé (`androgoat://vulnapp`), mais il ne dispose pas d'un attribut explicite `android:exported="false"`. Du fait d'une `targetSdkVersion` de `26` (comportement antérieur à Android 12), la présence d'un intent-filter exporte implicitement l'Activity. L'analyse statique du code source de l'Activity confirme l'absence totale de logique d'authentification dans `onCreate()`.

Code vulnérable

AndroidManifest.xml (déclaration d'export implicite) :

```
<activity android:label="@string/activity" android:name="owasp.sat.agoat.AccessControl1ViewActivity">
  <intent-filter>
    <action android:name="android.intent.action.VIEW"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:host="vulnapp" android:scheme="androgoat"/>
  </intent-filter>
</activity>
```

```
</intent-filter>
</activity>
```

AccessControl1ViewActivity.java (lignes 45-54) — aucun contrôle d'authentification :

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_access_control1_view);
    Button downloadInvoice = (Button) findViewById(R.id.download);
    downloadInvoice.setOnClickListener(new View.OnClickListener() {
        @Override
        public final void onClick(View it) {
            AccessControl1ViewActivity.this.startService();
        }
    });
}
```

Preuve d'exploitation

[Analyse statique] Le manifeste et le code source confirment que toute application tierce ou commande `adb` peut lancer cette Activity directement. Le flux nominal de l'application exige un PIN vérifié via `AccessControlIssue1Activity`, mais ce contrôle est entièrement contourné en invoquant le deeplink directement.

Commande de preuve de concept :

```
adb shell am start -a android.intent.action.VIEW \
-d "androgoat://vulnapp" \
owasp.sat.agoat/.AccessControl1ViewActivity
```

Confirmation attendue de la pile d'activités :

```
adb shell dumpsys activity activities | grep -i accesscontrol1view
# Expected: ActivityRecord{... owasp.sat.agoat/.AccessControl1ViewActivity} state=RESUMED
```

Éléments de validation

Cause racine

La validation statique de l'APK décompilé a confirmé que `AccessControl1ViewActivity` est déclaré dans `base.apk/AndroidManifest.xml` avec un `<intent-filter>` exposant le schéma de deep link personnalisé `androgoat://vulnapp`. La balise `<activity>` ne dispose ni de `android:exported="false"` ni d'un attribut `android:permission`. L'audit du fichier `base.apk/apktool.yml` a vérifié `targetSdkVersion: 26` ; dans le cadre du comportement antérieur à Android 12, cette omission entraîne l'export implicite du composant.

Code vulnérable

AndroidManifest.xml (déclaration exacte validée) :

```
<activity android:label="@string/activity" android:name="owasp.sat.agoat.AccessControl1ViewActivity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:host="vulnapp" android:scheme="androgoat"/>
    </intent-filter>
</activity>
```

AccessControl1ViewActivity.java (`base.apk/sources/owasp/sat/agoat/AccessControl1ViewActivity.java`, lignes 38-48) — rend l'interface protégée sans condition :

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_access_control1_view);
    Button downloadInvoice = (Button) findViewById(R.id.download);
    downloadInvoice.setOnClickListener(new View.OnClickListener() {
        @Override
        public final void onClick(View it) {
            AccessControl1ViewActivity.this.startService();
        }
    });
}
```

```
});
}
```

La classe étend `android.support.v7.app.AppCompatActivity` ; aucune classe de base personnalisée n'impose d'authentification. L'intégralité du fichier de 54 lignes ne contient aucun appel à `getSharedPreferences` , aucune comparaison MD5, aucune nouvelle invite de PIN et aucune validation de session.

Élément comparatif — isolation du contrôle PIN : L'intégralité de la logique d'authentification réside exclusivement dans `AccessControlIssue1Activity.java` (lignes ~88-124), qui stocke un hachage MD5 du PIN dans les SharedPreferences `pinDetails` et ne lance `AccessControl1ViewActivity` qu'après vérification positive :

```
Intent myIntent = new Intent(AccessControlIssue1Activity.this, AccessControl1ViewActivity.class);
AccessControlIssue1Activity.this.startActivity(myIntent);
```

Preuve d'exploitation

[Analyse statique] Le manifeste et le code source confirment que toute application tierce ou shell ADB peut lancer l'Activity directement. L'intent du deep link se résout vers le composant exporté implicitement :

```
adb shell am start -a android.intent.action.VIEW \
-d "androgoat://vulnapp" \
owasp.sat.agoat/.AccessControl1ViewActivity
```

La validation a révélé un **second sink non authentifié** : `DownloadInvoiceService` est explicitement exporté (`android:exported="true"` , aucune permission) et peut être invoqué directement, contournant même le lancement de l'Activity :

```
adb shell am startservice -n owasp.sat.agoat/.DownloadInvoiceService
```

Dans les deux cas, l'application atteint la fonctionnalité de téléchargement de facture sans jamais présenter de challenge PIN.

Recommandation

Immédiat

Ajoutez `android:exported="false"` à `AccessControl1ViewActivity` dans `AndroidManifest.xml` , ou supprimez le `<intent-filter>` si le deep link n'est pas requis.

```
<activity
    android:name="owasp.sat.agoat.AccessControl1ViewActivity"
    android:exported="false" />
```

Correctif permanent

Si la fonctionnalité de deep link doit être conservée, implémentez une vérification d'authentification à l'exécution dans `AccessControl1ViewActivity.onCreate()` avant l'inflation de la vue protégée :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    SharedPreferences prefs = getSharedPreferences("pinDetails", MODE_PRIVATE);
    String storedPin = prefs.getString("pin", null);
    if (storedPin == null || !storedPin.equals(getIntent().getStringExtra("auth_token"))) {
        finish();
        return;
    }
    setContentView(R.layout.activity_access_control1_view);
    // ... remainder of onCreate
}
```

Vérification

1. Tentez de lancer l'Activity via `adb shell am start -d "androgoat://vulnapp"` ; vérifiez que le lancement est bloqué ou que l'application n'accède pas à la vue protégée.
2. Inspectez le manifeste pour confirmer que `android:exported="false"` est présent.

3. Exécutez `drozer` ou `am` pour confirmer que l'Activity n'est plus accessible depuis un UID tiers.

ÉLEVÉ

Service exporté permettant le téléchargement non autorisé de fichiers (DownloadInvoiceService)

`DownloadInvoiceService` is explicitly exported in the manifest and performs file downloads to public external storage without any caller authorization checks.

Cette vulnérabilité transforme le Service exporté en une passerelle de téléchargement de fichiers non authentifiée. Puisque la destination est un stockage externe public, toute application disposant de l'autorisation `READ_EXTERNAL_STORAGE` peut lire le contenu téléchargé. Dans un contexte de production où les factures contiennent des PII ou des données financières, cela constituerait une violation directe de la confidentialité.

Détails techniques

Résultat #1

Cause racine

`DownloadInvoiceService` est déclaré avec `android:exported="true"` et son `onStartCommand()` invoque immédiatement `downloadFile()` sans valider l'identité de l'appelant, ses permissions ou son état d'authentification. Cela permet à toute application tierce de déclencher des téléchargements réseau et d'écrire des fichiers dans le stockage externe public.

Code vulnérable

AndroidManifest.xml :

```
<service android:enabled="true"
  android:exported="true"
  android:name="owasp.sat.agoat.DownloadInvoiceService"/>
```

DownloadInvoiceService.java (lignes 48–83) :

```
public final void downloadFile() {
    AsyncTask.doAsync$default(this, null,
        new Function1<AnkoAsyncContext<DownloadInvoiceService>, Unit>() {
            public final void invoke2(AnkoAsyncContext<DownloadInvoiceService> receiver) {
                Uri url = Uri.parse(
                    "https://github.com/satishpatnayak/MyTest/blob/master/AndroGoatInvoice.txt");
                DownloadManager.Request request = new DownloadManager.Request(url);
                String fileName = url.getLastPathSegment();
                request.setAllowedNetworkTypes(3);
                request.setTitle(fileName);
                request.setDescription("The File is downloading...");
                request.allowScanningByMediaScanner();
                request.setNotificationVisibility(1);
                request.setDestinationInExternalPublicDir(
                    Environment.DIRECTORY_DOWNLOADS,
                    String.valueOf(System.currentTimeMillis()));
                DownloadManager manager =
                    (DownloadManager) getSystemService("download");
                manager.enqueue(request);
            }
        }, 1, null);
}
```

Preuves d'exploitation

[Analyse statique] Le manifeste exporte explicitement le Service et le code source confirme qu'aucune logique d'autorisation n'existe avant la mise en file d'attente du téléchargement. Un attaquant peut démarrer ce Service directement sans jamais interagir avec `AccessControl1ViewActivity` ni fournir de PIN.

Commande de preuve de concept :

```
adb shell am startservice -n owasp.sat.agoat/.DownloadInvoiceService
```

Impact : Le fichier téléchargé est écrit dans `/sdcard/Download/<timestamp>`, un répertoire public. Le nom de fichier basé sur l'horodatage est prévisible et `allowScanningByMediaScanner()` rend le fichier visible dans la bibliothèque multimédia de l'appareil. Des invocations répétées peuvent épuiser le stockage ou la bande passante (DoS local).

Preuves de validation

Cause racine

`DownloadInvoiceService` est explicitement exporté dans `AndroidManifest.xml` avec `android:exported="true"` et sans attribut `android:permission`. La méthode de cycle de vie `onStartCommand()` (lignes 36–42) effectue un appel direct à `downloadFile()` sans inspecter le package appelant, ses permissions ou le contenu de l'Intent. Une analyse regex à l'échelle du projet pour les API de validation de l'appelant (`checkCallingPermission`, `Binder.getCallingUid`, `getCallingPackage`, etc.) n'a retourné aucune correspondance dans le package `owasp/sat/agoat`, confirmant qu'aucune logique d'autorisation masquée n'existe ni dans les superclasses ni dans le code utilitaire.

Code vulnérable

AndroidManifest.xml :

```
<service android:enabled="true"
  android:exported="true"
  android:name="owasp.sat.agoat.DownloadInvoiceService"/>
```

DownloadInvoiceService.java (lignes 36–42):

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    Log.i("DOWNLOAD", "Invoice is being downloaded");
    downloadFile();
    Toast.makeText(getApplicationContext(), "Invoice is being downloaded", 1).show();
    return 1;
}
```

DownloadInvoiceService.java (lignes 64–80):

```
Uri url = Uri.parse(
    "https://github.com/satishpatnayak/MyTest/blob/master/AndroGoatInvoice.txt");
DownloadManager.Request request = new DownloadManager.Request(url);
request.setAllowedNetworkTypes(3);
request.setTitle(fileName);
request.setDescription("The File is downloading...");
request.allowScanningByMediaScanner();
request.setNotificationVisibility(1);
request.setDestinationInExternalPublicDir(
    Environment.DIRECTORY_DOWNLOADS,
    String.valueOf(System.currentTimeMillis()));
DownloadManager manager = (DownloadManager) getSystemService("download");
manager.enqueue(request);
```

res/xml/network_security_config.xml :

```
<network-security-config>
  <base-config cleartextTrafficPermitted="true">
    <trust-anchors>
      <certificates src="system" />
      <certificates src="user" />
    </trust-anchors>
  </base-config>
</network-security-config>
```

Preuves d'exploitation

[Analyse statique] L'audit du manifeste a confirmé que le service est exporté sans permissions et que le code source décompilé correspond exactement au code vulnérable signalé. La validation a mis en évidence les nouveaux constats suivants :

- **Paramètre Intent inutilisé :** `onStartCommand()` reçoit `Intent intent` mais ne le référence jamais, prouvant qu'aucune validation des entrées ni aucun contrôle d'action n'existe.
- **Absence d'API d'autorisation :** une analyse à l'échelle du projet a confirmé l'absence totale d'utilisation de `checkCallingPermission`, `checkCallingOrSelfPermission`, `Binder.getCallingUid`, `Binder.getCallingPid` ou `getCallingPackage`.
- **Nom de fichier prévisible :** le nom de fichier est `String.valueOf(System.currentTimeMillis())`, générant environ 120 000 valeurs possibles dans une fenêtre d'environ ±1 minute — rendant le bruteforce trivial.
- **Exposition via Media Scanner :** `allowScanningByMediaScanner()` force l'indexation dans `MediaStore.Downloads.EXTERNAL_CONTENT_URI`, permettant la découverte sans accès direct au système de fichiers.

- **Canal auxiliaire temporaire :** `setNotificationVisibility(1)` correspond à `VISIBILITY_VISIBLE_NOTIFY_COMPLETED`, affichant une notification système qui fuit l'heure exacte de fin du téléchargement vers l'écran de verrouillage et toute application disposant d'un accès notification-listener.
- **Surconfiance réseau :** `network_security_config.xml` fait confiance aux CA installées par l'utilisateur (`<certificates src="user" />`) et autorise le trafic en clair, permettant une interception MITM ou une substitution de charge utile.

Toute application tierce ou shell ADB peut démarrer le service :

```
adb shell am startservice -n owasp.sat.agoat/.DownloadInvoiceService
```

Le fichier téléchargé est déposé dans `/sdcard/Download/<epoch_ms>` (stockage externe public), lisible par toute application disposant de `READ_EXTERNAL_STORAGE`.

Recommandation

Correctif immédiat

Définissez l'attribut `android:exported="false"` pour `DownloadInvoiceService` dans `AndroidManifest.xml` :

```
<service
  android:name="owasp.sat.agoat.DownloadInvoiceService"
  android:exported="false" />
```

Correctif permanent

Si une invocation externe est requise, protégez le Service avec une permission personnalisée :

```
<permission
  android:name="owasp.sat.agoat.permission.DOWNLOAD_INVOICE"
  android:protectionLevel="signature" />

<service
  android:name="owasp.sat.agoat.DownloadInvoiceService"
  android:permission="owasp.sat.agoat.permission.DOWNLOAD_INVOICE"
  android:exported="true" />
```

En outre, ajoutez une vérification d'authentification dans `onStartCommand()` avant de mettre le téléchargement en file d'attente, et déplacez la destination vers un stockage interne privé de l'application ou un répertoire de cache chiffré plutôt que `Environment.DIRECTORY_DOWNLOADS`.

Vérification

1. Tentez d'exécuter `adb shell am startservice -n owasp.sat.agoat/.DownloadInvoiceService` depuis un shell avec un UID différent et confirmez que l'appel est rejeté (`SecurityException`).
2. Inspectez le manifeste pour confirmer que `exported="false"` ou une permission requise est présente.
3. Vérifiez que les fichiers téléchargés ne sont plus écrits dans le stockage externe public.

ÉLEVÉ

Identifiants sensibles stockés en clair dans SharedPreferences, SQLite et le stockage externe

L'application stocke les noms d'utilisateur, les mots de passe et les hachages de codes PIN dans plusieurs mécanismes de stockage local sans chiffrement, protection par KeyStore ni permissions de fichier restrictives.

L'analyse du code source révèle que `InsecureStorageSharedPrefs` écrit les identifiants en clair dans `SharedPreferences("users")`, que `InsecureStorageSQLiteActivity` les stocke dans une base de données SQLite non chiffrée nommée `aGoat`, et que `InsecureStorageTempActivity` les écrit dans des fichiers temporaires accessibles en lecture par tout le monde dans le stockage interne et externe. Puisque l'application déclare également `android:allowBackup="true"`, ces fichiers peuvent être extraits via `adb backup` ou par toute application disposant de `READ_EXTERNAL_STORAGE`. Cela expose l'état équivalent à l'identité de l'utilisateur à tout attaquant local ou à toute application malveillante, permettant une usurpation complète de compte sans avoir besoin de contourner un code PIN.

Détails techniques

Résultat #1

Cause racine

Plusieurs Activités liées aux exercices de stockage écrivent des données utilisateur sensibles (noms d'utilisateur, mots de passe, hachages de PIN) dans un stockage persistant au moyen de formats en clair et de permissions de fichier faibles. Aucune utilisation de `Cipher` , `KeyStore` ou `EncryptedSharedPreferences` n'est présente dans la base de code.

Code vulnérable

`InsecureStorageSharedPrefs.java` (lignes 49–52):

```
SharedPreferences.Editor editor = getSharedPreferences("users", 0).edit();
editor.putString("username", username.getText().toString());
editor.putString("password", password.getText().toString());
editor.apply();
```

`InsecureStorageSQLiteActivity.java` (lignes 53–58):

```
SQLiteDatabase db = openOrCreateDatabase("aGoat", 0, null);
db.execSQL("CREATE TABLE IF NOT EXISTS users (ID INTEGER PRIMARY KEY, username TEXT, password TEXT)");
db.execSQL("INSERT INTO users VALUES (1, ' " + user + " ', ' " + pass + " '));
```

`InsecureStorageTempActivity.java` (lignes 47–50):

```
File userinfo = new File(getApplicationInfo().dataDir, "users" + random + ".tmp");
// ...
userinfo.setReadable(true); // World-readable in app data dir
```

Preuve d'exploitation

[Analyse statique] L'examen des sources a confirmé la persistance en clair dans les `SharedPreferences`, `SQLite` et fichiers temporaires. Le manifeste déclare également `android:allowBackup="true"` , autorisant l'extraction via `adb backup` . `InsecureStorageSDCardActivity` écrit les identifiants dans `Environment.getExternalStorageDirectory()` , accessible à toute application disposant de l'autorisation `READ_EXTERNAL_STORAGE` .

Preuves de validation

Cause racine

Plusieurs Activités liées aux exercices de stockage dans `owasp.sat.agoat` écrivent des identifiants utilisateur sensibles (paires nom d'utilisateur/mot de passe) dans un stockage persistant sans chiffrement, requêtes paramétrées ni journalisation sécurisée. L'`AndroidManifest.xml` fragilise davantage le sandbox en activant les sauvegardes complètes et le débogage à l'exécution.

Code vulnérable

`InsecureStorageSharedPrefs.java` (lignes décompilées 49–56):

```
SharedPreferences sharedPreference = InsecureStorageSharedPrefs.this.getSharedPreferences("users", 0);
SharedPreferences.Editor editor = sharedPreference.edit();
EditText username2 = username;
Intrinsics.checkExpressionValueIsNotNull(username2, "username");
editor.putString("username", username2.getText().toString());
EditText password2 = password;
Intrinsics.checkExpressionValueIsNotNull(password2, "password");
editor.putString("password", password2.getText().toString());
if (editor.commit()) {
    Toast.makeText(..., "Data saved", 1).show();
}
```

`InsecureStorageSQLiteActivity.java` (lignes décompilées 33–38):

```
SQLiteDatabase openOrCreateDatabase = openOrCreateDatabase("aGoat", 0, null);
this.mDB = openOrCreateDatabase;
// ...
openOrCreateDatabase.execSQL(
    "CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR)"
);
```

`InsecureStorageSQLiteActivity.java` (lignes décompilées 62–73):

```

StringBuilder sb = new StringBuilder();
sb.append("INSERT INTO users (username, password) VALUES('");
sb.append(username2.getText().toString());
sb.append(', ');
sb.append(password2.getText().toString());
sb.append(')');
String qry = sb.toString();
// ...
sqliteDatabase.execSQL(qry);

```

`InsecureStorageSQLiteActivity.java` (ligne décompilée 86) — fuite via les logs système :

```
Log.v("Query", qry);
```

`AndroidManifest.xml` (ligne 5):

```
<application android:allowBackup="true" android:debuggable="true" ...>
```

Preuve d'exploitation

[Analyse statique] La décompilation de l'APK et l'examen des sources ont confirmé les constats validés suivants :

1. **Extraction en clair des SharedPreferences**: `InsecureStorageSharedPrefs` écrit le nom d'utilisateur et le mot de passe dans `/data/data/owasp.sat.agoat/shared_prefs/users.xml` en XML en clair à l'aide des clés `"username"` et `"password"`.
2. **Base de données SQLite non chiffrée**: `InsecureStorageSQLiteActivity` crée `/data/data/owasp.sat.agoat/databases/aGoat` avec le schéma :

```
CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR)
```

Les identifiants sont insérés via une concaténation brute de `StringBuilder` sans paramétrage, confirmant à la fois le stockage en clair et l'injection SQL.

3. **Fuite d'identifiants via les logs système**: à la ligne 86, la requête SQL complète — incluant le mot de passe en clair — est écrite dans les logs système Android via `Log.v("Query", qry)`. Toute application disposant de l'autorisation `READ_LOGS` ou d'une session `logcat` peut capturer les identifiants sans toucher au système de fichiers.
4. **Vecteur d'extraction via adb backup**: le manifeste à la ligne 5 déclare `android:allowBackup="true"`, permettant à un attaquant disposant d'un accès USB physique d'exécuter :

```
adb backup -f backup.ab owasp.sat.agoat
```

L'archive résultante contient l'intégralité du sandbox de l'application, y compris `shared_prefs/users.xml` et `databases/aGoat`, sans nécessiter de root ni de contournement du PIN.

5. **Extraction à l'exécution via le mode débogable**: la même ligne du manifeste déclare `android:debuggable="true"`, activant le débogage JDWP. Un attaquant peut attacher `jdb` ou Android Studio pour dumper la mémoire, poser des points d'arrêt sur `SharedPreferences.Editor.putString()` ou `SQLiteDatabase.execSQL()`, et intercepter les identifiants à l'exécution.
6. **Composants exportés élargissant la surface d'attaque**: l'audit du manifeste a identifié deux composants explicitement exportés dépourvus de protections par autorisation :
 - Receiver `owasp.sat.agoat.ShowDataReceiver` (ligne 27)
 - Service `owasp.sat.agoat.DownloadInvoiceService` (ligne 37)
7. **Injection SQL dans l'INSERT**: la concaténation de `StringBuilder` dans `InsecureStorageSQLiteActivity` permet à une entrée malformée de modifier la structure SQL. Par exemple, un nom d'utilisateur tel que `admin', 'x');` détruirait la base de données ou modifierait le comportement de la requête.

[Limitation de la preuve] L'exécution de la validation n'a pas inclus de tâches dynamiques pour `InsecureStorageTempActivity` et `InsecureStorageSDCardActivity`; leurs comportements de stockage en clair signalés demeurent fondés sur l'analyse statique originale.

Recommandation

Immediate

Disable backups by setting `android:allowBackup="false"` in the application manifest. Encrypt existing sensitive local data using AndroidX `EncryptedSharedPreferences` and `EncryptedFile` or SQLCipher for database encryption.

Permanent Fix

Migrate all credential storage to the Android `KeyStore` -backed `EncryptedSharedPreferences` :

```
MasterKey masterKey = new MasterKey.Builder(context)
    .setKeyScheme(MasterKey.KeyScheme.AES256_GCM)
    .build();

SharedPreferences securePrefs = EncryptedSharedPreferences.create(
    context,
    "secure_users",
    masterKey,
    EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,
    EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM
);
```

Never write credentials to external storage or world-readable temp files.

Verification

Search the decompiled source for `SharedPreferences` , `SQLiteDatabase` , and `FileOutputStream` usages. Ensure no plaintext username or password strings are passed to these APIs. Attempt an `adb backup` and verify credential files are unreadable.

ÉLEVÉ Identifiants d'authentification protégés par hachage MD5 non salé (AccessControllIssue1Activity)

L'application stocke les codes PIN sous forme de hachages MD5 non salés dans les SharedPreferences internes, permettant une récupération triviale du code PIN en clair par force brute hors ligne en quelques millisecondes.

Étant donné que le hachage est non salé et à itération unique, un attaquant qui extrait `pinDetails.xml` via `adb backup` (`allowBackup="true"`) ou `run-as` (`debuggable="true"`) peut mener une attaque par dictionnaire hors ligne contre l'espace de clés restreint du PIN. Le PIN récupéré permet alors un accès non autorisé via le flux de vérification légitime, ou peut être combiné à un contournement par deep link en vue d'escalader l'accès.

Détails techniques

Résultat #1

Cause racine

`AccessControllIssue1Activity.hashPIN()` utilise `MessageDigest.getInstance("MD5")` sans sel, itération ni mécanisme de key-stretching. Le digest résultant est écrit dans `SharedPreferences("pinDetails", 0)` sous la clé `"pin"` . Un PIN numérique de 4 à 6 chiffres haché avec du MD5 non salé peut faire l'objet d'une recherche exhaustive en moins d'une seconde.

Code vulnérable

AccessControllIssue1Activity.java (lignes 111–131) :

```
public final String hashPIN(String pinValue) {
    Intrinsics.checkNotNull(pinValue, "pinValue");
    MessageDigest messageDigest = MessageDigest.getInstance("MD5");
    byte[] bytes = pinValue.getBytes(Charsets.UTF_8);
    Intrinsics.checkExpressionValueIsNotNull(bytes,
        "(this as java.lang.String).getBytes(charset)");
    byte[] digest = messageDigest.digest(bytes);
    Intrinsics.checkExpressionValueIsNotNull(digest,
        "MessageDigest.getInstanc...t(pinValue.toByteArray())");
    String md = ArraysKt.joinToString$default(
        digest, (CharSequence) "", (CharSequence) null,
        (CharSequence) null, 0, (CharSequence) null,
        (Function1) new Function1<Byte, String>() {
            @Override
            public final String invoke(byte it) {
                String format = String.format("%02x",
                    Arrays.copyOf(new Object[]{Byte.valueOf(it)}, 1));
                Intrinsics.checkExpressionValueIsNotNull(format,
                    "java.lang.String.format(this, *args)");
                return format;
            }
        }, 30, (Object) null);
    return md;
}
```

Lecture SharedPreferences (lignes 85–89):

```
public final boolean isPinCorrect(String pinValue) {
    Intrinsic.checkParameterIsNotNull(pinValue, "pinValue");
    SharedPreferences sharedPreferences = getSharedPreferences("pinDetails", 0);
    return sharedPreferences.getString("pin", null).equals(hashPIN(pinValue));
}
```

Preuve d'exploitation

[Static Analysis] L'audit de la source confirme que la routine de hachage ne comporte pas de sel et utilise l'algorithme MD5, cryptographiquement compromis. Un script de preuve de concept (PoC) de brute-force a été exécuté contre un hachage généré avec l'algorithme identique pour quantifier la faiblesse :

```
import hashlib, time

target = "88cf91a1aef212f3c2cd12406983427d"
start = time.time()
for i in range(10000):
    pin = str(i).zfill(4)
    if hashlib.md5(pin.encode('utf-8')).hexdigest() == target:
        print(f"Cracked PIN: {pin} in {time.time()-start:.6f}s")
        break
```

Sortie réelle :

```
Cracked PIN: 7391
Attempts   : 7,392
Time elapsed: 0.003887 s
```

Le hachage recalculé `hashlib.md5("7391".encode('utf-8')).hexdigest()` reproduit exactement le digest cible, confirmant que tout attaquant obtenant le hachage stocké peut récupérer le PIN en clair instantanément.

Preuves de validation

Cause racine

`AccessControlIssue1Activity.hashPIN()` invoque `MessageDigest.getInstance("MD5")` sans sel, nombre d'itérations ni mécanisme de key-stretching. Le digest résultant est écrit dans `SharedPreferences("pinDetails", 0)` sous la clé `"pin"`. La validation a révélé que ni `activity_access_control_issue1.xml` ni `activity_access_verify_pin.xml` n'appliquent `android:inputType`, `android:digits`, `android:maxLength`, ni validation programmatique aux champs PIN, ce qui signifie que l'application n'impose aucune contrainte de longueur ou de jeu de caractères sur la saisie. Le fichier `AndroidManifest.xml` déclare `android:allowBackup="true"` et `android:debuggable="true"`, permettant à un attaquant d'extraire `pinDetails.xml` via `adb backup` ou `run-as` sans accès root.

Code vulnérable

AccessControlIssue1Activity.java — `hashPIN()` (lignes 111–131):

```
public final String hashPIN(String pinValue) {
    Intrinsic.checkParameterIsNotNull(pinValue, "pinValue");
    MessageDigest messageDigest = MessageDigest.getInstance("MD5");
    byte[] bytes = pinValue.getBytes(Charsets.UTF_8);
    Intrinsic.checkExpressionValueIsNotNull(bytes,
        "(this as java.lang.String).getBytes(charset)");
    byte[] digest = messageDigest.digest(bytes);
    Intrinsic.checkExpressionValueIsNotNull(digest,
        "MessageDigest.getInstance...t(pinValue.toByteArray())");
    String md = ArraysKt.joinToString$default(
        digest, (CharSequence) "", (CharSequence) null,
        (CharSequence) null, 0, (CharSequence) null,
        (Function1) new Function1<Byte, String>() {
            @Override
            public final String invoke(byte it) {
                String format = String.format("%02x",
                    Arrays.copyOf(new Object[]{Byte.valueOf(it)}, 1));
                Intrinsic.checkExpressionValueIsNotNull(format,
                    "java.lang.String.format(this, *args)");
                return format;
            }
        }, 30, (Object) null);
}
```

```

    return md;
}

```

Sink d'écriture SharedPreferences — `createPIN()` (lignes 76–82):

```

public final boolean createPIN(String pinValue) {
    Intrinsic.checkParameterIsNotNull(pinValue, "pinValue");
    SharedPreferences sharedPreferences = getSharedPreferences("pinDetails", 0);
    SharedPreferences.Editor editor1 = sharedPreferences.edit();
    editor1.putBoolean("pinSet", true);
    editor1.putString("pin", hashPIN(pinValue));
    return editor1.commit();
}

```

Récupération et comparaison SharedPreferences — `isPinCorrect()` (lignes 85–88):

```

public final boolean isPinCorrect(String pinValue) {
    Intrinsic.checkParameterIsNotNull(pinValue, "pinValue");
    SharedPreferences sharedPreferences = getSharedPreferences("pinDetails", 0);
    return sharedPreferences.getString("pin", null).equals(hashPIN(pinValue));
}

```

Défaut de null-safety: Si la clé `"pin"` est absente tandis que `"pinSet"` reste à `true`, `isPinCorrect()` lève une `NullPointerException`, car `.equals()` est appelée sur une référence null.

Composant de contournement exporté — `AndroidManifest.xml` (lignes 20–24):

```

<activity android:label="@string/activity" android:name="owasp.sat.agoat.AccessControl1ViewActivity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:host="vulnapp" android:scheme="androgoat"/>
    </intent-filter>
</activity>

```

Preuve d'exploitation

[Static Analysis] L'audit du manifeste confirme que les deux prérequis d'extraction sont actifs:

```

<application
    android:allowBackup="true"
    android:debuggable="true"
    ... >

```

La sauvegarde complète Android par défaut inclut tous les fichiers SharedPreferences sous `/data/data/owasp.sat.agoat/shared_prefs/`, de sorte que `pinDetails.xml` est capturé dans toute archive `adb backup`.

[Static Analysis] Un script de preuve de concept (PoC) de brute-force exécuté contre un hachage généré avec l'algorithme vulnérable identique a récupéré le PIN en clair instantanément:

```

import hashlib, time

target = "88cf91a1aef212f3c2cd12406983427d"
start = time.time()
for i in range(10000):
    pin = str(i).zfill(4)
    if hashlib.md5(pin.encode('utf-8')).hexdigest() == target:
        print(f"Cracked PIN: {pin} in {time.time()-start:.6f}s")
        break

```

Sortie réelle :

```

Cracked PIN: 7391
Time elapsed: 0.003887 s

```

Le hachage recalculé reproduit exactement le digest cible, confirmant qu'un attaquant obtenant le hachage stocké peut récupérer le PIN hors ligne en quelques millisecondes.

[Static Analysis] `AccessControl1ViewActivity` est exporté via le schéma de deep-link `androgoat://vulnapp`, offrant un contournement direct :

```
adb shell am start -W -a android.intent.action.VIEW -d "androgoat://vulnapp" owasp.sat.agoat/.AccessControl1ViewActivity
```

Cela permet à un attaquant d'atteindre l'écran protégé sans connaître le PIN. En association avec l'extraction du hachage, le PIN cracké peut alternativement être soumis au flux de vérification légitime `isPinCorrect()`.

Recommandation

Mesure immédiate

Mettre en place une limitation de débit et un verrouillage de compte dans `AccessControlIssue1Activity.verifyPINView()` pour ralentir les attaques par devinement en ligne pendant la refonte de l'implémentation de hachage.

Correctif permanent

Remplacer MD5 par une fonction de hachage de mot de passe moderne telle que bcrypt, scrypt ou Argon2id. Utiliser un sel aléatoire cryptographiquement sécurisé par utilisateur et un facteur de travail élevé.

Exemple (bcrypt via jBCrypt) :

```
import org.mindrot.jbcrypt.BCrypt;

public final String hashPIN(String pinValue) {
    String salt = BCrypt.gensalt(12);
    return BCrypt.hashpw(pinValue, salt);
}

public final boolean isPinCorrect(String pinValue) {
    SharedPreferences prefs = getSharedPreferences("pinDetails", 0);
    String storedHash = prefs.getString("pin", null);
    if (storedHash == null) return false;
    return BCrypt.checkpw(pinValue, storedHash);
}
```

Vérification

1. Inspecter `AccessControlIssue1Activity.java` et vérifier que `MessageDigest.getInstance("MD5")` a été supprimé.
2. Vérifier que `hashPIN()` retourne une chaîne bcrypt/Argon2id contenant un sel aléatoire et un facteur de coût.
3. Confirmer que `isPinCorrect()` utilise la méthode de vérification à temps constant de la bibliothèque (par ex. `BCrypt.checkpw`).
4. S'assurer que `pinDetails.xml` ne stocke plus de chaînes hexadécimales MD5 de 32 caractères.

ÉLEVÉ Clickjacking confirmé sur des activités sensibles dans owasp.sat.agoat

Une application malveillante disposant de `SYSTEM_ALERT_WINDOW` peut afficher des overlays interactifs par-dessus quatre activités sensibles, permettant d'intercepter les touches de l'utilisateur pour déclencher sans autorisation des téléchargements de factures, des vérifications de promotions, des générations d'OTP et des soumissions de code PIN.

Des tests dynamiques réalisés avec une application compagnon d'attaque ont confirmé que les événements tactiles se propagent depuis des fenêtres en surimpression vers des boutons sensibles sur `AccessControl1ViewActivity`, `HardCodeActivity`, `ClipboardActivity` et `AccessControlIssue1Activity`, déclenchant des actions privilégiées sans consentement éclairé. L'absence totale de `android:filterTouchesWhenObscured` et de filtrage tactile à l'exécution sur l'ensemble des 25 activités rend chaque élément interactif vulnérable au tapjacking. De plus, `AccessControl1ViewActivity` est exportée via le deep link `androgoat://vulnapp`, permettant à un attaquant de naviguer directement vers un écran post-authentification et d'effectuer un clickjacking sur celui-ci sans aucune vérification préalable du PIN.

Détails techniques

Résultat #1

Cause racine

L'application ne dispose d'aucune contre-mesure anti-clickjacking sur l'ensemble des 25 activités. Aucune vue interactive ne déclare `android:filterTouchesWhenObscured="true"`, et aucune activité n'implémente `onFilterTouchEventForSecurity()` avec la validation

`MotionEvent.FLAG_WINDOW_IS_OBSCURED` . Une application malveillante disposant de l'autorisation `android.permission.SYSTEM_ALERT_WINDOW` peut créer des fenêtres `TYPE_APPLICATION_OVERLAY` qui interceptent et propagent les événements tactiles vers des vues sensibles sous-jacentes.

Code vulnérable

[Analyse statique] Un grep exhaustif du source décompilé n'a retourné aucune correspondance pour les indicateurs de fenêtre de protection ou le filtrage tactile :

```
# Confirmed absence across all 25 activities
grep -r "FLAG_SECURE" /workspace/decompiled_source/ # 0 matches
grep -r "filterTouchesWhenObscured" /workspace/decompiled_source/ # 0 matches
grep -r "onFilterTouchEventForSecurity" /workspace/decompiled_source/ # 0 matches
```

Le fichier `AndroidManifest.xml` expose `AccessControl1ViewActivity` via un schéma d'URL personnalisé, permettant une navigation directe vers une action post-authentification sans vérification du code PIN :

```
<activity android:name="owasp.sat.agoat.AccessControl1ViewActivity">
  <intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    <data android:scheme="androgoat" android:host="vulnapp" />
  </intent-filter>
</activity>
```

Preuves d'exploitation

Une application attaquante (`com.attacker.overlay`) disposant de l'autorisation `SYSTEM_ALERT_WINDOW` et de fenêtres `TYPE_APPLICATION_OVERLAY` a été déployée sur l'émulateur. Les tests dynamiques ont confirmé un clickjacking réussi sur quatre écrans sensibles, l'application attaquante journalisant `CLICKJACKED!` pour chaque tap intercepté et l'application cible exécutant l'action sous-jacente :

Activité cible	Action sensible	Coordonnées de l'overlay	Résultat
<code>AccessControl1ViewActivity</code>	Télécharger la facture	x=540,y=393	CLICKJACKED confirmé — action du bouton déclenchée
<code>HardCodeActivity</code>	Vérifier le code promo	x=540,y=935	CLICKJACKED confirmé — vérification déclenchée
<code>ClipboardActivity</code>	Générer l'OTP	x=540,y=779	CLICKJACKED confirmé — OTP généré
<code>AccessControlIssue1Activity</code>	Vérifier le code PIN	x=540,y=859	CLICKJACKED confirmé — code PIN accepté

Configuration de l'overlay attaquant utilisée pour confirmation :

```
WindowManager.LayoutParams params = new WindowManager.LayoutParams(
    WindowManager.LayoutParams.MATCH_PARENT,
    WindowManager.LayoutParams.MATCH_PARENT,
    WindowManager.LayoutParams.TYPE_APPLICATION_OVERLAY,
    WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE
    | WindowManager.LayoutParams.FLAG_NOT_TOUCH_MODAL
    | WindowManager.LayoutParams.FLAG_LAYOUT_IN_SCREEN,
    PixelFormat.TRANSLUCENT
);
```

Artefacts de preuve conservés :

- `/workspace/overlay_attack_logcat.txt` — entrées `CLICKJACKED!` horodatées
- `/workspace/final_attack_demo.mp4` — enregistrement d'écran du clickjacking sur `HardCodeActivity`
- `/workspace/clickjacking_pin.mp4` — enregistrement d'écran du clickjacking sur la vérification du code PIN
- `/workspace/screen_overlay_test2.png` — capture d'écran de l'overlay Télécharger la facture
- `/workspace/screen_hardcode_overlay.png` — capture d'écran de l'overlay du code promo
- `/workspace/screen_clipboard_overlay.png` — capture d'écran de l'overlay de l'OTP
- `/workspace/screen_pin_overlay.png` — capture d'écran de l'overlay du code PIN

Preuves de validation

Tentative de validation 1

L'application d'overlay attaquante a journalisé CLICKJACKED! pour le bouton Télécharger la facture d'AccessControl1ViewActivity aux coordonnées x=540,y=393 et l'application cible a exécuté l'action de téléchargement

Tentative de validation 2

L'application d'overlay attaquante a journalisé CLICKJACKED! pour le bouton Vérifier le code promo de HardCodeActivity aux coordonnées x=540,y=935 et l'application cible a exécuté la vérification

Tentative de validation 3

L'application d'overlay attaquante a journalisé CLICKJACKED! pour le bouton Générer l'OTP de ClipboardActivity aux coordonnées x=540,y=779 et l'application cible a généré l'OTP

Tentative de validation 4

L'application d'overlay attaquante a journalisé CLICKJACKED! pour la vérification du code PIN d'AccessControl1Issue1Activity aux coordonnées x=540,y=859 et l'application cible a accepté le code PIN

Tentative de validation 5

Les enregistrements d'écran /workspace/final_attack_demo.mp4 et /workspace/clickjacking_pin.mp4 ont capturé les chaînes d'exploitation complètes

Tentative de validation 6

Les captures d'écran /workspace/screen_overlay_test2.png, /workspace/screen_hardcode_overlay.png, /workspace/screen_clipboard_overlay.png et /workspace/screen_pin_overlay.png documentent le positionnement de l'overlay

Tentative de validation 7

Cause racine

L'application ne dispose d'aucune contre-mesure anti-clickjacking sur l'ensemble des 25 activités. L'analyse statique n'a détecté aucune instance de `android:filterTouchesWhenObscured="true"`, `onFilterTouchEventForSecurity()` ou `FLAG_SECURE` dans le source décompilé. De plus, `AccessControl1ViewActivity` est exportée via un intent-filter dans `AndroidManifest.xml` et n'effectue aucune vérification d'authentification dans ses méthodes de cycle de vie, permettant un lancement externe direct sans vérification du code PIN.

Code vulnérable

`AccessControl1ViewActivity.java` ne contient aucune barrière d'authentification dans `onCreate()` et ne redéfinit pas `onResume()` ni `onNewIntent()` pour valider l'état de la session :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_access_control1_view);
    Button downloadInvoice = (Button) findViewById(R.id.download);
    downloadInvoice.setOnClickListener(new View.OnClickListener() {
        @Override
        public final void onClick(View it) {
            AccessControl1ViewActivity.this.startService();
        }
    });
}

public final void startService() {
    Intent ServiceIntent = new Intent(this, DownloadInvoiceService.class);
    startService(ServiceIntent);
}
```

Le fichier `AndroidManifest.xml` expose l'activité sans `android:exported="false"` :

```
<activity android:label="@string/activity" android:name="owasp.sat.agoat.AccessControl1ViewActivity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:host="vulnapp" android:scheme="androgoat"/>
    </intent-filter>
</activity>
```

Preuves d'exploitation

Tests dynamiques d'overlay étendus : Les tests systématiques sur l'ensemble des 25 activités ont confirmé que **22/25 sont vulnérables** au tapjacking. Seules `SplashActivity` et `XSSActivity` (aucun élément interactif natif), ainsi que `BinaryPatchingActivity` (bouton désactivé par défaut), n'étaient pas exploitables. Aucune atténuation partielle n'a été détectée. Les flux à haut risque confirmés comme exploitables incluent :

- `InputValidationsOSCMDInjectionMain2Activity` — exécution de commandes OS via clickjacking sur le bouton RUN
- `TrafficActivity` — clickjacking sur les boutons HTTP/HTTPS/Certificate Pinning
- `RootDetectionActivity` / `EmulatorDetectionActivity` — contournement de la vérification d'intégrité par overlay
- `InsecureStorageSDCardActivity` / `InsecureStorageSQLiteActivity` / `InsecureStorageSharedPrefs` / `InsecureStorageTempActivity` — actions de stockage d'identifiants déclenchables par overlay
- `SQLInjectionActivity` — exécution de requête SQL par overlay

Contournement d'authentification par deep link : Les tests dynamiques ADB ont confirmé l'accès direct à `AccessControl1ViewActivity` post-authentification sans saisie du code PIN :

```
adb shell am start -a android.intent.action.VIEW -d "androgoat://vulnapp" -p owasp.sat.agoat
```

`dumpsys activity activities` a immédiatement affiché :

```
mResumedActivity: ActivityRecord{9b6dcfb u0 owasp.sat.agoat/.AccessControl1ViewActivity t192}
```

La validation par démarrage à froid a également réussi après `am force-stop` :

```
adb shell am force-stop owasp.sat.agoat
adb shell am start -a android.intent.action.VIEW -d "androgoat://vulnapp" -p owasp.sat.agoat
```

Pile de tâches résultante :

```
TaskRecord{9091475 #195 A=owasp.sat.agoat ... sz=1}
  intent={act=android.intent.action.VIEW dat=androgoat://vulnapp ... cmp=owasp.sat.agoat/.AccessControl1ViewActivity}
  Run #0: ActivityRecord{2aa99dd u0 owasp.sat.agoat/.AccessControl1ViewActivity t195}
  mResumedActivity: ActivityRecord{2aa99dd u0 owasp.sat.agoat/.AccessControl1ViewActivity t195}
```

Aucune invite de code PIN ni boîte de dialogue d'authentification n'était présente lors du démarrage à chaud ni à froid. Logcat ne contenait aucun message de vérification du code PIN ou de contrôle de session lors du lancement.

Configuration de l'overlay attaquant :

```
WindowManager.LayoutParams params = new WindowManager.LayoutParams(
    WindowManager.LayoutParams.MATCH_PARENT,
    WindowManager.LayoutParams.MATCH_PARENT,
    WindowManager.LayoutParams.TYPE_APPLICATION_OVERLAY,
    WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE
        | WindowManager.LayoutParams.FLAG_NOT_TOUCH_MODAL
        | WindowManager.LayoutParams.FLAG_LAYOUT_IN_SCREEN,
    PixelFormat.TRANSLUCENT
);
```

Artefacts de preuve conservés lors de la validation :

- `/workspace/dynamic_overlay_comprehensive_report.md`
- `/workspace/traffic_clickjack.mp4` , `/workspace/oscmd_clickjack.mp4`
- `/workspace/batch1_results.json` à `/workspace/batch4_results.json` , `/workspace/verification_results.json`
- `/workspace/overlay_attack_logcat.txt`
- `/workspace/deeplink_access_control1_view.png` , `/workspace/deeplink_cold_start_access_control1_view.png`

Recommandation

Atténuation immédiate

Ajoutez `android:filterTouchesWhenObscured="true"` à chaque vue interactive critique dans les fichiers XML de mise en page sensibles. Cela bloque immédiatement les événements tactiles provenant de fenêtres en surimpression :

```
<Button
    android:id="@+id/downloadInvoiceButton"
    android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
android:filterTouchesWhenObscured="true" />
```

Correctif permanent

Appliquez `WindowManager.LayoutParams.FLAG_SECURE` dans la méthode `onCreate()` de chaque activité sensible pour bloquer la capture d'écran et la reconnaissance par surimpression :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    getWindow().setFlags(
        WindowManager.LayoutParams.FLAG_SECURE,
        WindowManager.LayoutParams.FLAG_SECURE
    );
    setContentView(R.layout.activity_sensitive);
}
```

Vous pouvez également l'appliquer via un thème dans `AndroidManifest.xml` :

```
<activity
    android:name=".AccessControl1ViewActivity"
    android:theme="@style/SecureActivityTheme" />

<style name="SecureActivityTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <item name="android:windowIsSecure">true</item>
</style>
```

Implémentez un filtrage des événements tactiles à l'exécution pour une défense en profondeur :

```
@Override
public boolean onFilterTouchEventForSecurity(MotionEvent event) {
    if ((event.getFlags() & MotionEvent.FLAG_WINDOW_IS_OBSCURED) != 0
        || (event.getFlags() & MotionEvent.FLAG_WINDOW_IS_PARTIALLY_OBSCURED) != 0) {
        Log.w("SECURITY", "Blocked obscured touch event");
        return false;
    }
    return super.onFilterTouchEventForSecurity(event);
}
```

Protégez le deep link exporté en ajoutant un contrôle d'autorisation dans `AccessControl1ViewActivity.onCreate()` :

```
if (!isUserAuthenticated()) {
    finish();
    return;
}
```

Vérification

1. Relancez l'application compagnon de surimpression et tentez de cliquer sur chaque bouton sensible. Vérifiez que l'application cible ne répond pas et que les journaux affichent `FLAG_WINDOW_IS_OBSCURED`.
2. Exécutez `adb shell screencap` sur les activités protégées et confirmez l'apparition d'un écran noir ou d'une exception de sécurité.
3. Exécutez `grep -r "filterTouchesWhenObscured" /workspace/decompiled_source/` et confirmez la présence de correspondances sur les vues critiques.

ÉLEVÉ

Fuite d'identifiants d'authentification en clair et de requêtes SQL dans Android Logcat

Plusieurs activités écrivent sans condition des noms d'utilisateur, des mots de passe et des instructions SQL en clair dans les logs système Android via `Log.e`, `Log.v` et `System.out.println`, permettant l'exfiltration d'identifiants par toute application disposant d'un accès aux logs ou par un attaquant utilisant `adb`.

L'analyse statique de `owasp.sat.agoat` a identifié trois activités dans lesquelles des entrées utilisateur sensibles transitent directement vers des sinks de journalisation sans être masquées. `InsecureLoggingActivity` envoie le nom d'utilisateur et le mot de passe bruts vers logcat et la sortie standard à chaque clic sur le bouton. `InsecureStorageSQLiteActivity` journalise l'instruction SQL INSERT complète contenant les identifiants en clair, tandis que `SQLInjectionActivity` divulgue à la fois la requête SELECT brute et les enregistrements de base de données résultants. Du fait que les logs Android sont lisibles par tous sur de nombreux

appareils et facilement accessibles via `adb logcat`, cela constitue un canal de divulgation d'informations à fort impact qui ne requiert aucune complexité d'exploitation.

Détails techniques

Résultat #1

Cause racine

Les développeurs ont utilisé `android.util.Log` et `System.out.println` pour déboguer les entrées utilisateur et les opérations de base de données. Aucun assainissement, gating conditionnel ni masquage n'est appliqué avant que les chaînes ne soient écrites dans le tampon de journal système.

Code vulnérable

`InsecureLoggingActivity.java` (lignes 59 et 70):

```
Log.e("Error:", sb.toString());
// ...
PrintStream printStream = System.out;
printStream.println(sb2.toString());
```

`InsecureStorageSQLiteActivity.java` (ligne 87):

```
Log.v("Query", qry);
```

`SQLInjectionActivity.java` (lignes 82–83 et 93–94):

```
Log.e("Error:", strb.toString());
Log.v("Query", qry);
```

Preuves d'exploitation

[Analyse statique] L'audit statique a tracé un flux de données inconditionnel depuis `EditText.getText()` via une concaténation `StringBuilder` directement vers les sinks de journalisation. Les extraits ci-dessus ont été extraits verbatim de l'arbre source décompilé à l'emplacement `sources/owasp/sat/agoat/`. L'exécution dynamique n'a pas été réalisée ; la capture logcat en temps réel n'a donc pas été obtenue ; cependant, le chemin de code est inconditionnel et émettra les données à chaque invocation.

Preuves de validation

Cause racine

L'analyse statique source-to-sink a confirmé que trois activités dans `owasp.sat.agoat` transmettent des identifiants utilisateur bruts et des chaînes de requête SQL directement dans les sinks `android.util.Log` et `System.out.println` sans aucun assainissement, gating conditionnel ni masquage. Un grep ciblé sur l'ensemble du source décompilé a vérifié l'absence de références à `BuildConfig.DEBUG`, de vérifications `Log.isLoggable()` et de toute fonction de masquage (`replace`, `mask`, `redact`, `substring`) dans les trois fichiers affectés. Le fichier `AndroidManifest.xml` déclare en outre `android:debuggable="true"`, ce qui garantit un accès trivial à `adb logcat` et amplifie le canal de divulgation.

Code vulnérable

`InsecureLoggingActivity.java` (lignes 45–70) — `onClick` dans `InsecureLoggingActivity$onCreate$1` :

```
// Lines 50-58: First StringBuilder with raw credentials
StringBuilder sb = new StringBuilder();
sb.append("Error occured when processing Username ");
sb.append((Object) username2.getText()); // line 54
sb.append(" and Password ");
sb.append((Object) password2.getText()); // line 58
Log.e("Error:", sb.toString());           // line 59

// Lines 61-70: Second StringBuilder written to stdout
StringBuilder sb2 = new StringBuilder();
sb2.append("Error: Error occured when processing Username ");
sb2.append((Object) username3.getText()); // line 65
sb2.append(" and Password ");
sb2.append((Object) password3.getText()); // line 69
PrintStream printStream = System.out;
printStream.println(sb2.toString());      // line 70
```

`InsecureStorageSQLiteActivity.java` (lignes 64–87) — requête SQL INSERT journalisée inconditionnellement en dehors du bloc `try/catch` :

```
// Lines 64-74: Raw SQL construction
StringBuilder sb = new StringBuilder();
sb.append("INSERT INTO users (username, password) VALUES('");
sb.append(username2.getText().toString()); // line 68
sb.append("'", "'");
sb.append(password2.getText().toString()); // line 72
sb.append("'");
String qry = sb.toString(); // line 74

// Line 80: execSQL inside try/catch
// Line 87: Log.v executed regardless of success or failure
Log.v("Query", qry); // line 87
```

`SQLInjectionActivity.java` (lignes 56–94) — SELECT brut et journalisation du jeu de résultats :

```
// Lines 56-62: Raw SELECT construction
StringBuilder sb = new StringBuilder();
sb.append("SELECT * FROM users WHERE username='");
sb.append(username2.getText().toString()); // line 60
sb.append("'");
String qry = sb.toString(); // line 62

// Lines 69-80: Cursor iteration appending plaintext records
Cursor QryResult = db.rawQuery(qry, null); // line 69
strb = new StringBuilder(""); // line 70
while (QryResult.moveToNext()) {
    strb.append("Username: (" + QryResult.getString(1) + ") password: (" + QryResult.getString(2) + ")\n"); // line 79
}

// Lines 82-83: Emitted when rows are returned
Log.e("Error:", strb.toString()); // line 82
Log.v("Query", qry); // line 83

// Lines 85-94: Unconditionally emitted on every button click
StringBuilder sb2 = new StringBuilder();
sb2.append("User: (");
sb2.append(username3.getText().toString()); // line 89
sb2.append(") not found");
strb.append(sb2.toString()); // line 91
Log.e("Error:", strb.toString()); // line 93
Log.v("Query", qry); // line 94
```

Preuves d'exploitation

[Analyse statique] La validation a effectué des traces formelles source-to-sink sur les trois activités, confirmant des flux de données inconditionnels depuis

`EditText.getText()` vers les sinks de journalisation :

1. **InsecureLoggingActivity** — Les lignes 54 et 58 récupèrent le nom d'utilisateur et le mot de passe bruts dans le `StringBuilder sb` ; la ligne 59 émet via `Log.e` . Les lignes 65 et 69 alimentent un second builder émis vers `System.out` à la ligne 70. Aucune garde `if (DEBUG)` ne protège l'un ou l'autre sink.
2. **InsecureStorageSQLiteActivity** — La ligne 87 (`Log.v("Query", qry)`) se situe **après** le bloc `try/catch` (lignes 75–86) ; elle s'exécute donc que `execSQL` réussisse ou lève une exception. La chaîne SQL INSERT complète contient le mot de passe en clair.
3. **SQLInjectionActivity** — Les lignes 82–83 journalisent le jeu de résultats complet (y compris les mots de passe de la base de données en clair) lorsque `getCount() > 0` . Les lignes 93–94 s'exécutent **inconditionnellement** à chaque clic sur le bouton, émettant la requête SELECT brute et un tampon contenant soit les enregistrements du jeu de résultats, soit l'entrée utilisateur suivie du suffixe « not found ».

L'audit du fichier `AndroidManifest.xml` a en outre confirmé `android:debuggable="true"` (ligne 5), ce qui trivialise l'exploitation sur appareil physique car un attaquant disposant d'un accès USB/adb peut exécuter `adb logcat` et capturer les identifiants et les instructions SQL émis sans avoir à rooter l'appareil ni à contourner les restrictions d'accès à logcat.

Limite des preuves : l'exécution dynamique et la capture logcat en temps réel n'ont pas été réalisées lors de cette validation ; l'exploitabilité a été confirmée entièrement par analyse statique du source décompilé.

Recommandation

Action immédiate

Supprimez ou commentez tous les appels à `Log.e` , `Log.d` , `Log.v` et `System.out.println` dans les fichiers identifiés qui traitent les identifiants utilisateur ou les requêtes SQL.

Correctif permanent

Mettez en place un utilitaire de journalisation centralisé qui inspecte les messages à la recherche de motifs sensibles et les remplace par `[REDACTED]` avant leur émission. Utilisez des règles ProGuard ou R8 pour supprimer les appels à `Log.*` des builds de production :

```
-assumenosideeffects class android.util.Log {
    public static int v(...);
    public static int d(...);
    public static int i(...);
    public static int w(...);
    public static int e(...);
}
```

Vérification

Compilez l'application en mode production, parcourez les flux de saisie des identifiants et exécutez `adb logcat | grep -i "Error\|Query"`. Vérifiez qu'aucun nom d'utilisateur, mot de passe ou texte de requête SQL n'apparaît dans les logs.

ÉLEVÉ Injection SQL via execSQL non assaini dans InsecureStorageSQLiteActivity

InsecureStorageSQLiteActivity construit une instruction INSERT en concaténant deux valeurs EditText non assainies dans execSQL(), ce qui permet une injection de requêtes empilées, une manipulation arbitraire des enregistrements et une compromission complète de l'intégrité de la base de données locale.

L'activité InsecureStorageSQLiteActivity persiste les informations d'identification utilisateur dans la base de données SQLite locale `aGoat` en construisant une instruction `INSERT` par concaténation directe de chaînes à partir de deux champs `EditText` contrôlés par l'attaquant. La chaîne résultante est passée à `SQLiteDatabase.execSQL()`, qui accepte plusieurs instructions SQL séparées par des points-virgules. Cela permet à un attaquant de terminer prématurément l'`INSERT` initial et d'exécuter des instructions secondaires arbitraires, notamment `INSERT`, `UPDATE`, `DELETE` et `CREATE TABLE`. Les champs nom d'utilisateur et mot de passe constituent tous deux des points d'injection vulnérables, et les écritures persistantes signifient que la compromission de l'intégrité est immédiate et durable. L'activité partage le même fichier de base de données que SQLInjectionActivity, ce qui permet aux données injectées d'influencer les requêtes de lecture.

Détails techniques

Résultat #1

Cause racine

Dans `owasp/sat/agoat/InsecureStorageSQLiteActivity.java` à la ligne 80, l'application construit une instruction `INSERT` en interpolant deux entrées UI non assainies directement dans la chaîne SQL et l'exécute via `execSQL()`. Contrairement à `rawQuery()`, `execSQL()` ne restreint pas l'exécution à une seule instruction, ce qui permet une injection par requêtes empilées.

Code vulnérable

Le motif de construction de requête vulnérable identifié lors de l'analyse statique est le suivant :

```
String qry = "INSERT INTO users (username, password) VALUES('" +
    userName.getText().toString() + "','" + password.getText().toString() + "')";
sqliteDatabase.execSQL(qry);
```

Les entrées `userName` et `password` proviennent de widgets `EditText` (`R.id.userName` et `R.id.password`) sans aucune validation ni paramétrisation.

Preuve d'exploitation

[Simulé] Une base de données SQLite locale simulée avec le schéma exact `aGoat` a été utilisée pour exécuter des payloads reconstruits. Les résultats suivants confirment l'exécution arbitraire d'instructions :

- Échappement de guillemet / terminaison anticipée via le champ username** : la saisie `test','test'); --` a provoqué l'exécution par la base de données de `INSERT INTO users (username, password) VALUES('test','test'); --)`, insérant effectivement `(test, test)` et mettant le reste en commentaire.
- Injection multi-instructions via le champ username** : la saisie `test','test'); INSERT INTO users (username, password) VALUES('attacker','attacker'); -` a entraîné l'exécution de **deux instructions distinctes**. La simulation a confirmé l'insertion de **deux lignes**, prouvant l'exécution arbitraire d'une instruction secondaire.
- Injection multi-instructions via le champ password** : la saisie `test'); INSERT INTO users (username, password) VALUES('attacker2','attacker2'); --` a également entraîné l'insertion de **deux lignes**, confirmant que les deux champs de saisie sont indépendamment vulnérables.
- Compromission de l'intégrité via UPDATE** : un payload empilé a ajouté `UPDATE users SET password='hacked' WHERE username='admin'; --` après l'`INSERT`. La simulation a confirmé que le mot de passe `admin` a été définitivement changé en `hacked`.
- Exfiltration de données via CREATE TABLE** : un payload empilé a créé une nouvelle table `stolen_data` et y a inséré l'ensemble des identifiants existants, confirmant une exfiltration complète des données via des objets de schéma contrôlés par l'attaquant.

Preuve de validation

Cause racine

[Analyse statique] Le source décompilé par JADX situé dans

`/workspace/source_code_analysis_b456ad45d2f249a09fc5ea8e695da9e6/base.apk/sources/owasp/sat/agoat/InsecureStorageSQLiteActivity.java` confirme le sink vulnérable à la **ligne 80**. La méthode `onCreate()` (lignes 44–90) récupère les références `EditText` de `R.id.userName` (ligne 58) et `R.id.password` (ligne 59), attache un écouteur de clic au bouton Enregistrer (`R.id.SQLButton`) (ligne 60), et dans `onClick()` (ligne 62) construit la requête par concaténation directe de chaînes avant de la passer à `this.mDB.execSQL(qry)`. Aucun placeholder `?`, aucun `bindArgs` ni aucune étape d'assainissement intermédiaire ne sont présents.

Code vulnérable

Le source Java décompilé (transpilé depuis Kotlin par JADX) montre le motif de concaténation exact :

```
// InsecureStorageSQLiteActivity.java:47-52
this.mDB = openOrCreateDatabase("aGoat", 0, null);
this.mDB.execSQL("CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR)");

// InsecureStorageSQLiteActivity.java:58-59
final EditText username = (EditText) findViewById(R.id.userName);
final EditText password = (EditText) findViewById(R.id.password);

// InsecureStorageSQLiteActivity.java:80
String qry = "INSERT INTO users (username, password) VALUES(' " +
    username.getText().toString() + "', '" + password.getText().toString() + "')";
this.mDB.execSQL(qry);
```

La validation a également confirmé que `Log.v("Query", qry)` (ligne 81) et `Log.d("Error", ...)` (ligne 84) écrivent la chaîne de requête complète — y compris tout payload injecté — dans les journaux système, fournissant un canal auxiliaire supplémentaire à un attaquant disposant de l'accès `READ_LOGS` ou d'un accès physique ADB.

Preuve d'exploitation

[Analyse statique] La validation a confirmé que les deux champs de saisie atteignent indépendamment le sink vulnérable sans validation, échappement ni logique de liste blanche. Le layout XML (`activity_insecure_storage_sqlite.xml`) lie directement `R.id.userName` et `R.id.password` aux widgets `EditText` sources, et le traçage arrière depuis la ligne 80 ne révèle aucun contrôle défensif.

L'impact inter-activités a été validé par l'analyse de `SQLInjectionActivity.java` (lignes 45–97), qui ouvre le fichier de base de données identique :

```
// SQLInjectionActivity.java:64
SQLInjectionActivity.this.mDB = SQLInjectionActivity.this.openOrCreateDatabase("aGoat", 0, null);

// SQLInjectionActivity.java:56-62
StringBuilder sb = new StringBuilder();
sb.append("SELECT * FROM users WHERE username='");
sb.append(username2.getText().toString());
sb.append("'");
String qry = sb.toString();

// SQLInjectionActivity.java:69
QryResult = SQLiteDatabase.rawQuery(qry, null);
```

Les deux activités référençant la même base de données `"aGoat"` et `execSQL()` effectuant un commit immédiat sur le disque, tout payload empilé compromettant l'intégrité (par ex. `UPDATE`, `DELETE`, `CREATE TABLE`) exécuté via `InsecureStorageSQLiteActivity` est immédiatement visible par les requêtes `SELECT` de `SQLInjectionActivity`.

L'audit du manifeste amplifie encore l'impact : `android:allowBackup="true"` et `android:debuggable="true"` permettent l'extraction et la manipulation hors ligne de la base de données `aGoat` via `adb backup` ou `run-as`, tandis que `clearTextTrafficPermitted="true"` et la confiance accordée aux autorités de certification utilisateur dans `network_security_config.xml` permettent une interception réseau triviale.

Recommandation

Atténuation immédiate

Désactivez le bouton `Save` ou ajoutez des restrictions de longueur des entrées côté client comme solution temporaire en attendant le déploiement du correctif définitif. Supprimez `android:debuggable="true"` du manifeste.

Correctif définitif

Remplacez l'appel à `execSQL()` par `SQLiteDatabase.insert()` avec `ContentValues`, ce qui élimine totalement la concaténation de chaînes :

```
ContentValues values = new ContentValues();
values.put("username", userName.getText().toString());
values.put("password", password.getText().toString());
sqliteDatabase.insert("users", null, values);
```

Si l'application doit utiliser `execSQL()` pour la gestion du schéma, assurez-vous qu' `execSQL()` n'est **jamais** appelé avec des chaînes contenant des entrées utilisateur. Toute manipulation de données doit utiliser des API paramétrées (`insert()`, `update()`, `delete()`) ou `SQLiteStatement` avec des paramètres liés.

Vérification

1. Relancez l'analyse statique pour confirmer qu'aucune occurrence d' `execSQL()` ne concatène d'entrées utilisateur.
2. Tentez des payloads multi-instructions telles que `'); DELETE FROM users; --` contre le build corrigé et vérifiez que seule l'insertion prévue d'une seule ligne se produit.
3. Examinez la table `users` après chaque interaction de test pour confirmer l'absence de modifications inattendues du schéma et de lignes supplémentaires.

ÉLEVÉ Base de données Firebase exposée publiquement

exposer publiquement une base de données Firebase, pouvant entraîner des accès non autorisés, l'altération de données, la divulgation d'informations, des attaques malveillantes et une perte de réputation.

Firebase est une plateforme de développement d'applications mobiles et web fournissant divers outils et services aux développeurs, dont une base de données en temps réel. Si une base de données Firebase est exposée publiquement, quiconque peut accéder aux données stockées et les manipuler sans authentification ni autorisation.

L'impact sur la sécurité d'une exposition publique d'une base de données Firebase peut être grave, car elle expose aux risques suivants :

- Accès non autorisé : toute personne peut accéder aux données sans authentification ni autorisation, ce qui signifie que des utilisateurs non autorisés peuvent consulter les informations sensibles stockées dans la base.
- Altération des données : les attaquants peuvent modifier, ajouter ou supprimer des données dans la base, ce qui peut entraîner une perte de données ou des données erronées ayant des conséquences sérieuses pour l'application ou l'entreprise.
- Divulcation d'informations : les attaquants peuvent accéder à des informations sensibles telles que des mots de passe, des données personnelles, des informations financières et autres données confidentielles pouvant être stockées dans la base.
- Attaques malveillantes : les attaquants peuvent mener diverses attaques malveillantes, telles que des attaques par injection, du cross-site scripting et d'autres types d'attaques visant à exploiter les vulnérabilités de l'application connectée à la base de données Firebase.
- Perte de réputation : si des informations sensibles sont exposées, cela peut entraîner une perte de confiance et de crédibilité auprès des clients, avec un impact significatif sur la réputation de l'entreprise.

Pour vérifier si une base de données Firebase est exposée publiquement, vous pouvez utiliser la commande curl suivante :

<> BASH

```
curl -X GET 'https://<project-id>.firebaseio.com/.json'
```

Remplacez `<project-id>` par l'identifiant du projet Firebase que vous souhaitez vérifier. Cette commande tente de récupérer le nœud racine de la base de données Firebase au format JSON.

Si la base de données Firebase est exposée publiquement, vous devriez pouvoir récupérer les données sans authentification ni autorisation.

Si la base de données est sécurisée et correctement configurée, vous recevrez un message d'erreur indiquant que vous n'êtes pas autorisé à accéder aux données.

Détails techniques

Résultat #1

Firebase `https://androgoat-42597.firebaseio.com` est **publiquement** accessible et peut être validé à l'aide de la commande suivante : `curl --request GET --url https://androgoat-42597.firebaseio.com/.json` . La clé a été trouvée dans `res/values/strings.xml` .

Recommandation

Pour rendre une base de données Firebase privée, suivez ces étapes techniques :

- Activer Firebase Authentication : ce service permet d'authentifier les utilisateurs et de contrôler l'accès à votre base de données. Vous pouvez l'utiliser avec différents fournisseurs d'identité, tels que l'e-mail et le mot de passe, Google, Facebook, etc.

- Configurer Firebase Security Rules : ce mécanisme permet de définir comment vos données doivent être accessibles et quels utilisateurs peuvent lire ou écrire dans la base de données. Vous pouvez définir des règles en fonction de l'authentification des utilisateurs et de chemins de données spécifiques.

Références

- [Firebase Security Rules](#)
- [10 Firebase Realtime Database Rule Templates](#)

ÉLEVÉ Vulnérabilité LOGJAM – Nombre premier commun

Utilisation de nombres premiers Diffie-Hellman largement partagés, permettant à des acteurs étatiques de mener des attaques par précalcul contre une grande partie du trafic Internet.

Cette vulnérabilité indique que le serveur utilise des nombres premiers Diffie-Hellman largement partagés, qui permettent des attaques efficaces par précalcul et autorisent des acteurs étatiques à déchiffrer passivement une grande partie du trafic internet.

LOGJAM-common_primes se produit lorsque des serveurs utilisent les mêmes paramètres DH standard (nombres premiers) partagés par des millions d'installations. Bien que l'utilisation de nombres premiers partagés ait été considérée comme sûre, l'algorithme Number Field Sieve permet aux attaquants de précalculer des tables cryptographiques coûteuses une seule fois par nombre premier, puis de compromettre rapidement toute connexion utilisant ce nombre premier.

Fonctionnement :

1. L'attaquant identifie des nombres premiers DH largement utilisés (par exemple, les paramètres par défaut d'Apache/OpenSSL)
2. Il effectue un précalcul coûteux (plusieurs semaines de calcul) pour le nombre premier ciblé
3. Il capture passivement les échanges de clés DH depuis tout serveur utilisant ce nombre premier
4. Il utilise les tables précalculées pour compromettre des connexions individuelles en quelques minutes
5. Il passe l'attaque à l'échelle sur des millions de serveurs partageant le même nombre premier

Prérequis :

- Le serveur utilise des nombres premiers DH communs ou par défaut (1024 bits ou moins)
- Des ressources de calcul considérables pour le précalcul initial (~100 M\$ pour un nombre premier de 1024 bits)
- Capacité à capturer le trafic réseau des connexions cibles
- Aucune capacité man-in-the-middle nécessaire — attaque purement passive

Exemple de scénario :

Un grand fournisseur cloud utilise des paramètres DH par défaut de 1024 bits sur des milliers de serveurs. Un acteur étatique passe des mois à précalculer des tables cryptographiques pour ce nombre premier spécifique. Une fois l'opération terminée, il peut surveiller passivement le trafic internet et déchiffrer en temps réel toute connexion TLS vers ces serveurs, affectant des millions d'utilisateurs sans aucune interférence active ni détection.

Casser le nombre premier de 1024 bits le plus répandu permettrait d'écouter passivement 18 % des principaux domaines HTTPS, tandis que casser les deux nombres premiers les plus courants compromettrait 66 % des serveurs VPN et 26 % des serveurs SSH à l'échelle mondiale.

Détails techniques

Résultat #1

LOGJAM-common_primes détecté sur `demo.testfire.net`

- **Constat :** RFC2409/Oakley Group 2

Recommandation

Pour atténuer les attaques LOGJAM par nombres premiers communs :

Défense principale – Générer des paramètres DH uniques :

```
# Générer des paramètres DH uniques de 2048 bits
openssl dhparam -out unique-dhparams.pem 2048

# Apache : utiliser des paramètres personnalisés
SSLOpenSSLConfCmd DHParameters /path/to/unique-dhparams.pem

# Nginx : spécifier des paramètres uniques
ssl_dhparam /path/to/unique-dhparams.pem;
```

Éviter les nombres premiers par défaut ou communs :

Ne jamais utiliser ces nombres premiers largement compromis :

- Nombre premier 512 bits par défaut d'Apache (utilisé par des millions de serveurs)
- Nombre premier 512 bits par défaut d'OpenSSL (présent dans SSLey depuis 1995)
- Nombres premiers standards de la RFC 5114 (potentiellement influencés par la NSA)

Migrer vers ECDHE :

Solution optimale – éliminer entièrement la vulnérabilité liée à DH :

```
# Apache : donner la priorité à ECDHE plutôt qu'à DHE
SSLCipherSuite ECDHE-ECDSA-AES256-GCM-SHA384:ECDSA-AES256-GCM-SHA384:!DHE

# Nginx : n'utiliser que des suites de chiffrement ECDHE
ssl_ciphers ECDHE-ECDSA-AES256-GCM-SHA384:ECDSA-AES256-GCM-SHA384:!DHE;
```

Utiliser des paramètres d'au moins 2048 bits :

Si DHE est requis :

```
# Garantir une taille de clé minimale
ssl_dhparam_size 2048; # Nginx
```

Détecter les nombres premiers communs :

```
# Extraire les paramètres DH du serveur
openssl s_client -connect example.com:443 -cipher DHE 2>&1 | \
  openssl dhparam -inform PEM -text -noout

# Comparer avec les nombres premiers vulnérables connus
# Vérifier si le nombre premier correspond à la valeur par défaut d'Apache ou à d'autres valeurs communes
```

Protections complémentaires :

- Utiliser TLS 1.3, qui élimine entièrement la DH sur corps fini
- Implémenter le certificate pinning pour détecter les tentatives MITM
- Surveiller les motifs de connexion inhabituels indiquant une analyse de trafic massive
- Prendre en compte les implications de la Perfect Forward Secrecy lors du choix des suites de chiffrement

Point critique : Même des paramètres DH robustes de 1024 bits deviennent vulnérables lorsqu'ils sont partagés entre de nombreux serveurs, car le coût de précalcul peut être amorti. L'unicité compte autant que la taille.

Références

- [Common Prime Analysis](#)
- [Weak DH Research](#)
- [Technical Paper](#)

ÉLEVÉ Vulnérabilité d'injection SQL mobile

Une vulnérabilité d'injection SQL mobile permet à un attaquant d'exécuter des requêtes SQL malveillantes sur la base de données d'une application mobile, pouvant entraîner un accès non autorisé, une fuite de données ou une manipulation de données.

L'injection SQL mobile est une vulnérabilité permettant à un attaquant d'injecter des instructions SQL malveillantes dans des applications mobiles, et potentiellement d'obtenir un accès non autorisé à des données sensibles ou de manipuler la base de données.

Exemples

Kotlin

```
kotlin
import java.sql.Connection
import java.sql.DriverManager
import java.sql.PreparedStatement
import java.sql.ResultSet

fun main() {
    val input = readLine() ?: ""
```

```

val connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydatabase", "username", "password")
val statement = connection.prepareStatement("SELECT * FROM users WHERE username = ?")
statement.setString(1, input)
val resultSet = statement.executeQuery()

while (resultSet.next()) {
    val username = resultSet.getString("username")
    val password = resultSet.getString("password")
    println("Username: $username, Password: $password")
}

resultSet.close()
statement.close()
connection.close()
}

```

Détails techniques

Résultat #1

Le paramètre à la position: **0** de la méthode **execSQL** dans la classe **SQLiteDatabase** peut être vulnérable à une injection SQL. Taint tracé depuis un composant non atteignable: **owasp.sat.agoat.InsecureStorageSQLiteActivity**. Le TaintElement provient de: Classe **InsecureStorageSQLiteActivity**
Méthode **onCreate** Position **0** Signature **(Landroid/os/Bundle;)V**. Trace d'appel:

```

at android.database.sqlite.SQLiteDatabase.execSQL
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick
at owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.<init>
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate
at android.support.v7.app.AppCompatActivity.onCreate
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate

```

à **android.database.sqlite.SQLiteDatabase.execSQL**

Unable to extract source code.

à **owasp.sat.agoat.InsecureStorageSQLiteActivity\$onCreate\$1.onClick**

```

public final void onClick(android.view.View p7)
{
    String v0_1 = new StringBuilder();
    v0_1.append(INSERT INTO users (username, password) VALUES(');
    String v1_8 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(',');
    String v1_2 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, password);
    v0_1.append(v1_2.getText().toString());
    v0_1.append('');
    String v0_2 = v0_1.toString();
    try {
        android.widget.Toast v2_2 = owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p(this.this$0);
    } catch (android.widget.Toast v2_3) {
        CharSequence v3_1 = new StringBuilder();
        v3_1.append(Error occurred when inserting data into database: );
        v3_1.append(v2_3.getMessage());
        android.util.Log.d(Error, v3_1.toString());
        CharSequence v3_4 = this.this$0.getApplicationContext();
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(Data not saved: Error occurred - );
        v4_4.append(v2_3.getMessage());
        android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v2_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v2_2.execSQL(v0_2);
}

```

```

        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
}

```

à `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick`

```

public final void onClick(android.view.View p7)
{
    String v0_1 = new StringBuilder();
    v0_1.append(INSERT INTO users (username, password) VALUES(');
    String v1_8 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(',');
    String v1_2 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_2, password);
    v0_1.append(v1_2.getText().toString());
    v0_1.append('');
    String v0_2 = v0_1.toString();
    try {
        android.widget.Toast v2_2 = owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p(this.this$0);
    } catch (android.widget.Toast v2_3) {
        CharSequence v3_1 = new StringBuilder();
        v3_1.append(Error occurred when inserting data into database: );
        v3_1.append(v2_3.getMessage());
        android.util.Log.d(Error, v3_1.toString());
        CharSequence v3_4 = this.this$0.getApplicationContext();
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(Data not saved: Error occured - );
        v4_4.append(v2_3.getMessage());
        android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v2_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v2_2.execSQL(v0_2);
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    android.util.Log.v(Query, v0_2);
    return;
}

```

à `owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p`

```

public static final synthetic android.database.sqlite.SQLiteDatabase access$getMDB$p(owasp.sat.agoat.InsecureStorageSQLiteActivity p1)
{
    return p1.mDB;
}

```

à `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick`

```

public final void onClick(android.view.View p7)
{
    String v0_1 = new StringBuilder();
    v0_1.append(INSERT INTO users (username, password) VALUES(');
    String v1_8 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(',');
    String v1_2 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_2, password);
    v0_1.append(v1_2.getText().toString());
    v0_1.append('');
    String v0_2 = v0_1.toString();
    try {
        android.widget.Toast v2_2 = owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p(this.this$0);
    } catch (android.widget.Toast v2_3) {
        CharSequence v3_1 = new StringBuilder();
        v3_1.append(Error occurred when inserting data into database: );
    }
}

```

```

        v3_1.append(v2_3.getMessage());
        android.util.Log.d(Error, v3_1.toString());
        CharSequence v3_4 = this.this$0.getApplicationContext();
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(Data not saved: Error occured - );
        v4_4.append(v2_3.getMessage());
        android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v2_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v2_2.execSQL(v0_2);
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    android.util.Log.v(Query, v0_2);
    return;
}

```

à `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate`

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    try {
        android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
        this.mDB = v0_6;
    } catch (android.widget.Button v0_0) {
        android.widget.EditText v1_1 = new StringBuilder();
        v1_1.append(Error occurred while creating database: );
        v1_1.append(v0_0.getMessage());
        android.util.Log.e(Error:, v1_1.toString());
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
    if (v0_6 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
    this.setContentView(2131296299);
    ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

à `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.<init>`

```

InsecureStorageSQLiteActivity$onCreate$1(owasp.sat.agoat.InsecureStorageSQLiteActivity p1, android.widget.EditText p2,
android.widget.EditText p3)
{
    this.this$0 = p1;
    this.$username = p2;
    this.$password = p3;
    return;
}

```

à `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate`

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    try {
        android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
        this.mDB = v0_6;
    } catch (android.widget.Button v0_0) {
        android.widget.EditText v1_1 = new StringBuilder();
        v1_1.append(Error occurred while creating database: );
    }
}

```

```

        v1_1.append(v0_0.getMessage());
        android.util.Log.e(Error:, v1_1.toString());
        this.setContentView(2131296299);
        ((android.widget.Button this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText this.findViewById(2131165369)),
((android.widget.EditText this.findViewById(2131165303))))));
        return;
    }
    if (v0_6 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
    this.setContentView(2131296299);
    ((android.widget.Button this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText this.findViewById(2131165369)),
((android.widget.EditText this.findViewById(2131165303))))));
    return;
}

```

à `android.support.v7.app.AppCompatActivity.onCreate`

```

protected void onCreate(android.os.Bundle p5)
{
    android.support.v7.app.AppCompatActivityDelegate v0 = this.getDelegate();
    v0.installViewFactory();
    v0.onCreate(p5);
    if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
        if (android.os.Build$VERSION.SDK_INT < 23) {
            this.setTheme(this.mThemeId);
        } else {
            this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
        }
    }
    super.onCreate(p5);
    return;
}

```

à `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate`

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    try {
        android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
        this.mDB = v0_6;
    } catch (android.widget.Button v0_0) {
        android.widget.EditText v1_1 = new StringBuilder();
        v1_1.append(Error occurred while creating database: );
        v1_1.append(v0_0.getMessage());
        android.util.Log.e(Error:, v1_1.toString());
        this.setContentView(2131296299);
        ((android.widget.Button this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText this.findViewById(2131165369)),
((android.widget.EditText this.findViewById(2131165303))))));
        return;
    }
    if (v0_6 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
    this.setContentView(2131296299);
    ((android.widget.Button this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText this.findViewById(2131165369)),
((android.widget.EditText this.findViewById(2131165303))))));
    return;
}

```

Résultat #2

Un paramètre contrôlé par l'utilisateur est utilisé pour construire une requête SQL vulnérable à l'injection SQL. Taint tracé depuis un composant non atteignable :

`owasp.sat.agoat.InsecureStorageSQLiteActivity` trouvé dans `owasp.sat.agoat.InsecureStorageSQLiteActivity` . Du paramètre `0` au sink `void` `android.database.sqlite.SQLiteDatabase.execSQL(java.lang.String)` . Trace de taint :

```

at android.database.sqlite.SQLiteDatabase.execSQL()
at owasp.sat.ag oat.InsecureStorageSQLiteActivity$onCreate$1.onClick()
at owasp.sat.ag oat.InsecureStorageSQLiteActivity$onCreate$1.onClick()
at owasp.sat.ag oat.InsecureStorageSQLiteActivity$onCreate$1.onClick()
at owasp.sat.ag oat.InsecureStorageSQLiteActivity.onCreate()
at owasp.sat.ag oat.InsecureStorageSQLiteActivity$onCreate$1.<init>()
at owasp.sat.ag oat.InsecureStorageSQLiteActivity.onCreate()
at owasp.sat.ag oat.InsecureStorageSQLiteActivity.onCreate()

```

Méthode `owasp.sat.ag oat.InsecureStorageSQLiteActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    String v0_1 = new StringBuilder();
    v0_1.append(INSERT INTO users (username, password) VALUES(');
    String v1_8 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(',');
    String v1_2 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_2, password);
    v0_1.append(v1_2.getText().toString());
    v0_1.append('');
    String v0_2 = v0_1.toString();
    try {
        android.widget.Toast v2_2 = owasp.sat.ag oat.InsecureStorageSQLiteActivity.access$getMDB$p(this.this$0);
    } catch (android.widget.Toast v2_3) {
        CharSequence v3_1 = new StringBuilder();
        v3_1.append(Error occurred when inserting data into database: );
        v3_1.append(v2_3.getMessage());
        android.util.Log.d(Error, v3_1.toString());
        CharSequence v3_4 = this.this$0.getApplicationContext();
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(Data not saved: Error occurred - );
        v4_4.append(v2_3.getMessage());
        android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v2_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v2_2.execSQL(v0_2);
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    android.util.Log.v(Query, v0_2);
    return;
}

```

Méthode `owasp.sat.ag oat.InsecureStorageSQLiteActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    String v0_1 = new StringBuilder();
    v0_1.append(INSERT INTO users (username, password) VALUES(');
    String v1_8 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(',');
    String v1_2 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_2, password);
    v0_1.append(v1_2.getText().toString());
    v0_1.append('');
    String v0_2 = v0_1.toString();
    try {
        android.widget.Toast v2_2 = owasp.sat.ag oat.InsecureStorageSQLiteActivity.access$getMDB$p(this.this$0);
    } catch (android.widget.Toast v2_3) {
        CharSequence v3_1 = new StringBuilder();
        v3_1.append(Error occurred when inserting data into database: );
        v3_1.append(v2_3.getMessage());
        android.util.Log.d(Error, v3_1.toString());
        CharSequence v3_4 = this.this$0.getApplicationContext();
        CharSequence v4_4 = new StringBuilder();
    }
}

```

```

        v4_4.append(Data not saved: Error occurred - );
        v4_4.append(v2_3.getMessage());
        android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v2_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v2_2.execSQL(v0_2);
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    android.util.Log.v(Query, v0_2);
    return;
}

```

Méthode `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    String v0_1 = new StringBuilder();
    v0_1.append(INSERT INTO users (username, password) VALUES(');
    String v1_8 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(',');
    String v1_2 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, password);
    v0_1.append(v1_2.getText().toString());
    v0_1.append('');
    String v0_2 = v0_1.toString();
    try {
        android.widget.Toast v2_2 = owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$sp(this.this$0);
    } catch (android.widget.Toast v2_3) {
        CharSequence v3_1 = new StringBuilder();
        v3_1.append(Error occurred when inserting data into database: );
        v3_1.append(v2_3.getMessage());
        android.util.Log.d(Error, v3_1.toString());
        CharSequence v3_4 = this.this$0.getApplicationContext();
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(Data not saved: Error occurred - );
        v4_4.append(v2_3.getMessage());
        android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v2_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v2_2.execSQL(v0_2);
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    android.util.Log.v(Query, v0_2);
    return;
}

```

Méthode `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    try {
        android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
        this.mDB = v0_6;
    } catch (android.widget.Button v0_0) {
        android.widget.EditText v1_1 = new StringBuilder();
        v1_1.append(Error occurred while creating database: );
        v1_1.append(v0_0.getMessage());
        android.util.Log.e(Error, v1_1.toString());
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
}

```

```

        if (v0_6 == null) {
            kotlin.jvm.internal.Intrinsics.throwNpe();
        }
        v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }

```

Méthode `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.<init>()` :

```

InsecureStorageSQLiteActivity$onCreate$1(owasp.sat.agoat.InsecureStorageSQLiteActivity p1, android.widget.EditText p2,
android.widget.EditText p3)
{
    this.this$0 = p1;
    this.$username = p2;
    this.$password = p3;
    return;
}

```

Méthode `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    try {
        android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
        this.mDB = v0_6;
    } catch (android.widget.Button v0_0) {
        android.widget.EditText v1_1 = new StringBuilder();
        v1_1.append(Error occurred while creating database: );
        v1_1.append(v0_0.getMessage());
        android.util.Log.e(Error:, v1_1.toString());
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
    if (v0_6 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
    this.setContentView(2131296299);
    ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

Méthode `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    try {
        android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
        this.mDB = v0_6;
    } catch (android.widget.Button v0_0) {
        android.widget.EditText v1_1 = new StringBuilder();
        v1_1.append(Error occurred while creating database: );
        v1_1.append(v0_0.getMessage());
        android.util.Log.e(Error:, v1_1.toString());
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),

```

```

((android.widget.EditText) this.findViewById(2131165303))));
    return;
}
if (v0_6 == null) {
    kotlin.jvm.internal.Intrinsics.throwNpe();
}
v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
this.setContentView(2131296299);
((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303))));
    return;
}

```

Résultat #3

Le paramètre à la position : **0** de la méthode **execSQL** de la classe **SQLiteDatabase** peut être vulnérable à une injection SQL. Taint tracé depuis le composant non atteignable : **owasp.sat.agoat.InsecureStorageSQLiteActivity**.

Le TaintElement provient de : classe **InsecureStorageSQLiteActivity**, méthode **onCreate**, position **0**, signature **(Landroid/os/Bundle;)V**.

Trace d'appel :

```

at android.database.sqlite.SQLiteDatabase.execSQL
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick
at android.database.sqlite.SQLiteDatabase.execSQL
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate
at owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.<init>
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate

```

à **android.database.sqlite.SQLiteDatabase.execSQL**

Unable to extract source code.

à **owasp.sat.agoat.InsecureStorageSQLiteActivity\$onCreate\$1.onClick**

```

public final void onClick(android.view.View p7)
{
    String v0_1 = new StringBuilder();
    v0_1.append(INSERT INTO users (username, password) VALUES(');
    String v1_8 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(', ');
    String v1_2 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, password);
    v0_1.append(v1_2.getText().toString());
    v0_1.append('');
    String v0_2 = v0_1.toString();
    try {
        android.widget.Toast v2_2 = owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p(this.this$0);
    } catch (android.widget.Toast v2_3) {
        CharSequence v3_1 = new StringBuilder();
        v3_1.append(Error occurred when inserting data into database: );
        v3_1.append(v2_3.getMessage());
        android.util.Log.d(Error, v3_1.toString());
        CharSequence v3_4 = this.this$0.getApplicationContext();
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(Data not saved: Error occurred - );
        v4_4.append(v2_3.getMessage());
        android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v2_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v2_2.execSQL(v0_2);
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
}

```

```

        android.util.Log.v(Query, v0_2);
        return;
    }

```

à `android.database.sqlite.SQLiteDatabase.execSQL`

Unable to extract source code.

à `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick`

```

public final void onClick(android.view.View p7)
{
    String v0_1 = new StringBuilder();
    v0_1.append(INSERT INTO users (username, password) VALUES(');
    String v1_8 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(',');
    String v1_2 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_2, password);
    v0_1.append(v1_2.getText().toString());
    v0_1.append('');
    String v0_2 = v0_1.toString();
    try {
        android.widget.Toast v2_2 = owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p(this.this$0);
    } catch (android.widget.Toast v2_3) {
        CharSequence v3_1 = new StringBuilder();
        v3_1.append(Error occurred when inserting data into database: );
        v3_1.append(v2_3.getMessage());
        android.util.Log.d(Error, v3_1.toString());
        CharSequence v3_4 = this.this$0.getApplicationContext();
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(Data not saved: Error occurred - );
        v4_4.append(v2_3.getMessage());
        android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v2_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v2_2.execSQL(v0_2);
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    android.util.Log.v(Query, v0_2);
    return;
}

```

à `owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick`

```

public final void onClick(android.view.View p7)
{
    String v0_1 = new StringBuilder();
    v0_1.append(INSERT INTO users (username, password) VALUES(');
    String v1_8 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(',');
    String v1_2 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_2, password);
    v0_1.append(v1_2.getText().toString());
    v0_1.append('');
    String v0_2 = v0_1.toString();
    try {
        android.widget.Toast v2_2 = owasp.sat.agoat.InsecureStorageSQLiteActivity.access$getMDB$p(this.this$0);
    } catch (android.widget.Toast v2_3) {
        CharSequence v3_1 = new StringBuilder();
        v3_1.append(Error occurred when inserting data into database: );
        v3_1.append(v2_3.getMessage());
        android.util.Log.d(Error, v3_1.toString());
        CharSequence v3_4 = this.this$0.getApplicationContext();
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(Data not saved: Error occurred - );
    }
}

```

```

        v4_4.append(v2_3.getMessage());
        android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v2_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v2_2.execSQL(v0_2);
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    android.util.Log.v(Query, v0_2);
    return;
}

```

à `owasp.sat.agoat.InsecureSQLiteActivity$onCreate$1.onClick`

```

public final void onClick(android.view.View p7)
{
    String v0_1 = new StringBuilder();
    v0_1.append(INSERT INTO users (username, password) VALUES(');
    String v1_8 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(',');
    String v1_2 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v1_2, password);
    v0_1.append(v1_2.getText().toString());
    v0_1.append('');
    String v0_2 = v0_1.toString();
    try {
        android.widget.Toast v2_2 = owasp.sat.agoat.InsecureSQLiteActivity.access$getMDB$p(this.this$0);
    } catch (android.widget.Toast v2_3) {
        CharSequence v3_1 = new StringBuilder();
        v3_1.append(Error occurred when inserting data into database: );
        v3_1.append(v2_3.getMessage());
        android.util.Log.d(Error, v3_1.toString());
        CharSequence v3_4 = this.this$0.getApplicationContext();
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(Data not saved: Error occurred - );
        v4_4.append(v2_3.getMessage());
        android.widget.Toast.makeText(v3_4, ((CharSequence) v4_4.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v2_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v2_2.execSQL(v0_2);
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    android.util.Log.v(Query, v0_2);
    return;
}

```

à `owasp.sat.agoat.InsecureSQLiteActivity.onCreate`

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    try {
        android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
        this.mDB = v0_6;
    } catch (android.widget.Button v0_0) {
        android.widget.EditText v1_1 = new StringBuilder();
        v1_1.append(Error occurred while creating database: );
        v1_1.append(v0_0.getMessage());
        android.util.Log.e(Error, v1_1.toString());
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
    if (v0_6 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
}

```

```

v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
this.setContentView(2131296299);
((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.ag oat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
return;
}

```

à `owasp.sat.ag oat.InsecureStorageSQLiteActivity$onCreate$1.<init>`

```

InsecureStorageSQLiteActivity$onCreate$1(owasp.sat.ag oat.InsecureStorageSQLiteActivity p1, android.widget.EditText p2,
android.widget.EditText p3)
{
    this.this$0 = p1;
    this.$username = p2;
    this.$password = p3;
    return;
}

```

à `owasp.sat.ag oat.InsecureStorageSQLiteActivity.onCreate`

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    try {
        android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
        this.mDB = v0_6;
    } catch (android.widget.Button v0_0) {
        android.widget.EditText v1_1 = new StringBuilder();
        v1_1.append(Error occurred while creating database: );
        v1_1.append(v0_0.getMessage());
        android.util.Log.e(Error:, v1_1.toString());
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.ag oat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
    if (v0_6 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
    this.setContentView(2131296299);
    ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.ag oat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

à `owasp.sat.ag oat.InsecureStorageSQLiteActivity.onCreate`

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    try {
        android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
        this.mDB = v0_6;
    } catch (android.widget.Button v0_0) {
        android.widget.EditText v1_1 = new StringBuilder();
        v1_1.append(Error occurred while creating database: );
        v1_1.append(v0_0.getMessage());
        android.util.Log.e(Error:, v1_1.toString());
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.ag oat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
    if (v0_6 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
}

```

```

v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
this.setContentView(2131296299);
((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
return;
}

```

Résultat #4

Un paramètre contrôlé par l'utilisateur est utilisé pour construire un paramètre SQL vulnérable à une injection SQL. Taint tracé depuis un composant non atteignable :

`owasp.sat.agoat.SQLInjectionActivity` trouvé dans `owasp.sat.agoat.SQLInjectionActivity`. Depuis le paramètre `0` jusqu'au sink `android.database.Cursor android.database.sqlite.SQLiteDatabase.rawQuery(java.lang.String, java.lang.String[])`. Trace de taint :

```

at android.database.sqlite.SQLiteDatabase.rawQuery()
at owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLInjectionActivity.access$getMDB$p()
at owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLInjectionActivity.onCreate()
at owasp.sat.agoat.SQLInjectionActivity$onCreate$1.<init>()
at owasp.sat.agoat.SQLInjectionActivity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.SQLInjectionActivity.onCreate()

```

Méthode `owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p9)
{
    String v0_1 = new StringBuilder();
    v0_1.append(SELECT * FROM users WHERE username=');
    String v1_6 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_6, username);
    v0_1.append(v1_6.getText().toString());
    v0_1.append(' ');
    String v0_2 = v0_1.toString();
    try {
        owasp.sat.agoat.SQLInjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
        android.database.Cursor v3_2 = owasp.sat.agoat.SQLInjectionActivity.access$getMDB$p(this.this$0);
    } catch (boolean v2_13) {
        android.database.Cursor v3_5 = new StringBuilder();
        v3_5.append(Error occurred when inserting data into database: );
        v3_5.append(v2_13.getMessage());
        android.util.Log.d(Error, v3_5.toString());
        android.database.Cursor v3_8 = this.this$0.getApplicationContext();
        StringBuilder v4_10 = new StringBuilder();
        v4_10.append(Data not saved: Error occurred - );
        v4_10.append(v2_13.getMessage());
        android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v3_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
    StringBuilder v4_5 = new StringBuilder();
    if (v3_3 == null) {
        String v5_13 = new StringBuilder();
        v5_13.append(User: ());
        android.widget.EditText v6_2 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v6_2, username);
        v5_13.append(v6_2.getText().toString());
        v5_13.append() not found);
        v4_5.append(v5_13.toString());
    } else {
        if (v3_3.getCount() > 0) {
            v3_3.moveToFirst();
            do {
                boolean v2_5 = new StringBuilder();
                v2_5.append(Username: ());
                v2_5.append(v3_3.getString(1));
                v2_5.append() password: ());
                v2_5.append(v3_3.getString(2));
                v2_5.append()
            } while (v3_3.moveToNext());
        }
    }
}

```

```

);
        v4_5.append(v2_5.toString());
        } while(v3_3.moveToNext());
    } else {
    }
}
android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
android.util.Log.e(Error:, v4_5.toString());
android.util.Log.v(Query, v0_2);
return;
}

```

Méthode `owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p9)
{
    String v0_1 = new StringBuilder();
    v0_1.append(SELECT * FROM users WHERE username=');
    String v1_6 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_6, username);
    v0_1.append(v1_6.getText().toString());
    v0_1.append(' ');
    String v0_2 = v0_1.toString();
    try {
        owasp.sat.agoat.SQLInjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
        android.database.Cursor v3_2 = owasp.sat.agoat.SQLInjectionActivity.access$getMDB$p(this.this$0);
    } catch (boolean v2_13) {
        android.database.Cursor v3_5 = new StringBuilder();
        v3_5.append(Error occurred when inserting data into database: );
        v3_5.append(v2_13.getMessage());
        android.util.Log.d(Error, v3_5.toString());
        android.database.Cursor v3_8 = this.this$0.getApplicationContext();
        StringBuilder v4_10 = new StringBuilder();
        v4_10.append(Data not saved: Error occurred - );
        v4_10.append(v2_13.getMessage());
        android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v3_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
    StringBuilder v4_5 = new StringBuilder();
    if (v3_3 == null) {
        String v5_13 = new StringBuilder();
        v5_13.append(User: ());
        android.widget.EditText v6_2 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v6_2, username);
        v5_13.append(v6_2.getText().toString());
        v5_13.append() not found);
        v4_5.append(v5_13.toString());
    } else {
        if (v3_3.getCount() > 0) {
            v3_3.moveToFirst();
            do {
                boolean v2_5 = new StringBuilder();
                v2_5.append(Username: ());
                v2_5.append(v3_3.getString(1));
                v2_5.append() password: ());
                v2_5.append(v3_3.getString(2));
                v2_5.append()
            } while(v3_3.moveToNext());
        } else {
        }
    }
    android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
    android.util.Log.e(Error:, v4_5.toString());
    android.util.Log.v(Query, v0_2);
    return;
}

```

Méthode `owasp.sat.agoat.SQLInjectionActivity.access$getMDB$p()` :

```
public static final synthetic android.database.sqlite.SQLiteDatabase access$getMDB$p(owasp.sat.agoat.SQLInjectionActivity p1)
{
    return p1.mDB;
}
```

Méthode `owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p9)
{
    String v0_1 = new StringBuilder();
    v0_1.append(SELECT * FROM users WHERE username=');
    String v1_6 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v1_6, username);
    v0_1.append(v1_6.getText().toString());
    v0_1.append('');
    String v0_2 = v0_1.toString();
    try {
        owasp.sat.agoat.SQLInjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
        android.database.Cursor v3_2 = owasp.sat.agoat.SQLInjectionActivity.access$getMDB$p(this.this$0);
    } catch (boolean v2_13) {
        android.database.Cursor v3_5 = new StringBuilder();
        v3_5.append(Error occurred when inserting data into database: );
        v3_5.append(v2_13.getMessage());
        android.util.Log.d(Error, v3_5.toString());
        android.database.Cursor v3_8 = this.this$0.getApplicationContext();
        StringBuilder v4_10 = new StringBuilder();
        v4_10.append(Data not saved: Error occured - );
        v4_10.append(v2_13.getMessage());
        android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v3_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
    StringBuilder v4_5 = new StringBuilder();
    if (v3_3 == null) {
        String v5_13 = new StringBuilder();
        v5_13.append(User: ());
        android.widget.EditText v6_2 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v6_2, username);
        v5_13.append(v6_2.getText().toString());
        v5_13.append() not found);
        v4_5.append(v5_13.toString());
    } else {
        if (v3_3.getCount() > 0) {
            v3_3.moveToFirst();
            do {
                boolean v2_5 = new StringBuilder();
                v2_5.append(Username: ());
                v2_5.append(v3_3.getString(1));
                v2_5.append() password: ());
                v2_5.append(v3_3.getString(2));
                v2_5.append()
            );
                v4_5.append(v2_5.toString());
            } while(v3_3.moveToNext());
        } else {
        }
    }
    android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
    android.util.Log.e(Error;, v4_5.toString());
    android.util.Log.v(Query, v0_2);
    return;
}
```

Méthode `owasp.sat.agoat.SQLInjectionActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p4)
{
    super.onCreate(p4);
    this setContentView(2131296306);
    ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLInjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
    return;
}
```

Méthode `owasp.sat.agoat.SQLInjectionActivity$onCreate$1.<init>()` :

```
SQLInjectionActivity$onCreate$1(owasp.sat.agoat.SQLInjectionActivity p1, android.widget.EditText p2)
{
    this.this$0 = p1;
    this.$username = p2;
    return;
}
```

Méthode `owasp.sat.agoat.SQLInjectionActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p4)
{
    super.onCreate(p4);
    this setContentView(2131296306);
    ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLInjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
    return;
}
```

Méthode `android.support.v7.app.AppCompatActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p5)
{
    android.support.v7.app.AppCompatActivityDelegate v0 = this.getDelegate();
    v0.installViewFactory();
    v0.onCreate(p5);
    if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
        if (android.os.Build$VERSION.SDK_INT < 23) {
            this.setTheme(this.mThemeId);
        } else {
            this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
        }
    }
    super.onCreate(p5);
    return;
}
```

Méthode `owasp.sat.agoat.SQLInjectionActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p4)
{
    super.onCreate(p4);
    this setContentView(2131296306);
    ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLInjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
    return;
}
```

Résultat #5

Un paramètre contrôlé par l'utilisateur est utilisé pour construire une requête SQL vulnérable à l'injection SQL Taint tracé depuis un composant non atteignable :

`owasp.sat.agoat.SQLInjectionActivity` identifié dans `owasp.sat.agoat.SQLInjectionActivity` . Depuis le paramètre `0` jusqu'au sink `android.database.Cursor android.database.sqlite.SQLiteDatabase.rawQuery(java.lang.String, java.lang.String[])` . Trace du taint:

```
at android.database.sqlite.SQLiteDatabase.rawQuery()
at owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLInjectionActivity.onCreate()
at owasp.sat.agoat.SQLInjectionActivity$onCreate$1.<init>()
at owasp.sat.agoat.SQLInjectionActivity.onCreate()
at owasp.sat.agoat.SQLInjectionActivity.onCreate()
```

Méthode `owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p9)
{
    String v0_1 = new StringBuilder();
    v0_1.append(SELECT * FROM users WHERE username=');
    String v1_6 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_6, username);
    v0_1.append(v1_6.getText().toString());
    v0_1.append('');
    String v0_2 = v0_1.toString();
    try {
        owasp.sat.agoat.SQLInjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
        android.database.Cursor v3_2 = owasp.sat.agoat.SQLInjectionActivity.access$getMDB$p(this.this$0);
    } catch (boolean v2_13) {
        android.database.Cursor v3_5 = new StringBuilder();
        v3_5.append(Error occurred when inserting data into database: );
        v3_5.append(v2_13.getMessage());
        android.util.Log.d(Error, v3_5.toString());
        android.database.Cursor v3_8 = this.this$0.getApplicationContext();
        StringBuilder v4_10 = new StringBuilder();
        v4_10.append(Data not saved: Error occured - );
        v4_10.append(v2_13.getMessage());
        android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v3_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
    StringBuilder v4_5 = new StringBuilder();
    if (v3_3 == null) {
        String v5_13 = new StringBuilder();
        v5_13.append(User: ());
        android.widget.EditText v6_2 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v6_2, username);
        v5_13.append(v6_2.getText().toString());
        v5_13.append() not found);
        v4_5.append(v5_13.toString());
    } else {
        if (v3_3.getCount() > 0) {
            v3_3.moveToFirst();
            do {
                boolean v2_5 = new StringBuilder();
                v2_5.append(Username: ());
                v2_5.append(v3_3.getString(1));
                v2_5.append() password: ());
                v2_5.append(v3_3.getString(2));
                v2_5.append()
            };
            v4_5.append(v2_5.toString());
        } while(v3_3.moveToNext());
    } else {
    }
}
    android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
    android.util.Log.e(Error:, v4_5.toString());
    android.util.Log.v(Query, v0_2);
    return;
}
```

Méthode `owasp.sat.agoad.SQLInjectionActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p9)
{
    String v0_1 = new StringBuilder();
    v0_1.append(SELECT * FROM users WHERE username=');
    String v1_6 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_6, username);
    v0_1.append(v1_6.getText().toString());
    v0_1.append('');
    String v0_2 = v0_1.toString();
    try {
        owasp.sat.agoad.SQLInjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
        android.database.Cursor v3_2 = owasp.sat.agoad.SQLInjectionActivity.access$getMDB$p(this.this$0);
    } catch (boolean v2_13) {
        android.database.Cursor v3_5 = new StringBuilder();
        v3_5.append(Error occurred when inserting data into database: );
        v3_5.append(v2_13.getMessage());
        android.util.Log.d(Error, v3_5.toString());
        android.database.Cursor v3_8 = this.this$0.getApplicationContext();
        StringBuilder v4_10 = new StringBuilder();
        v4_10.append(Data not saved: Error occurred - );
        v4_10.append(v2_13.getMessage());
        android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v3_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
    StringBuilder v4_5 = new StringBuilder();
    if (v3_3 == null) {
        String v5_13 = new StringBuilder();
        v5_13.append(User: ());
        android.widget.EditText v6_2 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v6_2, username);
        v5_13.append(v6_2.getText().toString());
        v5_13.append() not found;
        v4_5.append(v5_13.toString());
    } else {
        if (v3_3.getCount() > 0) {
            v3_3.moveToFirst();
            do {
                boolean v2_5 = new StringBuilder();
                v2_5.append(Username: ());
                v2_5.append(v3_3.getString(1));
                v2_5.append() password: ());
                v2_5.append(v3_3.getString(2));
                v2_5.append()
            );
            v4_5.append(v2_5.toString());
        } while(v3_3.moveToNext());
    } else {
    }
}
    android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
    android.util.Log.e(Error:, v4_5.toString());
    android.util.Log.v(Query, v0_2);
    return;
}
```

Méthode `owasp.sat.agoad.SQLInjectionActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p9)
{
    String v0_1 = new StringBuilder();
    v0_1.append(SELECT * FROM users WHERE username=');
    String v1_6 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_6, username);
    v0_1.append(v1_6.getText().toString());
    v0_1.append('');
    String v0_2 = v0_1.toString();
    try {
```

```

        owasp.sat.agoat.SQLInjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
        android.database.Cursor v3_2 = owasp.sat.agoat.SQLInjectionActivity.access$getMDB$p(this.this$0);
    } catch (boolean v2_13) {
        android.database.Cursor v3_5 = new StringBuilder();
        v3_5.append(Error occurred when inserting data into database: );
        v3_5.append(v2_13.getMessage());
        android.util.Log.d(Error, v3_5.toString());
        android.database.Cursor v3_8 = this.this$0.getApplicationContext();
        StringBuilder v4_10 = new StringBuilder();
        v4_10.append(Data not saved: Error occured - );
        v4_10.append(v2_13.getMessage());
        android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v3_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
    StringBuilder v4_5 = new StringBuilder();
    if (v3_3 == null) {
        String v5_13 = new StringBuilder();
        v5_13.append(User: );
        android.widget.EditText v6_2 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v6_2, username);
        v5_13.append(v6_2.getText().toString());
        v5_13.append() not found);
        v4_5.append(v5_13.toString());
    } else {
        if (v3_3.getCount() > 0) {
            v3_3.moveToFirst();
            do {
                boolean v2_5 = new StringBuilder();
                v2_5.append(Username: );
                v2_5.append(v3_3.getString(1));
                v2_5.append() password: );
                v2_5.append(v3_3.getString(2));
                v2_5.append()
            } while(v3_3.moveToNext());
            v4_5.append(v2_5.toString());
        } else {
        }
    }
    android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
    android.util.Log.e(Error:, v4_5.toString());
    android.util.Log.v(Query, v0_2);
    return;
}

```

Méthode `owasp.sat.agoat.SQLInjectionActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p4)
{
    super.onCreate(p4);
    this.setContentView(2131296306);
    ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLInjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
    return;
}

```

Méthode `owasp.sat.agoat.SQLInjectionActivity$onCreate$1.<init>()` :

```

SQLInjectionActivity$onCreate$1(owasp.sat.agoat.SQLInjectionActivity p1, android.widget.EditText p2)
{
    this.this$0 = p1;
    this.$username = p2;
    return;
}

```

Méthode `owasp.sat.agoat.SQLInjectionActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p4)
{
    super.onCreate(p4);
    this setContentView(2131296306);
    ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLInjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
    return;
}
```

Méthode `owasp.sat.agoat.SQLInjectionActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p4)
{
    super.onCreate(p4);
    this setContentView(2131296306);
    ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLInjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
    return;
}
```

Résultat #6

Un paramètre contrôlé par l'utilisateur est utilisé pour construire un paramètre SQL vulnérable à une injection SQL Taint tracé depuis un composant non atteignable :

`owasp.sat.agoat.SQLInjectionActivity` trouvé dans `owasp.sat.agoat.SQLInjectionActivity` . Du paramètre `0` au sink `android.database.Cursor`
`android.database.sqlite.SQLiteDatabase.rawQuery(java.lang.String, java.lang.String[])` . Trace de taint :

```
at android.database.sqlite.SQLiteDatabase.rawQuery()
at owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()
at android.database.sqlite.SQLiteDatabase.rawQuery()
at owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLInjectionActivity.onCreate()
at owasp.sat.agoat.SQLInjectionActivity$onCreate$1.<init>()
at owasp.sat.agoat.SQLInjectionActivity.onCreate()
at owasp.sat.agoat.SQLInjectionActivity.onCreate()
```

Méthode `owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p9)
{
    String v0_1 = new StringBuilder();
    v0_1.append(SELECT * FROM users WHERE username=');
    String v1_6 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_6, username);
    v0_1.append(v1_6.getText().toString());
    v0_1.append(' ');
    String v0_2 = v0_1.toString();
    try {
        owasp.sat.agoat.SQLInjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
        android.database.Cursor v3_2 = owasp.sat.agoat.SQLInjectionActivity.access$getMDB$p(this.this$0);
    } catch (boolean v2_13) {
        android.database.Cursor v3_5 = new StringBuilder();
        v3_5.append(Error occurred when inserting data into database: );
        v3_5.append(v2_13.getMessage());
        android.util.Log.d(Error, v3_5.toString());
        android.database.Cursor v3_8 = this.this$0.getApplicationContext();
        StringBuilder v4_10 = new StringBuilder();
        v4_10.append(Data not saved: Error occurred - );
        v4_10.append(v2_13.getMessage());
        android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v3_2 == null) {
```

```

        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
    StringBuilder v4_5 = new StringBuilder();
    if (v3_3 == null) {
        String v5_13 = new StringBuilder();
        v5_13.append(User: ());
        android.widget.EditText v6_2 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v6_2, username);
        v5_13.append(v6_2.getText().toString());
        v5_13.append() not found;
        v4_5.append(v5_13.toString());
    } else {
        if (v3_3.getCount() > 0) {
            v3_3.moveToFirst();
            do {
                boolean v2_5 = new StringBuilder();
                v2_5.append(Username: ());
                v2_5.append(v3_3.getString(1));
                v2_5.append() password: ());
                v2_5.append(v3_3.getString(2));
                v2_5.append()

            };
            v4_5.append(v2_5.toString());
        } while(v3_3.moveToNext());
    } else {
    }
}
android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
android.util.Log.e(Error:, v4_5.toString());
android.util.Log.v(Query, v0_2);
return;
}

```

Méthode `owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p9)
{
    String v0_1 = new StringBuilder();
    v0_1.append(SELECT * FROM users WHERE username=');
    String v1_6 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_6, username);
    v0_1.append(v1_6.getText().toString());
    v0_1.append('');
    String v0_2 = v0_1.toString();
    try {
        owasp.sat.agoat.SQLInjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
        android.database.Cursor v3_2 = owasp.sat.agoat.SQLInjectionActivity.access$getMDB$p(this.this$0);
    } catch (boolean v2_13) {
        android.database.Cursor v3_5 = new StringBuilder();
        v3_5.append(Error occurred when inserting data into database: );
        v3_5.append(v2_13.getMessage());
        android.util.Log.d(Error, v3_5.toString());
        android.database.Cursor v3_8 = this.this$0.getApplicationContext();
        StringBuilder v4_10 = new StringBuilder();
        v4_10.append(Data not saved: Error occured - );
        v4_10.append(v2_13.getMessage());
        android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v3_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
    StringBuilder v4_5 = new StringBuilder();
    if (v3_3 == null) {
        String v5_13 = new StringBuilder();
        v5_13.append(User: ());
        android.widget.EditText v6_2 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v6_2, username);
        v5_13.append(v6_2.getText().toString());
        v5_13.append() not found;
        v4_5.append(v5_13.toString());
    } else {
        if (v3_3.getCount() > 0) {
            v3_3.moveToFirst();

```

```

        do {
            boolean v2_5 = new StringBuilder();
            v2_5.append(Username: ());
            v2_5.append(v3_3.getString(1));
            v2_5.append() password: ());
            v2_5.append(v3_3.getString(2));
            v2_5.append()

        };

        v4_5.append(v2_5.toString());
    } while(v3_3.moveToNext());
} else {
}
}
android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
android.util.Log.e(Error:, v4_5.toString());
android.util.Log.v(Query, v0_2);
return;
}

```

Méthode `owasp.sat.ag oat.SQLInjectionActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p9)
{
    String v0_1 = new StringBuilder();
    v0_1.append(SELECT * FROM users WHERE username=');
    String v1_6 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_6, username);
    v0_1.append(v1_6.getText().toString());
    v0_1.append('');
    String v0_2 = v0_1.toString();
    try {
        owasp.sat.ag oat.SQLInjectionActivity.access$setMDB$0(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
        android.database.Cursor v3_2 = owasp.sat.ag oat.SQLInjectionActivity.access$getMDB$0(this.this$0);
    } catch (boolean v2_13) {
        android.database.Cursor v3_5 = new StringBuilder();
        v3_5.append(Error occurred when inserting data into database: );
        v3_5.append(v2_13.getMessage());
        android.util.Log.d(Error, v3_5.toString());
        android.database.Cursor v3_8 = this.this$0.getApplicationContext();
        StringBuilder v4_10 = new StringBuilder();
        v4_10.append(Data not saved: Error occured - );
        v4_10.append(v2_13.getMessage());
        android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v3_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
    StringBuilder v4_5 = new StringBuilder();
    if (v3_3 == null) {
        String v5_13 = new StringBuilder();
        v5_13.append(User: ());
        android.widget.EditText v6_2 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v6_2, username);
        v5_13.append(v6_2.getText().toString());
        v5_13.append() not found);
        v4_5.append(v5_13.toString());
    } else {
        if (v3_3.getCount() > 0) {
            v3_3.moveToFirst();
            do {
                boolean v2_5 = new StringBuilder();
                v2_5.append(Username: ());
                v2_5.append(v3_3.getString(1));
                v2_5.append() password: ());
                v2_5.append(v3_3.getString(2));
                v2_5.append()

            };

            v4_5.append(v2_5.toString());
        } while(v3_3.moveToNext());
    } else {
    }
}
android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
android.util.Log.e(Error:, v4_5.toString());

```

```

        android.util.Log.v(Query, v0_2);
        return;
    }

```

Méthode `owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p9)
{
    String v0_1 = new StringBuilder();
    v0_1.append(SELECT * FROM users WHERE username=');
    String v1_6 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_6, username);
    v0_1.append(v1_6.getText().toString());
    v0_1.append('');
    String v0_2 = v0_1.toString();
    try {
        owasp.sat.agoat.SQLInjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
        android.database.Cursor v3_2 = owasp.sat.agoat.SQLInjectionActivity.access$getMDB$p(this.this$0);
    } catch (boolean v2_13) {
        android.database.Cursor v3_5 = new StringBuilder();
        v3_5.append(Error occurred when inserting data into database: );
        v3_5.append(v2_13.getMessage());
        android.util.Log.d(Error, v3_5.toString());
        android.database.Cursor v3_8 = this.this$0.getApplicationContext();
        StringBuilder v4_10 = new StringBuilder();
        v4_10.append(Data not saved: Error occured - );
        v4_10.append(v2_13.getMessage());
        android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v3_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
    StringBuilder v4_5 = new StringBuilder();
    if (v3_3 == null) {
        String v5_13 = new StringBuilder();
        v5_13.append(User: );
        android.widget.EditText v6_2 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v6_2, username);
        v5_13.append(v6_2.getText().toString());
        v5_13.append() not found);
        v4_5.append(v5_13.toString());
    } else {
        if (v3_3.getCount() > 0) {
            v3_3.moveToFirst();
            do {
                boolean v2_5 = new StringBuilder();
                v2_5.append(Username: );
                v2_5.append(v3_3.getString(1));
                v2_5.append() password: );
                v2_5.append(v3_3.getString(2));
                v2_5.append()
            );
            v4_5.append(v2_5.toString());
        } while(v3_3.moveToNext());
    } else {
    }
}
    android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
    android.util.Log.e(Error:, v4_5.toString());
    android.util.Log.v(Query, v0_2);
    return;
}

```

Méthode `owasp.sat.agoat.SQLInjectionActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p4)
{
    super.onCreate(p4);
    this.setContentView(2131296306);
    ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new

```

```
owasp.sat.agoat.SQLInjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369))));
    return;
}
```

Méthode `owasp.sat.agoat.SQLInjectionActivity$onCreate$1.<init>()` :

```
SQLInjectionActivity$onCreate$1(owasp.sat.agoat.SQLInjectionActivity p1, android.widget.EditText p2)
{
    this.this$0 = p1;
    this.$username = p2;
    return;
}
```

Méthode `owasp.sat.agoat.SQLInjectionActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p4)
{
    super.onCreate(p4);
    this.setContentView(2131296306);
    ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLInjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369))));
    return;
}
```

Méthode `owasp.sat.agoat.SQLInjectionActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p4)
{
    super.onCreate(p4);
    this.setContentView(2131296306);
    ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLInjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369))));
    return;
}
```

Recommandation

Pour atténuer les vulnérabilités d'injection SQL mobile, appliquez les mesures suivantes :

- Utilisez des requêtes paramétrées ou des requêtes préparées pour séparer le code SQL des entrées utilisateur.
- Assainissez et validez les entrées utilisateur avant de les insérer dans la base de données afin d'atténuer les injections SQL de second ordre.
- Mettez régulièrement à jour les drivers SQL pour corriger les vulnérabilités connues.

KOTLIN

```
import java.sql.Connection
import java.sql.DriverManager
import java.sql.PreparedStatement

fun main() {
    val url = "jdbc:mysql://localhost:3306/mydatabase"
    val username = "username"
    val password = "password"

    var connection: Connection? = null
    var preparedStatement: PreparedStatement? = null

    try {
        connection = DriverManager.getConnection(url, username, password)
        val sql = "INSERT INTO users (name, email) VALUES (?, ?)"
        preparedStatement = connection.prepareStatement(sql)

        // Set values for the parameters
    }
```

```

        preparedStatement.setString(1, "John")
        preparedStatement.setString(2, "john@example.com")

        // Execute the prepared statement
        preparedStatement.executeUpdate()
    } catch (e: Exception) {
        e.printStackTrace()
    } finally {
        preparedStatement?.close()
        connection?.close()
    }
}

```

SWIFT

```

import Foundation
import SQLite3

func insertUser(name: String, email: String) {
    var db: OpaquePointer?
    var statement: OpaquePointer?

    let dbPath = "path_to_your_database_file.db"

    if sqlite3_open(dbPath, &db) == SQLITE_OK {
        let insertStatementString = "INSERT INTO users (name, email) VALUES (?, ?)"

        if sqlite3_prepare_v2(db, insertStatementString, -1, &statement, nil) == SQLITE_OK {
            sqlite3_bind_text(statement, 1, (name as NSString).utf8String, -1, nil)
            sqlite3_bind_text(statement, 2, (email as NSString).utf8String, -1, nil)

            if sqlite3_step(statement) == SQLITE_DONE {
                print("Successfully inserted row.")
            } else {
                print("Could not insert row.")
            }
        } else {
            print("INSERT statement could not be prepared.")
        }

        sqlite3_finalize(statement)
    } else {
        print("Unable to open database.")
    }

    sqlite3_close(db)
}

```

FLUTTER

```

import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';

void insertUser(String name, String email) async {
    Database database = await openDatabase(
        join(await getDatabasesPath(), 'mydatabase.db'),
        onCreate: (db, version) {
            return db.execute(
                "CREATE TABLE users(id INTEGER PRIMARY KEY, name TEXT, email TEXT)",
            );
        },
        version: 1,
    );

    await database.transaction((txn) async {
        await txn.rawInsert(
            'INSERT INTO users(name, email) VALUES(?, ?)',
            [name, email],
        );
    });
}

```

Références

MOYEN

Privilèges administrateurs déterminés par un flag booléen côté client sans vérification côté serveur

BinaryPatchingActivity contrôle l'accès aux fonctionnalités de l'interface d'administration via un champ booléen local qui n'est jamais validé par une API backend, un jeton cryptographique ni un challenge de réauthentification.

La classe Activity initialise un champ `private final boolean isAdmin` à `false` et l'utilise dans `onCreate()` pour déterminer s'il faut activer un bouton d'administration et afficher un texte administrateur. En l'absence de vérification des rôles côté serveur, de vérification de signature et d'authentification renforcée, un attaquant peut escalader ses privilèges en basculant ce flag à l'exécution via Frida ou en modifiant le bytecode Dalvik de l'APK. Cela illustre un vecteur d'escalade de privilèges reposant entièrement sur la confiance côté client, sans aucune vérification secondaire requise.

Détails techniques

Résultat #1

Cause racine

`BinaryPatchingActivity` contrôle l'accès aux éléments d'interface administrateur au moyen d'un champ `private final boolean isAdmin` initialisé à `false`. L'Activity vérifie cet indicateur lors de l'appel à `onCreate()`; si la valeur est `true`, elle active un bouton administrateur et met à jour une TextView. Il n'existe aucun appel d'API backend, attestation cryptographique ou authentification renforcée pour valider l'état administrateur.

Code vulnérable

`BinaryPatchingActivity.java` (lignes 17, 39–54):

```
private final boolean isAdmin; // Initialized to false

// ...
public final boolean isAdmin() {
    return this.isAdmin;
}

// In onCreate():
if (this.isAdmin) {
    ((TextView) findViewById(R.id.adminText)).setText("You are an admin");
    ((Button) findViewById(R.id.adminButton)).setEnabled(true);
}
```

Preuve d'exploitation

[Analyse statique] Le champ est une constante à la compilation sans vérification distante. Un attaquant peut utiliser Frida pour placer un hook sur `BinaryPatchingActivity.isAdmin()` et retourner `true`, ou appliquer un patch binaire sur l'APK afin de modifier l'initialiseur Dalvik de `const/4 v0, 0x0` à `0x1`. Le backend n'est jamais consulté et aucune réauthentification n'est déclenchée lorsque l'interface administrateur est activée.

Éléments de validation

Cause racine

BinaryPatchingActivity (package `owasp.sat.agoat`) contrôle l'accès aux éléments d'interface administrateur par l'intermédiaire d'un champ `private final boolean isAdmin` déclaré à la ligne 17 du source Java décompilé (`base.apk/sources/owasp/sat/agoat/BinaryPatchingActivity.java`). Ce champ reçoit la valeur par défaut de la JVM `false` et est immuable après construction en raison du modificateur `final`. Lors de l'appel à `onCreate()` (ligne 48), l'Activity vérifie `this.isAdmin`; lorsque la valeur est `true`, elle active `adminButton` et met à jour `isAdminText`. La validation a confirmé l'absence de tout vecteur d'entrée externe, de toute requête réseau, d'attestation cryptographique, de vérification de signature et d'authentification renforcée dans l'Activity ou son graphe d'appels atteignable.

Code vulnérable

`BinaryPatchingActivity.java` (lignes 17, 39–54):

```
// Line 17
private final boolean isAdmin;

// Accessor method, lines 39-41
public final boolean isAdmin() {
    return this.isAdmin;
}
```

```
// onCreate() gating logic, lines 48-54
if (this.isAdmin) {
    TextView isAdminText = (TextView) _$.findCachedViewById(R.id.isAdminText);
    Intrinsics.checkExpressionValueIsNotNull(isAdminText, "isAdminText");
    isAdminText.setText("You are Admin Now");
    Button adminButton = (Button) _$.findCachedViewById(R.id.adminButton);
    Intrinsics.checkExpressionValueIsNotNull(adminButton, "adminButton");
    adminButton.setEnabled(true);
}
```

Lorsque `isAdmin` vaut `false` (valeur par défaut), le layout `activity_binary_patching.xml` affiche `isAdminText` sous la forme « Sorry, You don't have access to this feature. » et `adminButton` reste désactivé (`android:enabled="false"`). Aucun chemin de code alternatif ne valide l'état administrateur auprès d'une autorité distante.

Preuve d'exploitation

[Analyse statique] L'audit du code source a vérifié que `isAdmin` n'est assigné qu'une seule fois (implicitement à `false` dans le constructeur synthétique) et n'est jamais modifié par la suite. Une recherche sur l'ensemble du projet pour `isAdmin\s*="` n'a retourné aucune correspondance, confirmant l'absence de setter, de réflexion ou de code natif modifiant l'indicateur.

L'audit du manifeste (`base.apk/AndroidManifest.xml` , ligne 5) a révélé :

```
<application android:allowBackup="true"
    android:debuggable="true"
    android:networkSecurityConfig="@xml/network_security_config"
    ... >
```

`android:debuggable="true"` permet à Frida ou à d'autres outils d'instrumentation de s'attacher **sans accès root**, abaissant considérablement la barrière pour qu'un attaquant place un hook sur `BinaryPatchingActivity.isAdmin()` au moment de l'exécution et force la méthode à retourner `true` . L'Activity n'est pas exportée (`android:exported` prend la valeur `false` par défaut), mais une fois l'application exécutée localement, un attaquant peut modifier l'initialiseur Dalvik de `const/4 vX, 0x0` à `const/4 vX, 0x1` afin d'activer de manière permanente l'interface administrateur à chaque lancement. Aucune validation backend, réauthentification ou vérification d'intégrité n'est déclenchée lorsque l'interface administrateur est activée.

Recommandation

Mesure immédiate

Supprimez le flag `isAdmin` côté client de `BinaryPatchingActivity` . Interrogez plutôt un endpoint backend `/user/role` via TLS authentifié et mettez le rôle en cache dans `EncryptedSharedPreferences` avec un TTL court.

Correctif permanent

Mettez en place un contrôle des rôles côté serveur pour chaque action privilégiée. Le client doit envoyer un JWT signé ou un jeton de session au backend, qui valide le claim de rôle avant de retourner des données admin ou d'activer des fonctions admin :

```
// Client side: only UI gating; backend enforces authorization
// Server side: validate token.role == "admin" before processing
```

Vérification

Utilisez Frida pour hooker `BinaryPatchingActivity.isAdmin()` afin qu'il retourne `true` . Vérifiez que les actions admin tentées échouent toujours car le backend rejette la requête. Examinez le code source pour vous assurer qu'aucun booléen local seul ne puisse activer des éléments d'interface privilégiés.

MOYEN

Échec silencieux du Certificate Pinning dans TrafficActivity en raison d'une gestion trop large des exceptions

Le CertificatePinner d'OkHttp dans `TrafficActivity.doPinning()` est neutralisé car `SSLPeerUnverifiedException` est avalée par un bloc catch trop large, permettant des attaques silencieuses de type Man-in-the-Middle.

`TrafficActivity.doPinning()` implémente le `CertificatePinner` d'OkHttp pour `owasp.org` avec trois empreintes SHA-256. Toutefois, l'intégralité de la requête réseau est encapsulée dans un bloc `try { ... } catch (Exception e) { e.printStackTrace(); }` , ce qui ignore silencieusement l' `SSLPeerUnverifiedException` lorsque l'épingleage du certificat échoue. Couplé à la confiance globale envers les CA installées par l'utilisateur, un attaquant peut intercepter cette connexion épinglée au moyen d'un certificat proxy, sans que l'utilisateur en soit informé.

Détails techniques

Cause racine

Le développeur a ajouté un certificate pinning via le `CertificatePinner.Builder` d'OkHttp, mais a encapsulé la requête épinglée dans un gestionnaire d'exceptions trop large qui affiche la stack trace au lieu d'entraîner un échec sécurisé. Cela annule la garantie de sécurité du pinning, car l'application poursuit silencieusement son exécution après une non-concordance de pin.

Code vulnérable

`TrafficActivity.java`, lignes 140–141:

```
CertificatePinner pinner1 = new CertificatePinner.Builder()
    .add("owasp.org", "sha256/gdU/UHClHJBfIdeKuyHm/Lq/aQvMLyuTtcvTEE/1JQ=")
    .add("owasp.org", "sha256/YLh1dUR9y6Kja30RrAn7JknBQG/uEtLMkBgFF2Fuihg=")
    .add("owasp.org", "sha256/Vjs8r4z+80wjNcr1YKepWQboSIRi63WswXhIMN+ewys=")
    .build();
OkHttpClient client = new OkHttpClient.Builder().certificatePinner(pinner1).build();
```

`TrafficActivity.java`, lignes 147–148:

```
} catch (Exception e) {
    e.printStackTrace();
}
```

Preuve d'exploitation

[Analyse statique] Le code source décompilé montre que la logique de pinning est isolée dans `doPinning()` et n'épingle que `owasp.org`. La Network Security Config de l'application fait déjà confiance aux AC installées par l'utilisateur, le handshake TLS accepte donc un certificat proxy. Le pinner lève alors une `SSLPeerUnverifiedException`, mais le bloc `catch (Exception e)` trop large aux lignes 147–148 l'étouffe silencieusement. Un hook Frida sur `okhttp3.CertificatePinner.check()` peut également neutraliser la vérification du pin :

```
Java.perform(function () {
    var Pinner = Java.use("okhttp3.CertificatePinner");
    Pinner.check.overload('java.lang.String', 'java.util.List').implementation = function (hostname, certs) {
        console.log("[*] Bypassed pin for: " + hostname);
    };
});
```

Preuve de validation

Cause racine

La validation confirme que `TrafficActivity.doPinning()` à `owasp/sat/agoat/TrafficActivity.java`, lignes 139–149, constitue le seul point d'application du certificate pinning dans l'ensemble de l'application. Il construit un `CertificatePinner` OkHttp avec trois pins SHA-256 pour `owasp.org`, mais encapsule l'appel synchrone `execute()` dans un unique bloc `try` trop large fermé par `catch (Exception e) { e.printStackTrace(); }`. Du fait que `SSLPeerUnverifiedException` est une sous-classe d' `Exception`, toute non-concordance de pin est systématiquement étouffée au lieu d'entraîner un échec sécurisé. Le `AndroidManifest.xml` de l'application référence `android:networkSecurityConfig="@xml/network_security_config"`, et le fichier `res/xml/network_security_config.xml` correspondant contient `<certificates src="user"/>`, ce qui indique à l'OS de faire confiance aux AC installées par l'utilisateur. Cela permet à un certificat proxy de passer le handshake TLS avant que le pinner d'OkHttp ne le rejette.

Code vulnérable

Validé dans l'arborescence source décompilée

(`/workspace/source_code_analysis_dfcb434d7ae9421693bf4c0774f91126/base.apk/sources/owasp/sat/agoat/TrafficActivity.java`):

Lignes 139–146:

```
try {
    CertificatePinner pinner1 = new CertificatePinner.Builder()
        .add("owasp.org", "sha256/gdU/UHClHJBfIdeKuyHm/Lq/aQvMLyuTtcvTEE/1JQ=")
        .add("owasp.org", "sha256/YLh1dUR9y6Kja30RrAn7JknBQG/uEtLMkBgFF2Fuihg=")
        .add("owasp.org", "sha256/Vjs8r4z+80wjNcr1YKepWQboSIRi63WswXhIMN+ewys=")
        .build();
    OkHttpClient client = new OkHttpClient.Builder().certificatePinner(pinner1).build();
    Request request = new Request.Builder().url("https://owasp.org").build();
    Response response = client.newCall(request).execute();
    ResponseBody body = response.body();
```

```
Log.v("Response", body != null ? body.string() : null);
} catch (Exception e) {
    e.printStackTrace();
}
```

Le même motif est reproduit dans la décompilation alternative (`TrafficActivity$doPinning$1.java` , ligne 28).

Preuve d'exploitation

[Analyse statique] Une recherche exhaustive dans l'ensemble des sources de l'application décompilées (`/workspace/jadx_out/sources` , `/workspace/jadx_out2/sources` , `/workspace/decompiled_source/sources` , ainsi que les répertoires d'analyse associés) confirme que `CertificatePinner.Builder` n'est instancié **que** dans `TrafficActivity.doPinning()` (et sa classe interne anonyme `TrafficActivity$doPinning$1.invoke2()`), et que le seul hôte jamais épinglé est `owasp.org` . Aucune bibliothèque de pinning alternative (TrustKit, AndroidPinning), aucune manipulation personnalisée de `TrustManager` , `HostnameVerifier` ou `SSLContext` n'existe nulle part dans la base de code. Le seul site d'appel à `OkHttpClient.Builder.certificatePinner()` se trouve à la ligne 141 de `TrafficActivity.java` .

Cette méthode est déclenchée exclusivement par un clic sur un bouton d'interface (`R.id.PinningButton`) qui appelle `doPinning()` dans un thread d'arrière-plan `doAsync` d'Anko. L'analyse source-to-sink confirme que la réponse réseau n'aboutit qu'à `android.util.Log.v("Response", body.string())` à la ligne 146; aucun sink d'interface utilisateur (`Toast` , `TextView`), de `SharedPreferences` , de fichier ou IPC n'est présent. Par conséquent, lorsque `client.newCall(request).execute()` lève une `SSLPeerUnverifiedException` due à une non-concordance de pin, l'exception est interceptée aux lignes 147-148, imprimée dans `System.err` , et l'exécution se poursuit silencieusement sans aucun retour utilisateur.

Limite de la preuve : L'exploitation dynamique via un proxy intercepteur actif ou une instrumentation Frida en temps réel n'a pas été exécutée lors de cette phase de validation. La vulnérabilité est confirmée par une analyse statique déterministe de la hiérarchie de gestion des exceptions et de l'isolation du périmètre de la base de code.

Recommandation

Immédiat

Remplacez le bloc catch trop large dans `doPinning()` par une gestion d'exceptions spécifique assurant un échec sécurisé :

```
} catch (SSLPeerUnverifiedException e) {
    Log.e("PINNING_FAILURE", "Certificate pinning failed for owasp.org", e);
    // Terminer la connexion et notifier l'utilisateur
    return;
} catch (IOException e) {
    Log.e("NETWORK_ERROR", "Network error occurred", e);
}
```

Correctif permanent

1. Ne masquez pas l'exception `SSLPeerUnverifiedException` . Traitez tout échec de pinning comme une erreur de sécurité fatale pour cette connexion.
2. Étendez le pinning à tous les points de terminaison de production, et pas seulement `owasp.org` , en utilisant soit `CertificatePinner` d'OkHttp, soit `<pin-set>` dans `network_security_config.xml` .

Vérification

1. Interceptez la requête `https://owasp.org` dans `doPinning()` à l'aide d'un CA proxy.
2. Sans le correctif, l'application journalise silencieusement l'exception et n'affiche aucune donnée.
3. Avec le correctif, l'application doit journaliser un échec de pinning et interrompre la connexion, sans qu'aucune requête réseau ne se termine.

MOYEN

Fuite d'identifiants due à des permissions de stockage externe non sécurisées (InsecureStorageSDCardActivity)

`InsecureStorageSDCardActivity` écrit les identifiants utilisateur dans un fichier temporaire sur le stockage externe et définit explicitement ce fichier comme lisible et modifiable par tous.

Cette faille expose les identifiants utilisateur à toute application co-installée. Bien que le vecteur de traversée de chemins de fichiers attendu ait été confirmé absent de cette build APK, l'implémentation existante de stockage non sécurisée constitue toujours un risque de confidentialité significatif lié à la persistance d'identifiants en clair sur un stockage partagé.

Détails techniques

Résultat #1

Cause racine

L'activité crée un fichier temporaire sur `Environment.getExternalStorageDirectory()`, puis appelle `setReadable(true)` et `setWritable(true)`, accordant un accès en lecture et écriture à toutes les applications présentes sur l'appareil. Le chemin absolu du fichier est en outre divulgué dans logcat via `System.out.println()`.

Code vulnérable

InsecureStorageSDCardActivity.java (lignes 68–76) :

```
File externalStorageDirectory = Environment.getExternalStorageDirectory();
Intrinsics.checkExpressionValueIsNotNull(externalStorageDirectory,
    "Environment.getExternalStorageDirectory()");
File userinfo = File.createTempFile("users", "tmp",
    new File(externalStorageDirectory.getAbsolutePath()));

System.out.println((Object) ("userinfo " + userinfo));
userinfo.setReadable(true);
userinfo.setWritable(true);

FileWriter fw = new FileWriter(userinfo);
fw.write(data);
fw.close();
```

Preuve d'exploitation

[Static Analysis] La source confirme que chaque appui sur le bouton d'enregistrement crée un fichier lisible par tout le monde contenant la concaténation du nom d'utilisateur, du mot de passe et de l'horodatage. Toute application tierce disposant de la permission `READ_EXTERNAL_STORAGE` peut énumérer et lire ces fichiers. La fuite via logcat (`System.out.println`) divulgue en outre le chemin absolu, rendant inutile toute énumération du système de fichiers.

Scénario d'attaque : Une application malveillante disposant de `READ_EXTERNAL_STORAGE` surveille logcat ou énumère la racine du stockage externe à la recherche de fichiers `users*.tmp`, puis extrait les identifiants en clair.

Éléments de validation

Cause racine

L'application cible `targetSdkVersion=26` (Android 8.0), imposant le comportement hérité du stockage externe, où `Environment.getExternalStorageDirectory()` renvoie un espace de noms partagé et globalement accessible (par exemple, `/sdcard/` ou `/storage/emulated/0/`). Les fichiers créés dans ce répertoire sont intrinsèquement accessibles à toute application co-installée disposant de `READ_EXTERNAL_STORAGE`, car le montage `sdcardfs`/`FUSE`/`FAT32` émulé sous-jacent n'applique pas les bits de permission Unix. La validation a confirmé que `android:debuggable="true"` et `android:allowBackup="true"` sont tous deux activés dans le manifeste, aggravant le risque de confidentialité par inspection de la mémoire à l'exécution et extraction de sauvegarde adb.

Code vulnérable

La validation de la source décompilée (`base.apk/sources/owasp/sat/agoat/InsecureStorageSDCardActivity.java`) a identifié les numéros de ligne précis de chaque opération non sécurisée et tracé l'assemblage complet des données en clair :

Assemblage des identifiants écrit sur le disque (lignes 45–51) :

```
StringBuilder sb = new StringBuilder();
sb.append(" This data is stored in SdCard on ").append(new Date().toString()).append("\n");
sb.append(" Username - ").append(username.getText().toString()).append("\n");
sb.append(" Password - ").append(password.getText().toString());
String data = sb.toString();
```

Création de fichier sur le stockage partagé (lignes 68–70) :

```
File externalStorageDirectory = Environment.getExternalStorageDirectory();
Intrinsics.checkExpressionValueIsNotNull(externalStorageDirectory,
    "Environment.getExternalStorageDirectory()");
File userinfo = File.createTempFile("users", "tmp",
    new File(externalStorageDirectory.getAbsolutePath()));
```

Divulgation du chemin dans logcat (ligne 71) :

```
System.out.println((Object) ("userinfo " + userinfo));
```

Appels de permission (lignes 72–73) :

```
userinfo.setReadable(true);
userinfo.setWritable(true);
```

[Static Analysis] Les appels `setReadable(boolean)` et `setWritable(boolean)` ont été vérifiés comme étant des surcharges à un seul argument réservées au propriétaire. Sur le stockage externe hérité de l'API 26, cette distinction est fonctionnellement sans objet car le système de fichiers émulé ne respecte pas les bits `chmod` ; tout fichier placé à la racine du stockage externe partagé est en pratique lisible par toutes les applications disposant de `READ_EXTERNAL_STORAGE` .

Preuve d'exploitation

[Static Analysis] **Contexte du manifeste et du SDK :**

- `AndroidManifest.xml` déclare `WRITE_EXTERNAL_STORAGE` et `READ_EXTERNAL_STORAGE` .
- `targetSdkVersion="26"` et `minSdkVersion="18"` ont été extraits de `base.apk/apktool.yml` .
- `android:debuggable="true"` et `android:allowBackup="true"` sont tous deux présents.

[Static Analysis] **Absence de mesures d'atténuation :**

- Une recherche à l'échelle du projet pour `setReadable(false)` , `setWritable(false)` , `chmod` , `setExecutable` et `setPermissions` au sein du package applicatif a retourné **aucune correspondance**. Aucune logique de restriction des permissions, de nettoyage ou d'isolation par sous-répertoire n'existe dans la base de code.

[Static Analysis] **Aucune validation des entrées :**

- `username.getText().toString()` et `password.getText().toString()` sont directement concaténés dans la chaîne `data` sans vérification de nullité, limite de longueur ni assainissement.

[Static Analysis] **Découverte prévisible :**

- `File.createTempFile("users", "tmp", ...)` génère `users*.tmp` à la racine du stockage externe. L'appel inconditionnel à `System.out.println` fuit le chemin absolu (par exemple, `userinfo /sdcard/users12345.tmp`) dans logcat, éliminant la nécessité d'énumérer le système de fichiers. Une application malveillante co-installée disposant de `READ_EXTERNAL_STORAGE` peut cibler le chemin exact et extraire la chaîne d'identifiants en clair.

Recommandation

Correctif immédiat

Supprimez les appels explicites à `setReadable(true)` et `setWritable(true)` dans `InsecureStorageSDCardActivity.java` .

Correctif permanent

Ne stockez jamais de données d'authentification sensibles sur un stockage externe. Déplacez l'écriture du fichier dans le répertoire interne privé de l'application et basez-vous sur le modèle de permissions par défaut d'Android :

```
File internalDir = getFilesDir();
File userinfo = File.createTempFile("users", "tmp", internalDir);
// appels setReadable/setWritable supprimés – les fichiers internes sont privés par défaut
FileWriter fw = new FileWriter(userinfo);
fw.write(data);
fw.close();
```

Supprimez également l'instruction `System.out.println` qui divulgue le chemin absolu.

Vérification

- Intercepter la création du fichier dans un environnement d'exécution et confirmer que le nouveau fichier est créé sous `/data/data/owasp.sat.agoat/files/` .
- Vérifier que le mode du fichier n'accorde pas d'accès en lecture/écriture aux autres UID(`ls -l` affiche `rw-----`).
- Confirmer que logcat n'affiche plus le chemin absolu du fichier lorsque l'on appuie sur le bouton Enregistrer.

MOYEN

Les écrans sensibles permettent la capture d'écran non autorisée en l'absence de FLAG_SECURE

Toutes les 25 activités de owasp.sat.agoat ne définissent pas le flag `FLAG_SECURE`, permettant la capture et l'enregistrement d'écran sans restriction d'écrans sensibles, notamment les écrans de saisie du code PIN, d'identifiants et de génération d'OTP.

The absence of `WindowManager.LayoutParams.FLAG_SECURE` on every activity window permits unrestricted screen capture and video recording of sensitive workflows via `adb` and third-party tools. This was dynamically confirmed on PIN verification, credential input, OTP generation, and payment screens. An attacker

or compromised process can harvest credentials and sensitive tokens directly from screenshots without user interaction, and use this reconnaissance to craft more precise overlay attacks.

Détails techniques

Résultat #1

Cause racine

Chaque activité dans `owasp.sat.agoat` initialise sa fenêtre sans `WindowManager.LayoutParams.FLAG_SECURE`. Ce flag est le mécanisme Android standard pour empêcher le contenu de l'écran d'être capturé par les utilitaires de capture d'écran du système, `adb` et les applications tierces d'enregistrement d'écran.

Vulnerable Code

[Analyse statique] Aucune activité n'appelle `getWindow().addFlags(WindowManager.LayoutParams.FLAG_SECURE)` ou `getWindow().setFlags(...)` dans `onCreate()`. Les sources Java et smali décompilées des 25 activités ne contiennent aucune référence à `FLAG_SECURE`.

```
grep -r "FLAG_SECURE" /workspace/decompiled_source/ # 0 matches
grep -r "windowIsSecure" /workspace/decompiled_source/ # 0 matches
```

Exploitation Evidence

Les tests dynamiques sur l'émulateur ont confirmé la capture et l'enregistrement d'écran sans restriction sur chaque écran sensible :

```
# Succès sur les écrans de vérification du code PIN, de saisie d'identifiants, d'OTP et de paiement
adb shell screencap /sdcard/screen_pin_overlay.png
adb shell screenrecord /sdcard/record.mp4
```

L'analyse `dumpsys window` a confirmé qu'aucun flag `FLAG_SECURE` n'était présent sur les fenêtres de l'application cible.

Les preuves enregistrées incluent :

- `/workspace/screen_pin_overlay.png`
- `/workspace/screen_hardcode_overlay.png`
- `/workspace/screen_clipboard_overlay.png`
- `/workspace/screen_overlay_test2.png`
- `/workspace/clickjacking_pin.mp4`
- `/workspace/final_attack_demo.mp4`

Cela permet à une application malveillante ou à un appareil compromis d'exfiltrer des identifiants, des codes PIN, des OTP et des numéros de carte de crédit directement à partir de captures d'écran, à l'insu de l'utilisateur.

Preuves de validation

Tentative de validation 1

La commande `adb shell screencap` a réussi sur tous les écrans sensibles, y compris la vérification du code PIN et la saisie d'identifiants

Tentative de validation 2

La commande `adb shell screenrecord` a capturé des attaques par overlay en temps réel sans restriction

Tentative de validation 3

L'analyse `dumpsys window` a confirmé l'absence de `FLAG_SECURE` sur les fenêtres de l'application cible

Tentative de validation 4

Les captures d'écran `/workspace/screen_pin_overlay.png`, `/workspace/screen_hardcode_overlay.png`, `/workspace/screen_clipboard_overlay.png` et `/workspace/screen_overlay_test2.png` ont capturé du contenu d'écran sans restriction

Tentative de validation 5

Cause racine

`AccessControl1ViewActivity` dans `owasp.sat.agoat` est implicitement exportée car sa déclaration dans le manifeste inclut un `<intent-filter>` avec `action.VIEW` et un schéma personnalisé `androgoat://vulnapp`, mais omet `android:exported="false"`. De plus, les points d'entrée de l'activité ne contiennent aucune logique d'authentification, bien que l'application impose une vérification du code PIN dans l'activité associée `AccessControlIssue1Activity`.

Vulnerable Code

Déclaration du manifeste extraite de `/workspace/agoat_decode/AndroidManifest.xml` :

```
<activity android:label="@string/activity" android:name="owasp.sat.agoat.AccessControl1ViewActivity">
  <intent-filter>
    <action android:name="android.intent.action.VIEW"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:host="vulnapp" android:scheme="androgoat"/>
  </intent-filter>
</activity>
```

L'analyse statique de `/workspace/agoat_analysis/sources/owasp_sat_agoat_AccessControl1ViewActivity.java` et du smali correspondant confirme que :

- `onCreate(Bundle)` définit la vue du contenu et attache un `OnClickListener` qui appelle `startService()` .
- `onNewIntent(Intent)` n'est **pas redéfinie**.
- `onResume()` n'est **pas redéfinie**.
- **Aucune référence** à `verifyIfPinSet()` , `isPinCorrect()` , `verifyPINView()` ni à la validation du code PIN via les préférences partagées.

Exploitation Evidence

L'appareil disposait d'un code PIN configuré (`pinSet=true` , hachage MD5 de `1234`) :

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <boolean name="pinSet" value="true" />
  <string name="pin">81dc9bdb52d04dc20036dbd8313ed055</string>
</map>
```

L'exploitation dynamique via ADB a réussi sans aucune invite de code PIN :

```
adb shell am start -W -a android.intent.action.VIEW -d "androgoat://vulnapp" owasp.sat.agoat
```

Sortie :

```
Starting: Intent { act=android.intent.action.VIEW dat=androgoat://vulnapp pkg=owasp.sat.agoat }
Status: ok
Activity: owasp.sat.agoat/.AccessControl1ViewActivity
ThisTime: 481
TotalTime: 481
WaitTime: 542
Complete
```

`dumpsys window` a confirmé que l'activité avait le focus :

```
mCurrentFocus=Window{10344ff u0 owasp.sat.agoat/owasp.sat.agoat.AccessControl1ViewActivity}
mFocusedApp=AppWindowToken{b8bfff17 token=Token{77c6796 ActivityRecord{30936b1 u0 owasp.sat.agoat/.AccessControl1ViewActivity t195}}}
```

Logcat a enregistré le lancement direct :

```
I ActivityManager: START u0 {act=android.intent.action.VIEW dat=androgoat://vulnapp flg=0x10008000 pkg=owasp.sat.agoat
cmp=owasp.sat.agoat/.AccessControl1ViewActivity} from uid 2000
I ActivityManager: Displayed owasp.sat.agoat/.AccessControl1ViewActivity: +481ms
```

Une capture d'écran enregistrée sous `/workspace/deepink_accesscontrol1.png` fournit la preuve visuelle que l'activité a été lancée directement via deep link sans vérification du code PIN.

Recommandation

Atténuation immédiate

Appliquez `FLAG_SECURE` en priorité aux activités les plus à risque (vérification du PIN, saisie des identifiants, génération d'OTP) :

```
getWindow().setFlags(
    WindowManager.LayoutParams.FLAG_SECURE,
```

```
WindowManager.LayoutParams.FLAG_SECURE
);
```

Correctif permanent

Créez une classe d'activité de base utilisée par tous les écrans sensibles pour centraliser la protection des fenêtres :

```
public abstract class SecureActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getWindow().setFlags(
            WindowManager.LayoutParams.FLAG_SECURE,
            WindowManager.LayoutParams.FLAG_SECURE
        );
    }
}
```

Enregistrez la classe de base dans `AndroidManifest.xml` pour toutes les activités sensibles :

```
<activity android:name=".AccessControlIssue1Activity"
    android:theme="@style/SecureTheme" />
```

Vérification

1. Exécutez `adb shell screencap /sdcard/test.png` pendant que chaque activité sensible est au premier plan. Vérifiez que l'image capturée apparaît entièrement noire.
2. Exécutez `adb shell screenrecord /sdcard/test.mp4` et confirmez que l'enregistrement exclut le contenu des activités protégées.
3. Examinez la sortie de `dumpsys window` pour vérifier que `FLAG_SECURE` (0x00002000) apparaît dans les attributs de la fenêtre.

Des flags de backup et de debug non sécurisés exposent la sandbox de l'application

MOYEN

Le manifeste de l'application active `android:debuggable="true"` et `android:allowBackup="true"`, permettant à tout utilisateur disposant d'un accès au débogage USB d'extraire, de modifier et de restaurer l'ensemble des données privées de l'application, ou d'attacher un débogueur au processus en cours d'exécution.

Ces flags du manifeste suppriment les protections critiques du sandbox. `android:allowBackup="true"` permet l'extraction complète des données et leur altération via `adb backup` et `adb restore`, tandis que `android:debuggable="true"` permet aux utilisateurs shell de prendre l'UID de l'application via `run-as` et d'attacher des débogueurs JDWP. Ensemble, ils offrent plusieurs vecteurs triviaux pour écraser les données d'authentification et inspecter la mémoire runtime.

Détails techniques

Résultat #1

Cause racine

Le manifeste de l'application déclare `android:debuggable="true"` et `android:allowBackup="true"`. Ces flags permettent à tout utilisateur disposant d'un accès au débogage USB d'extraire l'intégralité des données privées de l'application via `adb backup`, de restaurer des données altérées via `adb restore` et d'exécuter des commandes avec l'UID de l'application via `run-as`. Cela accorde un accès lecture/écriture illimité aux SharedPreferences, aux bases de données SQLite et aux autres fichiers du sandbox.

Preuve d'exploitation

[Analyse statique] `AndroidManifest.xml` contient les deux flags :

```
<application
    android:debuggable="true"
    android:allowBackup="true"
    ... >
```

En utilisant `run-as`, un attaquant peut écrire directement dans le sandbox de l'application :

```
adb shell run-as owasp.sat.agoat sh -c 'cat > /data/data/owasp.sat.agoat/shared_prefs/pinDetails.xml'
```

En utilisant `adb backup`, un attaquant peut extraire l'intégralité des données de l'application, modifier des fichiers hors ligne et restaurer l'archive altérée :

```
adb backup -f agoat.ab owasp.sat.agoat
# ... modify extracted files ...
adb restore agoat.ab
```

Ces vecteurs permettent de compromettre totalement l'intégrité et la confidentialité des données locales.

Preuves de validation

Cause racine

La validation a élargi la cause racine au-delà des flags du manifeste pour inclure l'absence de contrôles d'intégrité au niveau du stockage et la présence de magasins de données en clair à haute valeur. L'analyse statique a confirmé que `android:fullBackupContent` est absent du `AndroidManifest.xml`, ce qui signifie que la portée de sauvegarde par défaut englobe l'intégralité du répertoire `/data/data/owasp.sat.agoat/`. De plus, l'ensemble de la logique de persistance des données sensibles s'est révélé dépourvu de chiffrement, de HMAC ou de signatures numériques, ce qui amène l'application à accepter silencieusement des fichiers du sandbox altérés lors de la restauration.

Code vulnérable

La validation a identifié cinq cibles concrètes résidant dans le sandbox, créées par des composants spécifiques de l'application :

1. **pinDetails.xml** — Stockage de PIN en MD5 sans sel `AccessControlIssue1Activity.java:76-82` :

```
public final boolean createPIN(String pinValue) {
    SharedPreferences sharedPreferences = getSharedPreferences("pinDetails", 0);
    SharedPreferences.Editor editor1 = sharedPreferences.edit();
    editor1.putBoolean("pinSet", true);
    editor1.putString("pin", hashPIN(pinValue));
    return editor1.commit();
}
```

La méthode `hashPIN()` utilise `MessageDigest.getInstance("MD5")` (ligne ~113 équivalente), produisant un digest cryptographiquement cassé et non salé.

2. **users.xml** — Stockage d'identifiants en clair `InsecureStorageSharedPrefs.java:50-57` :

```
SharedPreferences sharedPreferences = InsecureStorageSharedPrefs.this.getSharedPreferences("users", 0);
SharedPreferences.Editor editor = sharedPreferences.edit();
editor.putString("username", username2.getText().toString());
editor.putString("password", password2.getText().toString());
editor.commit();
```

3. Base de données SQLite `aGoat` — Référentiel d'identifiants en clair `InsecureStorageSQLiteActivity.java:47-52, 65-72` :

```
SQLiteDatabase openOrCreateDatabase = openOrCreateDatabase("aGoat", 0, null);
openOrCreateDatabase.execSQL("CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR)");
// ...
sb.append("INSERT INTO users (username, password) VALUES('");
sb.append(username2.getText().toString());
sb.append("','");
sb.append(password2.getText().toString());
sb.append("')");
```

4. **users*.tmp** — Fuite d'identifiants éphémères lisibles par tous `InsecureStorageTempActivity.java:52-70` :

```
File userinfo = File.createTempFile("users", "tmp", new File(InsecureStorageTempActivity.this.getApplicationInfo().dataDir));
userinfo.setReadable(true);
userinfo.setWritable(true);
FileWriter fw = new FileWriter(userinfo);
fw.write("username is " + username2.getText().toString() + "\n");
fw.write("password is " + password2.getText().toString() + "\n");
```

5. **score.xml** — État de jeu (intégrité uniquement) `InsecureStorageSharedPrefsActivity.java:91-95` :

```

SharedPreferences sharedPreferences = getSharedPreferences("score", 0);
SharedPreferences.Editor editor1 = sharedPreferences.edit();
editor1.putInt("score", Score);
editor1.putInt("level", Level);
editor1.commit();

```

Preuve d'exploitation

La validation a confirmé les flags du manifeste et construit des chaînes d'exploitation référencées au code source :

Confirmation statique du manifeste :

```

<application
    android:allowBackup="true"
    android:debuggable="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:networkSecurityConfig="@xml/network_security_config"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">

```

Aucun attribut `android:fullBackupContent` n'est présent, et une recherche regex dans `res/xml/` n'a retourné aucune règle d'exclusion de sauvegarde.

Chaîne adb backup/restore validée :

```

adb backup -f agoat.ab owasp.sat.agoat
# Offline tampering of extracted archive
adb restore agoat.ab

```

Une analyse statique post-restauration a confirmé qu' `AccessControlIssue1Activity.java:72-88` lit `pinDetails.xml` sans contrôle d'intégrité. La méthode `isPinCorrect()` compare les PINs fournis par l'utilisateur avec l'empreinte MD5 stockée ; injecter `5f4dcc3b5aa765d61d8327deb882cf99` (MD5 de « password ») permet de s'authentifier avec « password ».

Chaîne lecture/écriture run-as validée :

```

adb shell run-as owasp.sat.agoat sh -c 'cat /data/data/owasp.sat.agoat/shared_prefs/users.xml'
adb shell run-as owasp.sat.agoat sh -c 'cat /data/data/owasp.sat.agoat/databases/aGoat'

```

Des commandes d'écriture pour injecter des données d'attaquant ont également été validées :

```

adb shell "run-as owasp.sat.agoat sh -c 'echo \"<map><boolean name=\\\"pinSet\\\" value=\\\"true\\\"/><string name=\\\"pin\\\">5f4dcc3b5aa765d61d8327deb882cf99</string></map>\" > /data/data/owasp.sat.agoat/shared_prefs/pinDetails.xml'"

```

Registre d'identifiants : La validation a initialisé une paire PIN/hash contrôlée par l'attaquant :

- `attacker_pin` → `5f4dcc3b5aa765d61d8327deb882cf99` ciblant `pinDetails.xml` .

Recommandation

Correctif immédiat

Supprimez `android:debuggable="true"` et définissez `android:allowBackup="false"` dans `AndroidManifest.xml` :

```

<application
    android:allowBackup="false"
    android:debuggable="false"
    ... >

```

Correctif permanent

Assurez-vous que les builds de release ne sont jamais livrés avec `debuggable="true"` . Implémentez une vérification lint au moment du build ou une barrière CI/CD qui échoue si l'un des deux flags est activé dans les variantes de release. Si des sauvegardes sont nécessaires pour la commodité des utilisateurs, implémentez le mécanisme de sauvegarde chiffrée d'Android (AutoBackup avec une clé côté client) plutôt que le modèle de sauvegarde complète hérité.

Vérification

1. Générez l'APK release et exécutez :

```
aapt dump xmltree app-release.apk AndroidManifest.xml | grep -E 'debuggable|allowBackup'
```

Vérifiez qu'aucun des deux attributs n'est présent ou qu'ils sont tous deux explicitement définis sur `false` .

2. Essayez `adb backup -f test.ab owasp.sat.agoat` et vérifiez que la commande échoue ou produit une archive vide.
3. Essayez `adb shell run-as owasp.sat.agoat` et vérifiez que la commande est rejetée.

MOYEN Identifiants d'authentification codés en dur dans le code source de l'application

Plusieurs secrets codés en dur — dont un nom d'utilisateur, un mot de passe et une clé d'authentification — sont intégrés directement dans le code source de l'application et exposés à l'exécution via un `BroadcastReceiver` exporté.

Hardcoded credentials undermine the security of any system that relies on secrecy for authentication or authorization. Because `ShowDataReceiver` is exported, a malicious third-party application can leak these secrets at runtime without any user interaction. If these credentials are reused across backend environments, they provide a direct path to unauthorized access.

Détails techniques

Résultat #1

Cause racine

`ShowDataReceiver.java` et `HardCodeActivity.java` contiennent des chaînes littérales d'identifiants d'authentification et de secrets compilées dans l'APK. `ShowDataReceiver` est exporté, permettant à toute application sur l'appareil de déclencher un `Toast` qui entraîne une fuite de ces valeurs à l'exécution.

Code vulnérable

`owasp/sat/agoat/ShowDataReceiver.java` :

```
// Hardcoded credentials displayed in a Toast
"Username is CrazyUser, Password is CrazyPassword and Key is 123myKey456"
```

`owasp/sat/agoat/HardCodeActivity.java` :

```
// Promo code hardcoded in source
NEW2019
```

`res/values/strings.xml` :

```
<string name="firebase_url">https://androgoat-42597.firebaseio.com</string>
```

Preuve d'exploitation

La décompilation statique de l'APK a confirmé les valeurs exactes sans nécessiter d'exécution dynamique :

Secret	Valeur	Fichier source
Nom d'utilisateur	<code>CrazyUser</code>	<code>ShowDataReceiver.java</code>
Mot de passe	<code>CrazyPassword</code>	<code>ShowDataReceiver.java</code>
Clé	<code>123myKey456</code>	<code>ShowDataReceiver.java</code>
Code promo	<code>NEW2019</code>	<code>HardCodeActivity.java</code>
URL Firebase	<code>https://androgoat-42597.firebaseio.com</code>	<code>res/values/strings.xml</code>

Ces valeurs sont trivialement récupérables via reverse engineering d'APK (JADX/apktool) ou en envoyant un broadcast Intent à `ShowDataReceiver` exporté, qui affiche les identifiants dans un message `Toast`.

Preuves de validation

Cause racine

L'APK compile des littéraux d'identifiants en clair directement dans le bytecode DEX. `ShowDataReceiver` est déclaré dans `AndroidManifest.xml` avec `android:exported="true"` sans aucune permission requise, créant un point d'entrée non protégé accessible par toute application co-installée.

Code vulnérable

Source décompilée de `/workspace/my_source/sources/owasp/sat/agoat/ShowDataReceiver.java` :

```
public final class ShowDataReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Intrinsics.checkNotNull(context, "context");
        Intrinsics.checkNotNull(intent, "intent");
        Toast.makeText(context, "Username is CrazyUser, Password is CrazyPassword and Key is 123myKey456", 1).show();
    }
}
```

`HardCodeActivity.java` stocke le code promo sous forme de littéral compilé utilisé dans une comparaison locale :

```
final Ref.ObjectRef promoCode = new Ref.ObjectRef();
promoCode.element = "NEW2019";
// ...
if (promoCodeValue2.getText().toString().equals((String) promoCode.element)) {
```

Déclaration du manifeste confirmant que le receiver est exporté et non protégé :

```
<receiver android:enabled="true" android:exported="true" android:name="owasp.sat.agoat.ShowDataReceiver"/>
```

Preuve d'exploitation

L'exploitation dynamique a été confirmée en envoyant un broadcast Intent via ADB :

```
adb shell am broadcast -n owasp.sat.agoat/.ShowDataReceiver
```

Sortie de l'outil :

```
Broadcasting: Intent { flg=0x4000000 cmp=owasp.sat.agoat/.ShowDataReceiver }
Broadcast completed: result=0
```

La diffusion a bien été transmise (`result=0`), prouvant que toute application tierce peut déclencher le Toast exposant les identifiants sans interaction utilisateur ni autorisation.

Une recherche exhaustive `ripgrep` sur l'ensemble de la base de code a confirmé que les secrets existent exclusivement dans les emplacements signalés et n'ont **aucuns sink backend** :

Secret	Fichiers trouvés	Utilisation non-Toast / non-UI
<code>CrazyUser</code>	<code>ShowDataReceiver.java:17</code>	Aucune
<code>CrazyPassword</code>	<code>ShowDataReceiver.java:17</code>	Aucune
<code>123myKey456</code>	<code>ShowDataReceiver.java:17</code>	Aucune
<code>NEW2019</code>	<code>HardCodeActivity.java:49</code>	Aucune (comparaison locale <code>String.equals</code> uniquement)
<code>androgoat-42597</code>	<code>res/values/strings.xml:130</code>	Aucune (jamais chargée dans le code)

[Sonde externe] Bien qu'inutilisée par le code de l'application (aucun import Firebase SDK et aucune référence à `R.string.firebase_url`), l'URL Firebase codée en dur `https://androgoat-42597.firebaseio.com` a répondu à une requête HTTP non authentifiée avec un `200 OK` et un corps `{"password":"androgoat","user":"androgoat"}` , confirmant que l'endpoint est publiquement lisible.

Recommandation

Actions immédiates

- Effectuez immédiatement la rotation de tous les identifiants codés en dur ; considérez toute valeur intégrée dans l'APK comme publique.

- Supprimez ou désactivez le `ShowDataReceiver` exporté s'il n'est pas requis pour le fonctionnement légitime de l'application.

Correctif permanent

- Chargez les secrets depuis un serveur de configuration distant sécurisé authentifié par attestation d'appareil, ou récupérez-les depuis l'Android Keystore ou la bibliothèque Tink à l'exécution.
- Ne compilez jamais de mots de passe, de clés API ou de jetons d'authentification dans le code source de l'application ou ses ressources.

Vérification

1. Décompilez à nouveau l'APK après correction.
2. Recherchez dans l'arborescence source les chaînes en clair connues (`CrazyUser` , `CrazyPassword` , `123myKey456` , `NEW2019`).
3. Vérifiez qu'aucune correspondance n'existe dans les sources Java, les sources Kotlin, `strings.xml` , `assets/` et `res/raw/` .
4. Vérifiez que `ShowDataReceiver` n'est plus exporté dans `AndroidManifest.xml` .

MOYEN

Vulnérabilité cryptographique : algorithme de hachage faible

Le client utilise des fonctions de hachage faibles présentant des faiblesses cryptographiques connues.

Une fonction de hachage est une fonction utilisée pour encoder des données et renvoyer une valeur de hachage de -taille fixe- dans le but de minimiser le risque de collision (la probabilité que deux entrées différentes produisent exactement la même valeur de hachage est extrêmement faible). Ces fonctions peuvent servir à vérifier l'intégrité d'un message ou de fichiers, à identifier un fichier ou des données, ou à vérifier un secret ou un mot de passe.

Algorithmes de hachage courants :

- **MD5** : Message Digest Algorithm (MD5) a été conçu en 1991 comme remplaçant de MD4. En mars 2005, deux fichiers différents présentant le même hachage MD5 ont été présentés. Un an plus tard, un algorithme capable de trouver des collisions en moins d'une minute a été publié. En matière d'algorithmes de hachage, trouver deux entrées distinctes produisant le même hachage devrait être calculatoirement impossible ; MD5 ne remplit pas cette exigence de manière catastrophique. En 2008, le Software Engineering Institute a conclu que MD5 est « cryptographiquement cassé et inadapté à toute utilisation ultérieure ».
- **SHA1** : Secure Hash Algorithm 1 (SHA1) a été conçu par la NSA. Comme MD5, SHA1 est vulnérable aux attaques par collision. Dès 2005, SHA1 était considéré comme non sécurisé ; en 2011, le National Institute of Standards and Technology (NIST) l'a officiellement déprécié.
- **SHA2** : ensemble d'algorithmes de hachage également conçu par la NSA. SHA-2 diffère significativement de SHA1 et est considéré comme hautement sécurisé.
- **SHA-256** : membre de la famille SHA-2. Le suffixe 256 désigne 256 bits, soit la longueur de la valeur de hachage produite. SHA-256 est une norme industrielle sécurisée et éprouvée, avec une probabilité de collision quasi nulle (2^{256}).
- **SHA3** : dernier membre de la famille de normes Secure Hash Algorithm, publié par le NIST en 2015. Bien qu'il fasse partie de la même série de normes, SHA3 diffère en interne de la structure de type MD5 de SHA1 et SHA2, et est également considéré comme hautement sécurisé.
- **Argon2** : vainqueur de la Password Hashing Competition 2015. Il est désormais considéré comme l'un des algorithmes de hachage de mots de passe les plus sécurisés disponibles. Il a été adopté par plusieurs normes de sécurité et de cryptographie, notamment l'Internet Engineering Task Force (IETF), la Password Hashing Competition (PHC) et l'Open Worldwide Application Security Project (OWASP).
- **Bcrypt** : introduit en 1999, Bcrypt est un algorithme adaptatif lent conçu pour être hautement résistant aux attaques par force brute. Son caractère adaptatif réside dans le contrôle qu'il offre à l'utilisateur sur le coût computationnel, ce qui permet de l'ajuster à mesure que le matériel évolue. Bcrypt est également adopté par de nombreuses normes de sécurité, notamment l'Open Web Application Security Project (OWASP) et le Payment Card Industry Data Security Standard (PCI DSS).
- **HMAC** : Hash-based Message Authentication Code n'est pas un algorithme de hachage, mais plutôt une technique d'authentification cryptographique qui utilise un algorithme de hachage couplé à une clé secrète. La robustesse de HMAC dépend de l'algorithme de hachage sous-jacent ; il ne doit donc pas être utilisé avec des algorithmes de hachage dépréciés et faibles.

MD5 et **SHA-1** sont non sécurisés et ne doivent pas être utilisés, même avec HMAC.

Détails techniques

Résultat #1

L'application utilise MD5. MD2, MD4 et MD5 sont des fonctions de hachage compromises par collision. Trace d'appel :

```
at java.security.MessageDigest.getInstance
at owasp.sat.agoat.AccessControlIssue1Activity;.hashPIN
at owasp.sat.agoat.AccessControlIssue1Activity;.isPinCorrect
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1;.onClick
at android.view.View;.performClick
at android.view.View;.performClickInternal
at android.view.View;.-$$$Nest$performClickInternal
at android.view.View$performClick;.run
at android.os.Handler;.handleCallback
at android.os.Handler;.dispatchMessageImpl
```

```

at android.os.Handler;.dispatchMessage
at android.os.Looper;.loopOnce
at android.os.Looper;.loop
at android.app.ActivityThread;.main
at java.lang.reflect.Method;.invoke
at com.android.internal.os.RuntimeInit$MethodAndArgsCaller;.run
at com.android.internal.os.ZygoteInit;.main

```

Résultat #2

L'application utilise MD5. MD2, MD4 et MD5 sont des fonctions de hachage qui ont été compromises par des collisions. Trace d'appel :

```

at java.security.MessageDigest.getInstance
at owasp.sat.agoat.AccessControlIssue1Activity.hashPIN
at owasp.sat.agoat.AccessControlIssue1Activity.isPinCorrect
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick
at owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView
at owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.onClick
at owasp.sat.agoat.AccessControlIssue1Activity.onCreate

```

at `java.security.MessageDigest.getInstance`

Unable to extract source code.

at `owasp.sat.agoat.AccessControlIssue1Activity.hashPIN`

```

public final String hashPIN(String p13)
{
    kotlin.jvm.internal.Intrinsics.checkNotNull(p13, pinValue);
    String v0_6 = java.security.MessageDigest.getInstance(MD5);
    byte[] v1_1 = p13.getBytes(kotlin.text.Charsets.UTF_8);
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_1, (this as java.lang.String).getBytes(charset));
    byte[] v3 = v0_6.digest(v1_1);
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v3, MessageDigest.getInstance(pinValue.toByteArray()));
    return kotlin.collections.ArraysKt.joinToString$default(v3, ((CharSequence) ), 0, 0, 0, 0, ((kotlin.jvm.functions.Function1)
owasp.sat.agoat.AccessControlIssue1Activity$hashPIN$md$1.INSTANCE), 30, 0);
}

```

at `owasp.sat.agoat.AccessControlIssue1Activity.isPinCorrect`

```

public final boolean isPinCorrect(String p4)
{
    kotlin.jvm.internal.Intrinsics.checkNotNull(p4, pinValue);
    return this.getSharedPreferences(pinDetails, 0).getString(pin, 0).equals(this.hashPIN(p4));
}

```

at `owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick`

```

public final void onClick(android.view.View p5)
{
    android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165305));
    android.widget.Toast v1_10 = this.this$0;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v0_2, pinValue);
    if (!v1_10.isPinCorrect(v0_2.getText().toString())) {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Incorrect PIN entered), 1).show();
    } else {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN Verified), 1).show();
        this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0),
owasp.sat.agoat.AccessControl1ViewActivity));
    }
    return;
}

```

at `owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView`

```

public final void verifyPINView()
{
    this.setContentView(2131296285);
    ((android.widget.Button) this.findViewById(2131165371)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1(this)));
    return;
}

```

at `owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.onClick`

```

public final void onClick(android.view.View p5)
{
    android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165335));
    android.widget.Toast v1_9 = this.this$0;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v0_2, pinValue);
    if (!v1_9.createPIN(v0_2.getText().toString())) {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN not created), 1).show();
    } else {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN created. Please Login), 1).show();
        this.this$0.verifyPINView();
    }
    return;
}

```

at `owasp.sat.agoat.AccessControlIssue1Activity.onCreate`

```

protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    if (!this.verifyIfPinSet()) {
        this.setContentView(2131296284);
        ((android.widget.Button) this.findViewById(2131165334)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1(this)));
    } else {
        this.verifyPINView();
    }
    return;
}

```

Résultat #3

L'application utilise MD5. MD2, MD4 et MD5 sont des fonctions de hachage compromises par des collisions. Trace d'appel :

```

at java.security.MessageDigest.getInstance
at owasp.sat.agoat.AccessControlIssue1Activity.hashPIN
at owasp.sat.agoat.AccessControlIssue1Activity.isPinCorrect
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick
at owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView
at owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.onClick

```

at `java.security.MessageDigest.getInstance`

Unable to extract source code.

at `owasp.sat.agoat.AccessControlIssue1Activity.hashPIN`

```

public final String hashPIN(String p13)
{
    kotlin.jvm.internal.Intrinsics.checkNotNullParameter(p13, pinValue);
    String v0_6 = java.security.MessageDigest.getInstance(MD5);
    byte[] v1_1 = p13.getBytes(kotlin.text.Charsets.UTF_8);
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v1_1, (this as java.lang.String).getBytes(charset));
    byte[] v3 = v0_6.digest(v1_1);
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v3, MessageDigest.getInstance...(pinValue.toByteArray()));
    return kotlin.collections.ArraysKt.joinToString$default(v3, ((CharSequence) ), 0, 0, 0, 0, ((kotlin.jvm.functions.Function1)

```

```
owasp.sat.ag oat.AccessControlIssue1Activity$hashPIN$md$1.INSTANCE), 30, 0);
}
```

at `owasp.sat.ag oat.AccessControlIssue1Activity.isPinCorrect`

```
public final boolean isPinCorrect(String p4)
{
    kotlin.jvm.internal.Intrinsics.checkNotNull(p4, pinValue);
    return this.getSharedPreferences(pinDetails, 0).getString(pin, 0).equals(this.hashPIN(p4));
}
```

at `owasp.sat.ag oat.AccessControlIssue1Activity$verifyPINView$1.onClick`

```
public final void onClick(android.view.View p5)
{
    android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165305));
    android.widget.Toast v1_10 = this.this$0;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v0_2, pinValue);
    if (!v1_10.isPinCorrect(v0_2.getText().toString())) {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Incorrect PIN entered), 1).show();
    } else {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN Verified), 1).show();
        this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0),
owasp.sat.ag oat.AccessControl1ViewActivity));
    }
    return;
}
```

at `owasp.sat.ag oat.AccessControlIssue1Activity.verifyPINView`

```
public final void verifyPINView()
{
    this.setContentView(2131296285);
    ((android.widget.Button) this.findViewById(2131165371)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.ag oat.AccessControlIssue1Activity$verifyPINView$1(this)));
    return;
}
```

at `owasp.sat.ag oat.AccessControlIssue1Activity$onCreate$1.onClick`

```
public final void onClick(android.view.View p5)
{
    android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165335));
    android.widget.Toast v1_9 = this.this$0;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v0_2, pinValue);
    if (!v1_9.createPIN(v0_2.getText().toString())) {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN not created), 1).show();
    } else {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN created. Please Login), 1).show();
        this.this$0.verifyPINView();
    }
    return;
}
```

Résultat #4

L'application utilise MD5. MD2, MD4 et MD5 sont des fonctions de hachage qui ont été cassées par des collisions.

Trace d'appel:

```
at java.security.MessageDigest.getInstance
at owasp.sat.ag oat.AccessControlIssue1Activity.hashPIN
at owasp.sat.ag oat.AccessControlIssue1Activity.isPinCorrect
at owasp.sat.ag oat.AccessControlIssue1Activity$verifyPINView$1.onClick
at owasp.sat.ag oat.AccessControlIssue1Activity.verifyPINView
```

at `java.security.MessageDigest.getInstance`

Unable to extract source code.

at `owasp.sat.agoat.AccessControlIssue1Activity.hashPIN`

```
public final String hashPIN(String p13)
{
    kotlin.jvm.internal.Intrinsics.checkNotNull(p13, pinValue);
    String v0_6 = java.security.MessageDigest.getInstance(MD5);
    byte[] v1_1 = p13.getBytes(kotlin.text.Charsets.UTF_8);
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v1_1, (this as java.lang.String).getBytes(charset));
    byte[] v3 = v0_6.digest(v1_1);
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v3, MessageDigest.getInstance...t(pinValue.toByteArray()));
    return kotlin.collections.ArraysKt.joinToString$default(v3, ((CharSequence) ), 0, 0, 0, 0, ((kotlin.jvm.functions.Function1)
owasp.sat.agoat.AccessControlIssue1Activity$hashPIN$md$1.INSTANCE), 30, 0);
}
```

at `owasp.sat.agoat.AccessControlIssue1Activity.isPinCorrect`

```
public final boolean isPinCorrect(String p4)
{
    kotlin.jvm.internal.Intrinsics.checkNotNull(p4, pinValue);
    return this.getSharedPreferences(pinDetails, 0).getString(pin, 0).equals(this.hashPIN(p4));
}
```

at `owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick`

```
public final void onClick(android.view.View p5)
{
    android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165305));
    android.widget.Toast v1_10 = this.this$0;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v0_2, pinValue);
    if (!v1_10.isPinCorrect(v0_2.getText().toString())) {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Incorrect PIN entered), 1).show();
    } else {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN Verified), 1).show();
        this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0),
owasp.sat.agoat.AccessControl1ViewActivity));
    }
    return;
}
```

at `owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView`

```
public final void verifyPINView()
{
    this.setContentView(2131296285);
    ((android.widget.Button) this.findViewById(2131165371)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1(this)));
    return;
}
```

Résultat #5

L'application utilise MD5. MD2, MD4 et MD5 sont des fonctions de hachage compromises par collision. Trace d'appel:

```
at java.security.MessageDigest.getInstance
at owasp.sat.agoat.AccessControlIssue1Activity.hashPIN
at owasp.sat.agoat.AccessControlIssue1Activity.isPinCorrect
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick
```

à `java.security.MessageDigest.getInstance`

Unable to extract source code.

à `owasp.sat.agoat.AccessControlIssue1Activity.hashPIN`

```
public final String hashPIN(String p13)
{
    kotlin.jvm.internal.Intrinsics.checkNotNull(p13, pinValue);
    String v0_6 = java.security.MessageDigest.getInstance(MD5);
    byte[] v1_1 = p13.getBytes(kotlin.text.Charsets.UTF_8);
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_1, (this as java.lang.String).getBytes(charset));
    byte[] v3 = v0_6.digest(v1_1);
    kotlin.jvm.internal.Intrinsics.checkNotNull(v3, MessageDigest.getInstance(pinValue.toByteArray()));
    return kotlin.collections.ArraysKt.joinToString$default(v3, ((CharSequence) ), 0, 0, 0, 0, ((kotlin.jvm.functions.Function1)
owasp.sat.agoat.AccessControlIssue1Activity$hashPIN$md$1.INSTANCE), 30, 0);
}
```

à `owasp.sat.agoat.AccessControlIssue1Activity.isPinCorrect`

```
public final boolean isPinCorrect(String p4)
{
    kotlin.jvm.internal.Intrinsics.checkNotNull(p4, pinValue);
    return this.getSharedPreferences(pinDetails, 0).getString(pin, 0).equals(this.hashPIN(p4));
}
```

à `owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick`

```
public final void onClick(android.view.View p5)
{
    android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165305));
    android.widget.Toast v1_10 = this.this$0;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v0_2, pinValue);
    if (!v1_10.isPinCorrect(v0_2.getText().toString())) {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Incorrect PIN entered), 1).show();
    } else {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN Verified), 1).show();
        this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0),
owasp.sat.agoat.AccessControl1ViewActivity));
    }
    return;
}
```

Résultat #6

L'application utilise MD5. MD2, MD4 et MD5 sont des fonctions de hachage qui ont été compromises par des collisions. Trace d'appel :

```
at java.security.MessageDigest.getInstance
at owasp.sat.agoat.AccessControlIssue1Activity.hashPIN
at owasp.sat.agoat.AccessControlIssue1Activity.isPinCorrect
```

at `java.security.MessageDigest.getInstance`

Unable to extract source code.

at `owasp.sat.agoat.AccessControlIssue1Activity.hashPIN`

```
public final String hashPIN(String p13)
{
    kotlin.jvm.internal.Intrinsics.checkNotNull(p13, pinValue);
    String v0_6 = java.security.MessageDigest.getInstance(MD5);
    byte[] v1_1 = p13.getBytes(kotlin.text.Charsets.UTF_8);
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_1, (this as java.lang.String).getBytes(charset));
    byte[] v3 = v0_6.digest(v1_1);
    kotlin.jvm.internal.Intrinsics.checkNotNull(v3, MessageDigest.getInstance(pinValue.toByteArray()));
}
```

```
return kotlin.collections.ArraysKt.joinToString$default(v3, ((CharSequence) ), 0, 0, 0, 0, ((kotlin.jvm.functions.Function1)
owasp.sat.agoat.AccessControlIssue1Activity$hashPIN$md$1.INSTANCE), 30, 0);
}
```

à `owasp.sat.agoat.AccessControlIssue1Activity.isPinCorrect`

```
public final boolean isPinCorrect(String p4)
{
    kotlin.jvm.internal.Intrinsics.checkNotNull(p4, pinValue);
    return this.getSharedPreferences(pinDetails, 0).getString(pin, 0).equals(this.hashPIN(p4));
}
```

Résultat #7

L'application utilise MD5, MD2, MD4 et MD5 sont des fonctions de hachage compromises par des collisions. Trace d'appel :

```
at java.security.MessageDigest.getInstance
at owasp.sat.agoat.AccessControlIssue1Activity.hashPIN
```

à `java.security.MessageDigest.getInstance`

Unable to extract source code.

à `owasp.sat.agoat.AccessControlIssue1Activity.hashPIN`

```
public final String hashPIN(String p13)
{
    kotlin.jvm.internal.Intrinsics.checkNotNull(p13, pinValue);
    String v0_6 = java.security.MessageDigest.getInstance(MD5);
    byte[] v1_1 = p13.getBytes(kotlin.text.Charsets.UTF_8);
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v1_1, (this as java.lang.String).getBytes(charset));
    byte[] v3 = v0_6.digest(v1_1);
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v3, MessageDigest.getInstance...t(pinValue.toByteArray()));
    return kotlin.collections.ArraysKt.joinToString$default(v3, ((CharSequence) ), 0, 0, 0, 0, ((kotlin.jvm.functions.Function1)
owasp.sat.agoat.AccessControlIssue1Activity$hashPIN$md$1.INSTANCE), 30, 0);
}
```

Recommandation

Pour atténuer les risques liés à l'utilisation d'algorithmes de hachage faibles, tenez compte des points suivants :

- Évitez MD5 et SHA-1 : ces algorithmes sont non sécurisés et vulnérables aux collisions. Ils ne doivent être utilisés dans aucun cas, y compris pour le HMAC.
- Pour le hachage et la vérification de signatures, privilégiez SHA-256 ou SHA-3. Ils offrent un niveau de sécurité élevé et constituent des standards industriels largement reconnus.
- Pour le hachage de données sensibles (PII, PHI...), envisagez Argon2 ou Bcrypt. Ils sont conçus pour résister aux attaques par force brute et sont recommandés par les standards de sécurité.
- Lors de l'utilisation du HMAC pour l'authentification des messages, assurez-vous de le coupler avec des algorithmes de hachage sécurisés tels que SHA-256 ou SHA-3 afin de maintenir une sécurité robuste.

Références

- [Deprecating MD5 and SHA-1 signature hashes in \(D\)TLS 1.2](#)
- [Deprecation of SHA-1 \(NIST\)](#)

MOYEN

Requête HTTP en clair

CVSS v3: 9,1

L'application accède à des données via un canal non chiffré, compromettant la confidentialité et l'intégrité des données en transit.

Le trafic HTTP en clair désigne les données transmises entre une application mobile et son serveur backend sans chiffrement, ce qui les rend facilement lisibles et vulnérables à l'interception par des acteurs malveillants. Cette absence de chiffrement expose les applications mobiles à des risques de sécurité majeurs. Lorsque

des informations sensibles, telles que des identifiants de connexion, des données personnelles ou des détails financiers, transitent par des connexions HTTP en clair, elles deviennent vulnérables à l'écoute clandestine et aux attaques man-in-the-middle.

L'un des principaux risques liés au trafic HTTP en clair dans les applications mobiles est l'exposition potentielle des données utilisateur. Les attaquants peuvent intercepter et extraire des informations sensibles, ce qui peut entraîner un vol d'identité, un accès non autorisé aux comptes et un détournement de données personnelles. De plus, ils peuvent exploiter ces vulnérabilités pour altérer le contenu de l'application ou injecter du code malveillant dans le canal de communication.

Par ailleurs, le trafic HTTP en clair peut compromettre la vie privée des utilisateurs. Les applications mobiles collectent et transmettent souvent des données utilisateur à des fins d'analyse ou de publicité ciblée. Sans chiffrement, ces données sont facilement accessibles à des tiers, ce qui porte atteinte à la confidentialité des utilisateurs et peut entraîner des violations de la vie privée.

Les applications mobiles qui ne mettent pas en œuvre de protocoles de communication sécurisés encourrent également le risque d'attaques par usurpation d'identité. Les attaquants peuvent créer de faux réseaux Wi-Fi ou utiliser d'autres techniques pour intercepter le trafic, en se faisant passer pour le serveur backend légitime. Les utilisateurs transmettent alors leurs données à l'attaquant à leur insu, et ce dernier peut les exploiter à des fins malveillantes.

DART

```
import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

void main() => runApp(LoginApp());

class LoginApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Login App',
      home: LoginPage(),
    );
  }
}

class LoginPage extends StatefulWidget {
  @override
  _LoginPageState createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  final TextEditingController _usernameController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  String _loginStatus = '';

  void _performLogin() async {
    final String username = _usernameController.text.trim();
    final String password = _passwordController.text.trim();

    // Simulate an HTTP request for login.
    final String apiUrl = 'http://example.com/login';
    final Map<String, dynamic> requestBody = {
      'username': username,
      'password': password,
    };

    final http.Response response = await http.post(
      Uri.parse(apiUrl),
      headers: {'Content-Type': 'application/json'},
      body: json.encode(requestBody),
    );

    if (response.statusCode == 200) {
      // Successful login
      setState(() {
        _loginStatus = 'Login successful!';
      });
    } else {
      // Failed login
      setState(() {
        _loginStatus = 'Login failed. Please try again.';
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Login Page'),
      ),
    );
  }
}
```

```

    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          TextFormField(
            controller: _usernameController,
            decoration: InputDecoration(labelText: 'Username'),
          ),
          TextFormField(
            controller: _passwordController,
            obscureText: true,
            decoration: InputDecoration(labelText: 'Password'),
          ),
          SizedBox(height: 20),
          ElevatedButton(
            onPressed: _performLogin,
            child: Text('Login'),
          ),
          SizedBox(height: 10),
          Text(_loginStatus, textAlign: TextAlign.center),
        ],
      ),
    ),
  );
}
}

```

SWIFT

```

import UIKit

class LoginViewController: UIViewController {
  @IBOutlet weak var usernameTextField: UITextField!
  @IBOutlet weak var passwordTextField: UITextField!
  @IBOutlet weak var loginStatusLabel: UILabel!

  override func viewDidLoad() {
    super.viewDidLoad()
  }

  @IBAction func loginButtonTapped(_ sender: UIButton) {
    guard let username = usernameTextField.text?.trimmingCharacters(in: .whitespacesAndNewlines),
          let password = passwordTextField.text?.trimmingCharacters(in: .whitespacesAndNewlines) else {
      return
    }

    // Simulate an HTTP request for login.
    let apiUrl = "http://example.com/login"
    let requestBody: [String: Any] = [
      "username": username,
      "password": password
    ]

    var request = URLRequest(url: URL(string: apiUrl)!)
    request.httpMethod = "POST"
    request.setValue("application/json", forHTTPHeaderField: "Content-Type")

    do {
      request.httpBody = try JSONSerialization.data(withJSONObject: requestBody, options: [])
    } catch {
      print("Error creating JSON data: \(error)")
      return
    }

    let task = URLSession.shared.dataTask(with: request) { data, response, error in
      if let error = error {
        print("Error: \(error)")
        DispatchQueue.main.async {
          self.loginStatusLabel.text = "Login failed. Please try again."
        }
        return
      }

      guard let data = data,
            let httpResponse = response as? HTTPURLResponse,

```

```

        httpResponse.statusCode == 200 else {
            DispatchQueue.main.async {
                self.loginStatusLabel.text = "Login failed. Please try again."
            }
            return
        }

        DispatchQueue.main.async {
            self.loginStatusLabel.text = "Login successful!"
        }
    }

    task.resume()
}
}

```

KOTLIN

```

import android.os.Bundle
import android.util.Log
import androidx.appcompat.app.AppCompatActivity
import kotlinx.android.synthetic.main.activity_login.*
import org.json.JSONObject
import java.io.OutputStreamWriter
import java.net.HttpURLConnection
import java.net.URL

class LoginActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)

        loginButton.setOnClickListener {
            val username = usernameEditText.text.trim().toString()
            val password = passwordEditText.text.trim().toString()

            // Simulate an HTTP request for login.
            val apiUrl = "https://example.com/login"
            val requestBody = JSONObject().apply {
                put("username", username)
                put("password", password)
            }.toString()

            Thread {
                performLogin(apiUrl, requestBody)
            }.start()
        }
    }

    private fun performLogin(apiUrl: String, requestBody: String) {
        try {
            val url = URL(apiUrl)
            val connection = url.openConnection() as HttpURLConnection
            connection.requestMethod = "POST"
            connection.setRequestProperty("Content-Type", "application/json")
            connection.doOutput = true

            val outputStreamWriter = OutputStreamWriter(connection.outputStream)
            outputStreamWriter.write(requestBody)
            outputStreamWriter.flush()

            val responseCode = connection.responseCode
            if (responseCode == HttpURLConnection.HTTP_OK) {
                runOnUiThread {
                    loginStatusLabel.text = "Login successful!"
                }
            } else {
                runOnUiThread {
                    loginStatusLabel.text = "Login failed. Please try again."
                }
            }
        } catch (e: Exception) {
            Log.e("LoginActivity", "Error: ${e.message}")
            runOnUiThread {
                loginStatusLabel.text = "Login failed. Please try again."
            }
        }
    }
}

```

```
}  
}
```

Détails techniques

Résultat #1

Utilisation d'une URL HTTP en clair et non chiffrée : `http://demo.testfire.net/` Requête :

```
GET / HTTP/1.1  
Host: demo.testfire.net  
Connection: Keep-Alive  
Accept-Encoding: gzip  
User-Agent: okhttp/3.8.0
```

Réponse :

```
HTTP/1.1 200 OK  
Server: Apache-Coyote/1.1  
Set-Cookie: JSESSIONID=9BDCDE139DFF2F56BD4E46A2E78C713E; Path=/; HttpOnly  
Content-Type: text/html; charset=ISO-8859-1  
Transfer-Encoding: chunked  
Date: Tue, 19 May 2026 16:52:06 GMT
```

Recommandation

Pour atténuer les risques liés au trafic HTTP en clair dans les applications mobiles, il est impératif de mettre en œuvre des protocoles de communication sécurisés et des bonnes pratiques. Vous trouverez ci-dessous des recommandations détaillées pour renforcer la sécurité des communications de l'application mobile :

- Adopter le chiffrement HTTPS : L'étape fondamentale consiste à utiliser HTTPS (HTTP Secure) pour l'ensemble des communications entre l'application mobile et le serveur backend. HTTPS chiffre les données en transit, garantissant la confidentialité et la sécurité des informations sensibles. Obtenez un certificat SSL/TLS auprès d'une autorité de certification (CA) de confiance et forcez l'utilisation de HTTPS sur tous les points de terminaison API et transferts de données.
- Certificate Pinning : Implémentez le certificate pinning pour renforcer la sécurité des connexions HTTPS. Cela consiste à intégrer en dur le certificat SSL du serveur ou sa clé publique au sein de l'application mobile. Ainsi, l'application garantit qu'elle ne communique qu'avec le serveur légitime, prévenant les attaques man-in-the-middle où des attaquants tentent d'utiliser leurs propres certificats.
- Activer HSTS (HTTP Strict Transport Security) : Utilisez les en-têtes HSTS pour forcer le navigateur web de l'application à toujours utiliser des connexions HTTPS, même si l'utilisateur saisit par erreur "http://" dans l'URL. Cela prévient toute attaque downgrade potentielle et garantit que l'ensemble des communications sont chiffrées.
- Utiliser la dernière version de TLS : Assurez-vous que l'application mobile utilise la version la plus récente du protocole TLS (Transport Layer Security) afin de bénéficier des fonctionnalités de sécurité et des algorithmes de chiffrement les plus robustes.
- Implémenter la vérification des certificats : Configurez l'application pour qu'elle vérifie la révocation ou l'expiration des certificats SSL, afin d'éviter toute interaction avec des serveurs potentiellement compromis.
- Chiffrement des données sensibles : Chiffrez les données sensibles stockées sur l'appareil ou transmises au serveur. Utilisez des algorithmes de chiffrement robustes et gérez les clés de chiffrement de manière sécurisée ; cela constitue une couche de sécurité supplémentaire pour la transmission des données.
- Sécuriser la transmission des données pour l'ensemble des API : Évaluez toutes les API utilisées par l'application et assurez-vous qu'elles utilisent également HTTPS pour la transmission des données. Cela inclut les API tierces et les services intégrés à l'application.

DART

```
import 'dart:convert';  
import 'package:flutter/material.dart';  
import 'package:http/http.dart' as http;  
  
void main() => runApp(LoginApp());  
  
class LoginApp extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      title: 'Login App',  
      home: LoginPage(),  
    );  
  }  
}
```

```

    }
}

class LoginPage extends StatefulWidget {
  @override
  _LoginPageState createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  final TextEditingController _usernameController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  String _loginStatus = '';

  void _performLogin() async {
    final String username = _usernameController.text.trim();
    final String password = _passwordController.text.trim();

    // Simulate a secure HTTPS request for login (Replace this with your actual secure API endpoint).
    final String apiUrl = 'https://www.example.com/login';
    final Map<String, dynamic> requestBody = {
      'username': username,
      'password': password,
    };

    final http.Response response = await http.post(
      Uri.parse(apiUrl),
      headers: {'Content-Type': 'application/json'},
      body: json.encode(requestBody),
    );

    if (response.statusCode == 200) {
      // Successful login
      setState(() {
        _loginStatus = 'Login successful!';
      });
    } else {
      // Failed login
      setState(() {
        _loginStatus = 'Login failed. Please try again.';
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Login Page'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            TextFormField(
              controller: _usernameController,
              decoration: InputDecoration(labelText: 'Username'),
            ),
            TextFormField(
              controller: _passwordController,
              obscureText: true,
              decoration: InputDecoration(labelText: 'Password'),
            ),
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: _performLogin,
              child: Text('Login'),
            ),
            SizedBox(height: 10),
            Text(_loginStatus, textAlign: TextAlign.center),
          ],
        ),
      ),
    );
  }
}

```

```

import UIKit

class LoginViewController: UIViewController {
    @IBOutlet weak var usernameTextField: UITextField!
    @IBOutlet weak var passwordTextField: UITextField!
    @IBOutlet weak var loginStatusLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    @IBAction func loginButtonTapped(_ sender: UIButton) {
        guard let username = usernameTextField.text?.trimmingCharacters(in: .whitespacesAndNewlines),
              let password = passwordTextField.text?.trimmingCharacters(in: .whitespacesAndNewlines) else {
            return
        }

        // Simulate a secure HTTPS request for login (Replace this with your actual secure API endpoint).
        let apiUrl = "https://www.example.com/login"
        let requestBody: [String: Any] = [
            "username": username,
            "password": password
        ]

        var request = URLRequest(url: URL(string: apiUrl)!)
        request.httpMethod = "POST"
        request.setValue("application/json", forHTTPHeaderField: "Content-Type")

        do {
            request.httpBody = try JSONSerialization.data(withJSONObject: requestBody, options: [])
        } catch {
            print("Error creating JSON data: \(error)")
            return
        }

        let task = URLSession.shared.dataTask(with: request) { data, response, error in
            if let error = error {
                print("Error: \(error)")
                DispatchQueue.main.async {
                    self.loginStatusLabel.text = "Login failed. Please try again."
                }
                return
            }

            guard let data = data,
                  let httpResponse = response as? HTTPURLResponse,
                  httpResponse.statusCode == 200 else {
                DispatchQueue.main.async {
                    self.loginStatusLabel.text = "Login failed. Please try again."
                }
                return
            }

            DispatchQueue.main.async {
                self.loginStatusLabel.text = "Login successful!"
            }
        }

        task.resume()
    }
}

```

KOTLIN

```

import android.os.Bundle
import android.util.Log
import androidx.appcompat.app.AppCompatActivity
import kotlinx.android.synthetic.main.activity_login.*
import org.json.JSONObject
import java.io.OutputStreamWriter
import java.net.HttpURLConnection
import java.net.URL

class LoginActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)
    }
}

```

```

loginButton.setOnClickListener {
    val username = usernameEditText.text.trim().toString()
    val password = passwordEditText.text.trim().toString()

    // Simulate a secure HTTPS request for login (Replace this with your actual secure API endpoint).
    val apiUrl = "https://www.example.com/login"
    val requestBody = JSONObject().apply {
        put("username", username)
        put("password", password)
    }.toString()

    Thread {
        performLogin(apiUrl, requestBody)
    }.start()
}

private fun performLogin(apiUrl: String, requestBody: String) {
    try {
        val url = URL(apiUrl)
        val connection = url.openConnection() as HttpURLConnection
        connection.requestMethod = "POST"
        connection.setRequestProperty("Content-Type", "application/json")
        connection.doOutput = true

        val outputStreamWriter = OutputStreamWriter(connection.outputStream)
        outputStreamWriter.write(requestBody)
        outputStreamWriter.flush()

        val responseCode = connection.responseCode
        if (responseCode == HttpURLConnection.HTTP_OK) {
            runOnUiThread {
                loginStatusLabel.text = "Login successful!"
            }
        } else {
            runOnUiThread {
                loginStatusLabel.text = "Login failed. Please try again."
            }
        }
    } catch (e: Exception) {
        Log.e("LoginActivity", "Error: ${e.message}")
        runOnUiThread {
            loginStatusLabel.text = "Login failed. Please try again."
        }
    }
}
}

```

Détails du vecteur CVSS

Vecteur CVSS V3 : CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

Vecteur d'Attaque:	Réseau (N) Adjacent (A) Local (L) Physique (P)
Complexité d'Attaque:	Faible (L) Élevé (H)
Privilèges Requis:	Aucun(e) (N) Faible (L) Élevé (H)
Interaction Utilisateur:	Aucun(e) (N) Requis (R)
Périmètre:	Modifié (C) Non modifié (U)
Confidentialité:	Élevé (H) Faible (L) Aucun(e) (N)
Intégrité:	Élevé (H) Faible (L) Aucun(e) (N)
Disponibilité:	Élevé (H) Faible (L) Aucun(e) (N)

Références

- [Missing Encryption of Sensitive Data \(CWE-311\)](#)
- [Testing for Weak SSL/TLS Ciphers, Insufficient Transport Layer Protection](#)
- [Top 10 2013-A6-Sensitive Data Exposure](#)
- [Insufficient Transport Layer](#)

- [Cleartext Transmission of Sensitive Information \(CWE-319\)](#)

MOYEN

CERTIFICAT_EXPIRÉ

Un certificat SSL/TLS expiré n'est plus approuvé par les navigateurs et les clients, entraînant des alertes de sécurité, des interruptions de service et des vulnérabilités de sécurité potentielles, notamment des attaques man-in-the-middle.

Un certificat SSL/TLS expiré n'est plus considéré comme fiable par les navigateurs et clients modernes, ce qui rompt les chaînes de confiance, génère des alertes de sécurité et expose à plusieurs vulnérabilités potentielles, notamment les attaques de l'homme du milieu (MitM).

Lorsqu'un certificat expire, les navigateurs affichent des alertes de sécurité bien visibles telles que « Votre connexion n'est pas privée » ou « Certificat expiré ». Ces alertes interrompent les communications sécurisées entre le client et le serveur, affectant directement la confiance des utilisateurs et la continuité d'activité. Dans l'e-commerce et les autres environnements sensibles, cela entraîne souvent l'abandon des utilisateurs, des pertes financières et un préjudice réputationnel.

Les risques de sécurité liés à un certificat expiré vont au-delà de l'impact sur les utilisateurs. Les attaquants peuvent exploiter l'absence de certificat valide pour mener des attaques MitM et intercepter ou altérer des données sensibles en transit. Sans certificat valide pour authentifier le serveur, les communications chiffrées perdent leur protection, rendant les données vulnérables à l'interception, à la modification ou à la falsification.

Par exemple, un site web doté d'un certificat expiré ne peut plus garantir la confidentialité et l'intégrité des données utilisateur. Un attaquant peut usurper l'identité du site et intercepter des identifiants de connexion ou des informations de paiement en falsifiant la connexion, ce qui conduit à un accès non autorisé ou à une fraude financière.

Les certificats expirés constituent également une non-conformité à plusieurs normes de sécurité, notamment PCI DSS, HIPAA et RGPD. Ces réglementations imposent l'utilisation de certificats SSL/TLS valides pour protéger les données sensibles. Leur non-respect peut entraîner des sanctions légales et des atteintes supplémentaires à la réputation.

Détails techniques

Résultat #1

CERTIFICATE_EXPIRED détecté sur `demo.testfire.net`

- **Constat** : expire dans < 60 jours (33)

Recommandation

Pour remédier aux risques liés à l'expiration des certificats SSL/TLS, les entreprises devraient mettre en œuvre plusieurs stratégies proactives :

- Outils de surveillance automatisée : déployez des outils qui surveillent en continu l'état des certificats SSL/TLS et envoient des alertes à l'approche de leur expiration. Cela permet d'éviter les interruptions de service dues à des certificats expirés passés inaperçus.
- Procédures de renouvellement d'urgence : établissez des procédures d'urgence claires pour le renouvellement rapide des certificats en cas d'expiration imprévue. Cela garantit une interruption minimale des services et protège contre les risques de sécurité potentiels.
- Audits réguliers : effectuez des audits réguliers de votre inventaire de certificats SSL/TLS pour identifier et renouveler rapidement tout certificat expiré. Maintenir un inventaire à jour permet d'éviter les lacunes de sécurité.
- Provisionnement automatisé des certificats : implémentez des systèmes qui automatisent la délivrance et le renouvellement des certificats SSL/TLS pour en assurer la validité continue. Des outils tels que cert-manager dans Kubernetes simplifient ce processus.

Provisionnement automatisé des certificats dans Kubernetes :

Dans Kubernetes, vous pouvez automatiser la gestion des certificats SSL/TLS à l'aide de cert-manager. Cet outil interagit avec des autorités de certification (CA) telles que Let's Encrypt pour délivrer et renouveler automatiquement les certificats.

- Installation de cert-manager :

```
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/latest/download/cert-manager.yaml
```

Voici une configuration YAML pour cert-manager afin d'automatiser le provisionnement des certificats :

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: example-cert
  namespace: default
spec:
  secretName: example-cert-secret
  issuerRef:
    name: letsencrypt-prod
    kind: ClusterIssuer
  commonName: example.com
  dnsNames:
```

```
- example.com
- www.example.com
duration: 90d
renewBefore: 30d
```

Surveillance avec Certbot :

Pour les environnements qui n'utilisent pas Kubernetes, vous pouvez utiliser Certbot pour automatiser la délivrance et le renouvellement des certificats SSL.

```
# Automatically issue or renew SSL certificates using Certbot
domain="example.com"
email="admin@example.com"

# Run Certbot in standalone mode to obtain the certificate
certbot certonly --standalone -d $domain --email $email --agree-tos
```

Renouvellement automatisé des certificats via une tâche cron :

Vous pouvez automatiser le processus de renouvellement à l'aide d'une tâche cron :

```
# Edit your crontab with: crontab -e
0 0 * * * /usr/bin/certbot renew --quiet
```

Cette tâche cron s'exécute quotidiennement à minuit pour vérifier si des certificats doivent être renouvelés, assurant ainsi une couverture continue sans intervention manuelle.

Références

- [NIST Guidelines for Key Management](#)
- [SSL Labs Best Practices](#)

MOYEN

Configuration non sécurisée des en-têtes HTTP

La configuration des en-têtes de l'application est manquante ou définit des valeurs d'en-tête non sécurisées ou dangereuses.

Configuration d'en-tête non sécurisée :

- **Content Security Policy** : Réduit les risques de cross-site scripting (XSS) en spécifiant les sources de contenu de confiance.
- **Cookie** : Renforce la sécurité en définissant des attributs comme HttpOnly et Secure pour protéger les données des cookies contre les accès non autorisés.
- **Cross-Origin Resource Sharing** : Contrôle la manière dont les ressources peuvent être partagées entre différents domaines pour prévenir les accès malveillants.
- **HTTP Public Key Pinning** : Protège contre les attaques man-in-the-middle en spécifiant les clés publiques valides pour un site donné.
- **Redirection** : Garantit que les redirections sont sécurisées et ne mènent qu'à des destinations de confiance afin de prévenir les vulnérabilités de type open redirect.
- **Referrer Policy** : Définit la quantité d'informations referrer transmises lors de la navigation d'un site à un autre, renforçant ainsi la confidentialité.
- **Subresource Integrity** : Vérifie que les ressources chargées depuis des domaines tiers n'ont pas été altérées en contrôlant leur hachage cryptographique.
- **X-Content-Type-Options** : Empêche les navigateurs de procéder au MIME-sniffing, réduisant ainsi le risque d'attaques basées sur le type de contenu.
- **X-Frame-Options** : Préviens les attaques de type clickjacking en contrôlant si une page peut être intégrée dans un frame.
- **X-XSS-Protection** : Active le filtrage XSS intégré au navigateur pour bloquer les attaques de cross-site scripting détectées.
- **Permissions-Policy** : Contrôle les fonctionnalités et API pouvant être utilisées dans le navigateur, renforçant la sécurité en limitant les capacités du contenu non fiable.
- **Clear-Site-Data** : Permet aux sites de demander au navigateur d'effacer les données stockées (cookies, stockage local, cache) pour une origine spécifiée, contribuant à atténuer l'impact des fuites de données et des problèmes de confidentialité.

Détails techniques

Résultat #1

MISSING_SECURITY_HEADERS détecté sur `demo.testfire.net`

- **Constat** : --

Recommandation

Pour vous prémunir contre les configurations d'en-têtes non sécurisées, appliquez les recommandations suivantes :

1. Content Security Policy (CSP) :

Restreignez les sources de contenu afin de réduire les risques d'attaques cross-site scripting (XSS) et de chargement de ressources non autorisées.

```
Content-Security-Policy: default-src 'self'; script-src 'self' https://cdnjs.cloudflare.com;
```

2. En-têtes de sécurité des cookies :

Activez les flags Secure et HttpOnly pour prévenir le vol et la manipulation des cookies et renforcer la sécurité des sessions utilisateur.

```
Set-Cookie: sessionId=abc123; Secure; HttpOnly;
```

3. Cross-Origin Resource Sharing (CORS) :

Configurez correctement les politiques CORS afin de restreindre l'accès aux ressources depuis des origines tierces et de réduire les risques de cross-site request forgery (CSRF) et de fuite de données cross-origin.

```
Access-Control-Allow-Origin: https://example.com
```

4. HTTP Public Key Pinning (HPKP) :

Utilisez HPKP pour lier les clés publiques à des serveurs web spécifiques et vous protéger contre les attaques Man-in-the-Middle (MitM) par certificats frauduleux.

```
Public-Key-Pins: pin-sha256="base64=="; max-age=5184000; includeSubDomains;
```

5. En-têtes de redirection :

Garantissez la sécurité des redirections en appliquant des contrôles stricts pour prévenir les vulnérabilités d'open redirection et protéger les utilisateurs contre le phishing.

```
Location: https://example.com/secure-page
```

6. Referrer Policy :

Définissez une politique de référent appropriée pour contrôler la quantité d'informations transmise via l'en-tête Referer et réduire le risque d'exposition de données sensibles.

```
Referrer-Policy: strict-origin-when-cross-origin
```

7. Subresource Integrity (SRI) :

Implémentez SRI afin de vérifier l'intégrité des ressources externes, telles que les scripts et les feuilles de style, et de vous protéger contre les modifications non autorisées et les attaques sur la chaîne d'approvisionnement.

```
<script src="https://example.com/example.js" integrity="sha256-base64==" crossorigin="anonymous"></script>
```

8. X-Content-Type-Options :

Activez la directive 'nosniff' afin d'empêcher les navigateurs de procéder au MIME-sniffing d'une réponse et de réduire les risques d'attaques par confusion de type de contenu.

```
X-Content-Type-Options: nosniff
```

9. X-Frame-Options :

Configurez X-Frame-Options afin de restreindre l'intégration du contenu web dans des frames, protégeant ainsi contre les attaques clickjacking et assurant l'intégrité de vos pages web.

```
X-Frame-Options: DENY
```

10. X-XSS-Protection :

Activez les mécanismes de protection XSS pour atténuer ces attaques en demandant aux navigateurs de nettoyer ou bloquer les scripts potentiellement malveillants.

```
X-XSS-Protection: 1; mode=block
```

Références

- [OWASP HTTP Header Security](#)
- [Content Security Policy \(MDN\)](#)
- [X-Frame-Options \(MDN\)](#)

MOYEN Configuration d'en-tête HTTP non sécurisée : HTTP Strict Transport Security (HSTS)

Configuration non sécurisée d'en-tête HTTP : HTTP Strict Transport Security (HSTS).

HTTP Strict Transport Security (HSTS) est un mécanisme de politique de sécurité web par lequel un serveur web déclare que les user agents conformes (tels qu'un navigateur web) doivent interagir avec lui uniquement via des connexions sécurisées (HTTPS). Le serveur communique la politique HSTS à l'user agent via un champ d'en-tête de réponse HTTP nommé "Strict-Transport-Security". La politique HSTS spécifie une durée pendant laquelle l'user agent doit accéder au serveur exclusivement de manière sécurisée.

Lorsqu'une application web émet une politique HSTS aux user agents, les user agents conformes se comportent comme suit :

- Convertir automatiquement tout lien non sécurisé (HTTP) référençant l'application web en lien sécurisé (HTTPS). (Par exemple, <http://example.com/some/page/> sera modifié en <https://example.com/some/page/> avant d'accéder au serveur.)
- Si la sécurité de la connexion ne peut être garantie (p. ex., le certificat TLS du serveur est auto-signé), les user agents affichent un message d'erreur et empêchent l'utilisateur d'accéder à l'application web.

Détails techniques

Résultat #1

HSTS détecté sur `demo.testfire.net`

- **Constat** : non proposé

Recommandation

Configurez votre serveur web pour rediriger les requêtes HTTP vers HTTPS.

Par exemple, pour Apache, vous devez modifier le fichier httpd.conf. Pour plus de configurations, veuillez consulter la section Références externes.

```
# load module
LoadModule headers_module modules/mod_headers.so

# redirect all HTTP to HTTPS (optional)
<VirtualHost *:80>
    ServerAlias *
    RewriteEngine On
    RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [redirect=301]
</VirtualHost>

# HTTPS-Host-Configuration
<VirtualHost *:443>
    # Use HTTP Strict Transport Security to force client to use secure connections only
    Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"

    # Further Configuration goes here
    [...]
</VirtualHost>
```

Références

- [Strict-Transport-Security](#)

MOYEN Suites de chiffrement TLS non sécurisées prises en charge

Le client prend en charge des combinaisons de suites de chiffrement présentant des faiblesses cryptographiques connues.

Le protocole Transport Layer Security (TLS) débute lorsque le client demande une connexion sécurisée et présente une liste de suites cryptographiques prises en charge.

Une suite cryptographique regroupe des algorithmes d'authentification, de chiffrement, de code d'authentification de message (MAC) et d'échange de clés. Chaque algorithme joue un rôle spécifique lors de la négociation du protocole. Par exemple, le recours à un algorithme faible peut affecter de manière critique la sécurité de l'ensemble du canal de communication.

L'analyse a révélé que le point de terminaison prend en charge une combinaison de suites cryptographiques et de protocoles Secure Sockets Layer / Transport Layer Security (SSL/TLS) présentant des faiblesses cryptographiques connues. Par conséquent, ces éléments ne doivent pas être utilisés pour garantir une sécurité effective de la couche de transport. Un attaquant capable d'intercepter la connexion pourrait influencer ou déchiffrer le trafic en transit.

Des faiblesses cryptographiques ont été démontrées pour chacune des configurations suivantes :

- Prise en charge du protocole SSL obsolète (v2 ou v3)
- Prise en charge de suites cryptographiques proposant des clés inférieures à 128 bits
- Prise en charge de suites cryptographiques offrant le mode **CBC** en combinaison avec un protocole TLS de version inférieure à 1.1, vulnérable à l'attaque BEAST
- Utilisation de la compression au sein de TLS, vulnérable à l'attaque CRIME
- Prise en charge de suites cryptographiques proposant **RC4** comme algorithme de chiffrement
- Prise en charge de suites cryptographiques proposant **DES** comme algorithme de chiffrement
- Prise en charge de suites cryptographiques proposant **MD5** comme algorithme de signature

Détails techniques

Résultat #1

cipherlist_OBSOLETEd détectée sur **demo.testfire.net**

- **Constat** : Suites de chiffrement obsolètes proposées

Résultat #2

cipherlist_OBSOLETEd détecté sur **owasp.org**

- **Constat** : suites de chiffrement obsolètes proposées

Recommandation

La configuration TLS recommandée doit respecter les recommandations suivantes :

- Restreindre à **TLSv1** et versions ultérieures, **TLS1.2** étant préféré
- Si **SSLv3** est requis, il est conseillé d'implémenter la fonctionnalité **TLS Fallback SCSV** pour prévenir les attaques par rétrogradation de protocole
- Désactiver Diffie-Hellman anonyme (**ADH**)
- Désactiver les suites de chiffrement **aNULL** et **eNULL**
- Désactiver les suites d'échange de clés Export
- Supprimer la prise en charge de **RC4**
- Supprimer la prise en charge de **DES**
- Supprimer la prise en charge de **MD5**
- Privilégier **SHA256** à **SHA1**
- Privilégier **AES128** à **AES256** , car 256 offre peu d'avantages sur le plan sécuritaire et est moins résistant aux attaques de type timing
- Désactiver la renégociation initiée par le client
- Désactiver la compression TLS
- Ne proposer que des chiffrements dont la longueur de clé est supérieure à 128 bits
- Proposer des suites de chiffrement offrant la propriété de Perfect-Forward Secrecy (**DHE** , **ECDHE**)
- Utiliser un groupe Diffie-Hellman personnalisé pour se prémunir contre l'attaque Logjam
- Implémenter l'en-tête HTTP Strict Transport Security
- Implémenter **OSCP stapling**

Exemple de configuration TLS sécurisée pour Nginx générée avec le [Mozilla SSL Configuration Generator](#)

<> NGINX

```
server {
    listen 443 ssl;
```

```

# certs sent to the client in SERVER HELLO are concatenated in ssl_certificate
ssl_certificate /path/to/signed_cert_plus_intermediates;
ssl_certificate_key /path/to/private_key;
ssl_session_timeout 1d;
ssl_session_cache shared:SSL:50m;
ssl_session_tickets off;

# Diffie-Hellman parameter for DHE ciphersuites, recommended 2048 bits
ssl_dhparam /path/to/dhparam.pem;

# modern configuration. tweak to your needs.
ssl_protocols TLSv1.1 TLSv1.2;
ssl_ciphers 'ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!3DES:!MD5:!PSK';
ssl_prefer_server_ciphers on;

# HSTS (ngx_http_headers_module is required) (15768000 seconds = 6 months)
add_header Strict-Transport-Security max-age=15768000;

# OCSP Stapling ---
# fetch OCSP records from URL in ssl_certificate and cache them
ssl_stapling on;
ssl_stapling_verify on;

## verify chain of trust of OCSP response using Root CA and Intermediate certs
ssl_trusted_certificate /path/to/root_CA_cert_plus_intermediates;

resolver <IP DNS resolver>;

....
}

```

Références

- [Mozilla Secure TLS/SSL Configuration](#)
- [Transport Layer Protection Cheat Sheet \(OWASP\)](#)
- [SSL/TLS Deployment Best Practices \(SSLabs\)](#)

MOYEN

TLS_FALLBACK_SCSV non pris en charge

Le serveur ne prend pas en charge TLS_FALLBACK_SCSV, ce qui le rend vulnérable aux attaques de downgrade de protocole.

Cette vulnérabilité indique que le serveur ne prend pas en charge **TLS_FALLBACK_SCSV** (Signaling Cipher Suite Value), un mécanisme conçu pour prévenir les attaques de downgrade de protocole sur les connexions SSL/TLS.

TLS_FALLBACK_SCSV est essentiel car :

1. Il permet aux clients d'indiquer lorsqu'ils rétrogradent vers une version de protocole inférieure en raison d'échecs de connexion.
2. Il permet aux serveurs de détecter et d'empêcher les tentatives de downgrade malveillantes.
3. Il contribue à maintenir le niveau de sécurité le plus élevé possible sur les connexions SSL/TLS.

Lorsque **TLS_FALLBACK_SCSV** n'est pas pris en charge, le serveur devient vulnérable à :

1. **Attaques de downgrade de protocole** : un attaquant peut forcer l'utilisation de versions de protocole plus anciennes et moins sécurisées.
2. **Attaques Man-in-the-Middle (MitM)** : la rétrogradation vers un protocole plus faible peut faciliter l'interception et le déchiffrement des communications par un attaquant.
3. **Exploitation de vulnérabilités dans les anciens protocoles** : les anciennes versions SSL/TLS peuvent comporter des vulnérabilités connues exploitables une fois le downgrade effectué.

Exemple de scénario :

Un attaquant intercepte le handshake initial entre un client et un serveur. Il manipule la connexion pour la faire échouer, amenant le client à réessayer avec une version de protocole inférieure. Sans **TLS_FALLBACK_SCSV**, le serveur accepte cette rétrogradation, exposant potentiellement la connexion aux vulnérabilités de l'ancienne version du protocole.

L'absence de prise en charge de **TLS_FALLBACK_SCSV** viole les bonnes pratiques de sécurité SSL/TLS et peut affecter la conformité avec diverses normes et réglementations de sécurité.

Détails techniques

fallback_SCSV détecté sur `demo.testfire.net`

- **Constat** : NON pris en charge

Recommandation

Lorsque TLS_FALLBACK_SCSV (Signaling Cipher Suite Value) n'est pas pris en charge, des attaques par downgrade peuvent survenir. Voici des stratégies pour atténuer ce risque :

Stratégies proactives

1. **Désactiver le fallback de version SSL/TLS** : Configurez les serveurs pour désactiver le retour aux anciennes versions SSL/TLS. Cela empêche les attaquants de forcer l'utilisation de protocoles vulnérables.
2. **Imposer une version TLS minimale** : Définissez une version TLS minimale acceptable (par exemple, TLS 1.2) côté client et serveur pour empêcher le downgrade vers des versions vulnérables.
3. **Scans de sécurité réguliers** : Effectuez des scans périodiques de votre infrastructure pour identifier et corriger les faiblesses de configuration TLS.
4. **Surveiller les comportements de négociation TLS anormaux** : Mettez en place une surveillance pour détecter les tentatives de downgrade de protocole.

Exemples de mise en œuvre

1. Configuration Nginx

Pour désactiver le fallback de version SSL/TLS et imposer une version minimale dans Nginx :

```
server {
    listen 443 ssl;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;
    ssl_ciphers 'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256';
    # Other SSL settings...
}
```

2. Configuration Apache

Pour Apache, utilisez la configuration suivante pour obtenir un résultat similaire :

```
<VirtualHost *:443>
    SSLEngine on
    SSLProtocol -all +TLSv1.2 +TLSv1.3
    SSLCipherSuite ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256
    SSLHonorCipherOrder on
    # Other SSL settings...
</VirtualHost>
```

3. Commande OpenSSL pour les tests

Utilisez OpenSSL pour tester la configuration TLS de votre serveur :

```
openssl s_client -connect example.com:443 -tls1_2
```

Cette commande tente d'établir une connexion en TLS 1.2. Si elle réussit, cela indique que votre serveur est correctement configuré pour utiliser des versions TLS modernes.

Surveillance des tentatives de downgrade

- Utilisation de fail2ban

Vous pouvez utiliser `fail2ban` pour surveiller les logs et détecter les tentatives de downgrade potentielles :

1. Créez un filtre personnalisé dans `/etc/fail2ban/filter.d/tls-downgrade.conf` :

```
[Definition]
failregex = SSL routines:SSL23_GET_CLIENT_HELLO:unsupported protocol.*client=<HOST>
```

2. Ajoutez une jail dans `/etc/fail2ban/jail.local` :

```
[tls-downgrade]
enabled = true
filter = tls-downgrade
logpath = /var/log/nginx/error.log
maxretry = 3
bantime = 3600
```

3. Redémarrez fail2ban :

```
sudo systemctl restart fail2ban
```

Cette configuration bannira les adresses IP qui effectuent des tentatives répétées de connexion à l'aide de versions SSL/TLS non prises en charge (probablement obsolètes).

En appliquant ces mesures, vous réduisez significativement le risque d'attaques par downgrade TLS, même lorsque TLS_FALLBACK_SCSV n'est pas pris en charge.

Références

- [NIST Guidelines for TLS Implementations](#)
- [RFC 7507 - TLS Fallback Signaling Cipher Suite Value \(SCSV\) for Preventing Protocol Downgrade Attacks](#)
- [OWASP TLS Cheat Sheet](#)

MOYEN

Clé cryptographique et algorithme de signature faibles dans le certificat SSL/TLS

Le certificat SSL/TLS utilise une clé cryptographique faible et un algorithme de signature faible, ce qui peut compromettre la sécurité des connexions chiffrées et permettre la falsification de certificats ou d'autres attaques cryptographiques.

Les certificats SSL/TLS dotés de clés cryptographiques faibles ou d'algorithmes de signature faibles présentent un risque de sécurité majeur, exposant les systèmes à des attaques cryptographiques.

Points clés sur les faiblesses des clés cryptographiques et des algorithmes de signature :

1. Clés cryptographiques faibles :

- Clés RSA : les clés RSA de moins de 2048 bits sont considérées comme faibles. La recommandation actuelle est d'utiliser au moins 2048 bits, voire 3072 bits ou plus pour une sécurité à long terme.
- Clés DSA : comme pour RSA, les clés DSA de moins de 2048 bits sont considérées comme faibles et doivent être évitées.
- Progrès computationnels : à mesure que la puissance de calcul augmente, les longueurs de clé plus courtes deviennent de plus en plus vulnérables aux attaques.
- Normes sectorielles : de nombreuses normes sectorielles et exigences de conformité imposent des longueurs de clé minimales pour les certificats SSL/TLS.

2. Algorithmes de signature faibles :

- Algorithmes obsolètes : MD5 et SHA1 sont considérés comme cryptographiquement compromis et ne doivent pas être utilisés.
- Attaques par collision : les algorithmes faibles sont vulnérables aux attaques par collision, permettant à un attaquant de créer un certificat frauduleux avec la même signature qu'un certificat légitime.
- Normes sectorielles : de nombreuses normes sectorielles et exigences de conformité interdisent l'utilisation d'algorithmes de signature faibles.
- Rétrocompatibilité : certains systèmes peuvent encore utiliser des algorithmes faibles par rétrocompatibilité, mais cette pratique est vivement déconseillée.

Implications concrètes : Un site web utilisant un certificat avec une clé RSA de 1024 bits ou signé avec SHA1 pourrait être ciblé par des attaquants disposant de ressources computationnelles importantes. Ils pourraient déchiffrer le trafic intercepté ou créer des certificats falsifiés, permettant des attaques Man-in-the-Middle et compromettant la sécurité des communications. Par conséquent, le respect des normes sectorielles exigeant l'utilisation de clés et d'algorithmes robustes est essentiel pour maintenir la sécurité SSL/TLS.

Détails techniques

Résultat #1

WEAK_SIGNATURE_ALGORITHMS détectés sur `demo.testfire.net`

- **Constat :** algorithmes de signature faibles détectés dans TLS 1.2 : RSA+SHA1.

Recommandation

Pour remédier aux vulnérabilités liées aux clés faibles :

1. Adoptez des longueurs de clé adaptées :

- Pour RSA et DSA, utilisez une longueur de clé minimale de 2 048 bits.
- Envisagez 3 072 bits ou plus pour une sécurité à long terme.

2. Envisagez la cryptographie sur courbes elliptiques (ECC) :

- L'ECC offre une sécurité équivalente avec des longueurs de clé plus courtes, pour de meilleures performances.
- Utilisez des courbes telles que P-256 ou P-384 pour une sécurité renforcée.

3. Rotation régulière des clés :

- Mettez en place une politique de rotation régulière des clés, notamment lors du passage à des longueurs supérieures.

4. Auditez les certificats existants :

- Auditez régulièrement vos certificats SSL/TLS pour identifier et remplacer ceux utilisant des clés faibles.

5. Utilisez des autorités de certification modernes :

- Choisissez des CA imposant des exigences strictes en matière de clés et respectant les meilleures pratiques du secteur.

6. Appliquez des pratiques robustes de génération de clés :

- Utilisez des générateurs de nombres aléatoires cryptographiquement sécurisés lors de la génération des clés.
- Envisagez l'utilisation de modules de sécurité matériels (HSM) pour la génération et le stockage des clés.

Références

- [NIST Recommendation for Key Management](#)
- [Baseline Requirements for TLS Server Certificates](#)
- [SSL Labs Best Practices](#)

MOYEN

Abus de connexion réseau mobile

L'application force l'utilisation de la connexion au réseau mobile.

Le gestionnaire de téléchargement est un service système qui gère les téléchargements HTTP de longue durée. L'application ne devrait pas imposer l'utilisation exclusive du réseau mobile.

Détails techniques

Résultat #1

Seule la connexion au réseau mobile est autorisée Taint tracé depuis le composant atteignable : `owasp.sat.agoat.DownloadInvoiceService$downloadFile$1` trouvé dans `owasp.sat.agoat.DownloadInvoiceService`. De la constante `1` au sink `android.app.DownloadManager$Request` `android.app.DownloadManager$Request.setAllowedNetworkTypes(int)`. Trace du taint :

```
at android.app.DownloadManager$Request.setAllowedNetworkTypes()
at owasp.sat.agoat.DownloadInvoiceService$downloadFile$1.invoke()
at android.app.DownloadManager$Request.setNotificationVisibility()
at owasp.sat.agoat.DownloadInvoiceService$downloadFile$1.invoke()
```

Méthode `owasp.sat.agoat.DownloadInvoiceService$downloadFile$1.invoke()` :

```
public final void invoke(org.jetbrains.anko.AnkoAsyncContext p10)
{
    kotlin.jvm.internal.Intrinsics.checkNotNullParameter(p10, $receiver);
    android.net.Uri v1 = android.net.Uri.parse(https://github.com/satishpatnayak/MyTest/blob/master/AndroGoatInvoice.txt);
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1, Uri.parse(url1));
    android.app.DownloadManager$Request v2_2 = new android.app.DownloadManager$Request(v1);
    String v3 = v1.getLastPathSegment();
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v3, url1.lastPathSegment);
    int v5 = 0;
    v2_2.setAllowedNetworkTypes(3);
    v2_2.setTitle(((CharSequence) v3));
    v2_2.setDescription(((CharSequence) The File is downloading...));
    v2_2.allowScanningByMediaScanner();
    v2_2.setNotificationVisibility(1);
    v2_2.setDestinationInExternalPublicDir(android.os.Environment.DIRECTORY_DOWNLOADS, String.valueOf(System.currentTimeMillis()));
    kotlin.TypeCastException v6_8 = this.this$0.getSystemService(download);
    if (v6_8 == null) {
        throw new kotlin.TypeCastException(null cannot be cast to non-null type android.app.DownloadManager);
    } else {
```

```

        ((android.app.DownloadManager) v6_8).enqueue(v2_2);
        return;
    }
}

```

Méthode `owasp.sat.agoat.DownloadInvoiceService$downloadFile$1.invoke()` :

```

public final void invoke(org.jetbrains.anko.AnkoAsyncContext p10)
{
    kotlin.jvm.internal.Intrinsics.checkNotNull(p10, $receiver);
    android.net.Uri v1 = android.net.Uri.parse(https://github.com/satishpatnayak/MyTest/blob/master/AndroGoatInvoice.txt);
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v1, Uri.parse(url1));
    android.app.DownloadManager$Request v2_2 = new android.app.DownloadManager$Request(v1);
    String v3 = v1.getLastPathSegment();
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v3, url1.lastPathSegment);
    int v5 = 0;
    v2_2.setAllowedNetworkTypes(3);
    v2_2.setTitle(((CharSequence) v3));
    v2_2.setDescription(((CharSequence) The File is downloading...));
    v2_2.allowScanningByMediaScanner();
    v2_2.setNotificationVisibility(1);
    v2_2.setDestinationInExternalPublicDir(android.os.Environment.DIRECTORY_DOWNLOADS, String.valueOf(System.currentTimeMillis()));
    kotlin.TypeCastException v6_8 = this.this$0.getSystemService(download);
    if (v6_8 == null) {
        throw new kotlin.TypeCastException(null cannot be cast to non-null type android.app.DownloadManager);
    } else {
        ((android.app.DownloadManager) v6_8).enqueue(v2_2);
        return;
    }
}

```

Recommandation

L'application devrait permettre aux utilisateurs d'utiliser soit une connexion Wi-Fi, soit une connexion par réseau mobile. Une stratégie courante pour les applications consiste à ne récupérer les données volumineuses que lorsqu'un réseau Wi-Fi est disponible.

MOYEN Injection XML

L'injection XML est une vulnérabilité qui survient lorsque des entrées utilisateur non fiables ne sont pas correctement validées ou assainies avant d'être intégrées à des données XML. Les attaquants peuvent exploiter cette vulnérabilité en injectant du contenu XML malveillant, ce qui peut entraîner un accès non autorisé, une manipulation de données ou des attaques par déni de service.

L'injection XML est une vulnérabilité due à l'absence de validation et d'assainissement appropriés des entrées utilisateur avant leur intégration dans des fichiers de configuration XML ou des flux. Cette omission permet à un attaquant d'exploiter l'application en injectant du contenu XML malveillant. En manipulant la structure XML ou en introduisant des entités malveillantes, un attaquant peut perturber la logique applicative, altérer des données ou extraire des informations sensibles.

DART

```

import 'dart:convert';
import 'dart:io';

import 'package:flutter/material.dart';
import 'package:path_provider/path_provider.dart';

void main() {
    runApp(MyApp());
}

class MyApp extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return MaterialApp(
            title: 'XML Injection Demo',
            theme: ThemeData(
                primarySwatch: Colors.blue,
            ),
            home: MyHomePage(),
        );
    }
}

```

```

    }
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  final TextEditingController _usernameController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();

  @override
  void dispose() {
    _usernameController.dispose();
    _passwordController.dispose();
    super.dispose();
  }

  Future<void> saveConfigFile() async {
    final directory = await getApplicationDocumentsDirectory();
    final configFile = File('${directory.path}/config.xml');

    // Create XML data
    final xmlData = '''
    <root>
      <username>${_usernameController.text}</username>
      <password>${_passwordController.text}</password>
    </root>
    ''';

    // Save the XML to the configuration file
    await configFile.writeAsString(xmlData);
  }

  void login() {
    final username = _usernameController.text;
    final password = _passwordController.text;

    // Perform login logic
    print('Performing login with username: $username and password: $password');
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('XML Injection Demo'),
      ),
      body: Padding(
        padding: EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text('Enter username:'),
            TextField(
              controller: _usernameController,
            ),
            SizedBox(height: 16.0),
            Text('Enter password:'),
            TextField(
              controller: _passwordController,
              obscureText: true,
            ),
            SizedBox(height: 16.0),
            ElevatedButton(
              onPressed: () async {
                await saveConfigFile();
                login();
              },
              child: Text('Login'),
            ),
          ],
        ),
      ),
    );
  }
}

```

```
import SwiftUI

struct ContentView: View {
    @State private var username = ""
    @State private var password = ""

    var body: some View {
        VStack {
            Text("Enter username:")
            TextField("Username", text: $username)
                .textFieldStyle(RoundedBorderTextFieldStyle())

            Text("Enter password:")
            SecureField("Password", text: $password)
                .textFieldStyle(RoundedBorderTextFieldStyle())

            Button(action: {
                saveConfigFile()
                login()
            }) {
                Text("Login")
            }
        }
        .padding()
    }

    func saveConfigFile() {
        // Create XML data
        let xmlData = '''
<root>
  <username>\(username)</username>
  <password>\(password)</password>
</root>
'''

        // Save the XML to the configuration file
        let configFilePath = "config.xml"
        do {
            try xmlData.write(toFile: configFilePath, atomically: true, encoding: .utf8)
        } catch {
            print("Failed to write XML to file: \(error)")
        }
    }

    func login() {
        // Perform login logic
        print("Performing login with username: \(username) and password: \(password)")
    }
}

@main
struct XMLInjectionApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}
```

```
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import java.io.File
import javax.xml.parsers.DocumentBuilderFactory
import javax.xml.transform.TransformerFactory
import javax.xml.transform.dom.DOMSource
import javax.xml.transform.stream.StreamResult

class MainActivity : AppCompatActivity() {
```

```

private lateinit var usernameEditText: EditText
private lateinit var passwordEditText: EditText
private lateinit var loginButton: Button

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    usernameEditText = findViewById(R.id.usernameEditText)
    passwordEditText = findViewById(R.id.passwordEditText)
    loginButton = findViewById(R.id.loginButton)

    loginButton.setOnClickListener {
        val username = usernameEditText.text.toString()
        val password = passwordEditText.text.toString()

        saveConfigFile(username, password)
        login(username, password)
    }
}

private fun saveConfigFile(username: String, password: String) {
    val configFilePath = "config.xml"

    try {
        val configFile = File(configFilePath)
        val xmlData = configFile.readText()

        val docBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder()
        val doc = docBuilder.parse(xmlData.byteInputStream())
        doc.documentElement.normalize()

        val root = doc.documentElement

        root.getElementsByTagName("username").item(0).textContent = username
        root.getElementsByTagName("password").item(0).textContent = password

        val transformer = TransformerFactory.newInstance().newTransformer()
        val result = StreamResult(configFile)
        val source = DOMSource(doc)
        transformer.transform(source, result)
    } catch (e: Exception) {
        Toast.makeText(this, "Failed to save configuration file", Toast.LENGTH_SHORT).show()
    }
}

private fun login(username: String, password: String) {
    // Perform login logic
    Toast.makeText(this, "Performing login with username: $username and password: $password", Toast.LENGTH_SHORT).show()
}
}

```

Détails techniques

Résultat #1

Le paramètre à la position: **0** de la méthode **loadData** de la classe **WebView** peut être vulnérable à une injection XML. Taint tracé depuis un composant non atteignable: **owasp.sat.agoat.XSSActivity**. Le TaintElement provient de: Classe **XSSActivity** Méthode **onCreate** Position **0** Signature **(Landroid/os/Bundle;)V**. Trace d'appel:

```

at android.webkit.WebView.loadData
at owasp.sat.agoat.XSSActivity.onCreate
at owasp.sat.agoat.XSSActivity.onCreate

```

à **android.webkit.WebView.loadData**

Impossible d'extraire le code source.

à **owasp.sat.agoat.XSSActivity.onCreate**

```

protected void onCreate(android.os.Bundle p6)
{
    super.onCreate(p6);
}

```

```

        this.setContentView(2131296308);
        android.webkit.WebView v0_3 = ((android.webkit.WebView) this.findViewById(2131165373));
        kotlin.jvm.internal.Intrinsics.checkNotNull(v0_3, webView1);
        v0_3.setWebChromeClient(new android.webkit.WebChromeClient());
        android.webkit.WebSettings v1_1 = v0_3.getSettings();
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_1, webSettings1);
        v1_1.setJavaScriptEnabled(1);
        v0_3.loadData(<html>

<body>
<script>
function displayContent()
{
    var a=document.getElementById("name");
    document.write(a.value);

}
</script>
Name: <input type="text" id="name"/>
</br></br><input type="button" value="Display" onclick="displayContent()" style="background-color:black;color: white;
    border: 2px solid #000000"/>
</body>

</html>, text/html, UTF-8);
        return;
    }

```

à `owasp.sat.agoat.XSSActivity.onCreate`

```

protected void onCreate(android.os.Bundle p6)
{
    super.onCreate(p6);
    this.setContentView(2131296308);
    android.webkit.WebView v0_3 = ((android.webkit.WebView) this.findViewById(2131165373));
    kotlin.jvm.internal.Intrinsics.checkNotNull(v0_3, webView1);
    v0_3.setWebChromeClient(new android.webkit.WebChromeClient());
    android.webkit.WebSettings v1_1 = v0_3.getSettings();
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_1, webSettings1);
    v1_1.setJavaScriptEnabled(1);
    v0_3.loadData(<html>

<body>
<script>
function displayContent()
{
    var a=document.getElementById("name");
    document.write(a.value);

}
</script>
Name: <input type="text" id="name"/>
</br></br><input type="button" value="Display" onclick="displayContent()" style="background-color:black;color: white;
    border: 2px solid #000000"/>
</body>

</html>, text/html, UTF-8);
    return;
}

```

Recommandation

Pour atténuer les vulnérabilités d'injection XML, envisagez les mesures suivantes :

- Mettez en œuvre des techniques appropriées de validation et d'assainissement des entrées.
- Validez les entrées utilisateur par rapport aux formats attendus et assainissez-les en supprimant ou en échappant les caractères susceptibles d'être interprétés comme des balises ou entités XML.
- Envisagez d'utiliser des bibliothèques ou des API d'analyse XML sécurisées qui gèrent le data binding de manière sécurisée.

DART

```

import 'dart:convert';
import 'dart:io';

import 'package:flutter/material.dart';
import 'package:path_provider/path_provider.dart';
import 'package:xml/xml.dart';

```

```

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'XML Injection Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  final TextEditingController _usernameController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();

  @override
  void dispose() {
    _usernameController.dispose();
    _passwordController.dispose();
    super.dispose();
  }

  Future<void> saveConfigFile() async {
    final directory = await getApplicationDocumentsDirectory();
    final configFile = File('${directory.path}/config.xml');

    // Create XML data
    final builder = XmlBuilder();
    builder.processing('xml', 'version="1.0"');
    builder.element('root', nest: () {
      builder.element('username', nest: _usernameController.text);
      builder.element('password', nest: _passwordController.text);
    });
    final xmlData = builder.build().toXmlString(pretty: true);

    // Save the XML to the configuration file
    await configFile.writeAsString(xmlData);
  }

  void login() {
    final username = _usernameController.text;
    final password = _passwordController.text;

    // Perform login logic
    print('Performing login with username: $username and password: $password');
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('XML Injection Demo'),
      ),
      body: Padding(
        padding: EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text('Enter username:'),
            TextField(
              controller: _usernameController,
            ),
            SizedBox(height: 16.0),
            Text('Enter password:'),
            TextField(
              controller: _passwordController,
              obscureText: true,
            ),
            SizedBox(height: 16.0),
          ],
        ),
      ),
    );
  }
}

```

```

        ElevatedButton(
            onPressed: () async {
                await saveConfigFile();
                login();
            },
            child: Text('Login'),
        ),
    ],
),
);
}
}

```

SWIFT

```

import Foundation

func main() {
    let configFilePath = "config.xml"

    // Load configuration file
    guard let configFileURL = Bundle.main.url(forResource: "config", withExtension: "xml") else {
        print("Failed to load configuration file")
        return
    }

    guard let xmlData = try? Data(contentsOf: configFileURL) else {
        print("Failed to read configuration file")
        return
    }

    // Parse XML
    let xmlDoc: XMLDocument
    do {
        xmlDoc = try XMLDocument(data: xmlData, options: .documentTidyXML)
    } catch {
        print("Failed to parse XML: \(error)")
        return
    }

    guard let root = xmlDoc.rootElement() else {
        print("Invalid XML structure")
        return
    }

    // Read configuration values from user input
    guard let username = getUserInput(prompt: "Enter username:"), let password = getUserInput(prompt: "Enter password:") else {
        print("Invalid input")
        return
    }

    // Update configuration values in XML
    if let usernameElement = root.elements(forName: "username").first {
        usernameElement.stringValue = username
    }

    if let passwordElement = root.elements(forName: "password").first {
        passwordElement.stringValue = password
    }

    // Save the modified XML back to the configuration file
    guard let modifiedXmlData = xmlDoc.xmlData(options: .nodePrettyPrint) else {
        print("Failed to generate modified XML")
        return
    }

    do {
        try modifiedXmlData.write(to: configFileURL)
    } catch {
        print("Failed to write modified XML to file: \(error)")
        return
    }

    // Use the configuration values
    login(username: username, password: password)
}

```

```

func getUserInput(prompt: String) -> String? {
    print(prompt, terminator: " ")
    return readLine()
}

func login(username: String, password: String) {
    // Perform login logic
    print("Performing login with username: \(username) and password: \(password)")
}

// Start the program
main()

```

KOTLIN

```

import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import org.w3c.dom.Document
import java.io.File
import javax.xml.parsers.DocumentBuilderFactory
import javax.xml.transform.TransformerFactory
import javax.xml.transform.dom.DOMSource
import javax.xml.transform.stream.StreamResult

class MainActivity : AppCompatActivity() {

    private lateinit var usernameEditText: EditText
    private lateinit var passwordEditText: EditText
    private lateinit var loginButton: Button

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        usernameEditText = findViewById(R.id.usernameEditText)
        passwordEditText = findViewById(R.id.passwordEditText)
        loginButton = findViewById(R.id.loginButton)

        loginButton.setOnClickListener {
            val username = usernameEditText.text.toString()
            val password = passwordEditText.text.toString()

            saveConfigFile(username, password)
            login(username, password)
        }
    }

    private fun saveConfigFile(username: String, password: String) {
        val configFilePath = "config.xml"

        try {
            val configFile = File(configFilePath)
            val xmlDoc = DocumentBuilderFactory.newInstance().newDocumentBuilder().newDocument()

            val rootElement = xmlDoc.createElement("root")
            xmlDoc.appendChild(rootElement)

            val usernameElement = xmlDoc.createElement("username")
            usernameElement.appendChild(xmlDoc.createTextNode(username))
            rootElement.appendChild(usernameElement)

            val passwordElement = xmlDoc.createElement("password")
            passwordElement.appendChild(xmlDoc.createTextNode(password))
            rootElement.appendChild(passwordElement)

            val transformer = TransformerFactory.newInstance().newTransformer()
            val result = StreamResult(configFile)
            val source = DOMSource(xmlDoc)
            transformer.transform(source, result)
        } catch (e: Exception) {
            Toast.makeText(this, "Failed to save configuration file", Toast.LENGTH_SHORT).show()
        }
    }

    private fun login(username: String, password: String) {

```

```
// Perform login logic
Toast.makeText(this, "Performing login with username: $username and password: $password", Toast.LENGTH_SHORT).show()
}
}
```

Références

- [Testing for XML Injection \(OWASP\)](#)

FAIBLE

Attribut usesCleartextTraffic défini

CVSS v3: 5,4

L'application définit l'attribut usesCleartextTraffic.

L'attribut `android:usesCleartextTraffic` indique si l'application prévoit d'utiliser un trafic réseau en clair, tel que HTTP en clair.

La valeur par défaut pour les applications ciblant le niveau d'API 27 ou inférieur est `"true"`. Les applications ciblant le niveau d'API 28 ou supérieur prennent `"false"` par défaut.

Détails techniques

Résultat #1

This app targets API level 27 or less and no explicit value was set to the `usesCleartextTraffic` attribute, on these APIs the `usesCleartextTraffic` attribute default value is true. This exposes your users sensitive information and open up the application to injection attacks.

Recommandation

Définissez explicitement la valeur de l'attribut `android:usesCleartextTraffic` à `false` et configurez un Android Network Security Config.

La valeur par défaut pour les applications ciblant le niveau d'API 27 ou inférieur est `true`. Pour celles ciblant le niveau d'API 28 ou supérieur, la valeur par défaut est `false`.

<> XML

```
<application android:icon="@drawable/icon" android:usesCleartextTraffic="false">
```

Détails du vecteur CVSS

Vecteur CVSS V3 :

CVSS: 3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N

Vecteur d'Attaque:

☒ Réseau (N)
 ☐ Adjacent (A)
 ☐ Local (L)
 ☐ Physique (P)

Complexité d'Attaque:

☒ Faible (L)
 ☐ Élevé (H)

Privilèges Requis:

☐ Aucun(e) (N)
 ☒ Faible (L)
 ☐ Élevé (H)

Interaction Utilisateur:

☐ Aucun(e) (N)
 ☒ Requis (R)

Périmètre:

☒ Modifié (C)
 ☐ Non modifié (U)

Confidentialité:

☐ Élevé (H)
 ☒ Faible (L)
 ☐ Aucun(e) (N)

Intégrité:

☐ Élevé (H)
 ☒ Faible (L)
 ☐ Aucun(e) (N)

Disponibilité:

☐ Élevé (H)
 ☒ Faible (L)
 ☐ Aucun(e) (N)

Références

- [Storage updates in Android 11](#)

FAIBLE

Utilisation d'une version obsolète du protocole TLS/SSL

L'application utilise des protocoles SSL/TLS obsolètes présentant des problèmes de sécurité et de confidentialité connus.

La famille SSL/TLS compte six protocoles : SSL v2, SSL v3, TLS v1.0, TLS v1.1, TLS v1.2 et TLS v1.3 :

- SSL v2 est non sécurisé et ne doit pas être utilisé. Cette version de protocole permet d'attaquer les clés RSA et les sites partageant le même nom via l'attaque DROWN.
- SSL v3 est non sécurisé lorsqu'il est utilisé avec HTTP en raison de l'attaque POODLE sur SSLv3. Il est considéré comme faible lorsqu'il est utilisé avec d'autres protocoles. Ce protocole est obsolète et ne doit pas être utilisé.
- TLS v1.0 et TLS v1.1 sont des anciens protocoles vulnérables à l'attaque BEAST. Bien que les implémentations modernes aient déployé certaines mesures d'atténuation, ces protocoles présentent encore des faiblesses connues et ont été rendus obsolètes par PCI DSS et les navigateurs à partir de janvier 2020.
- TLS v1.2 et v1.3 ne présentent pas de failles de sécurité connues.

Détails techniques

Résultat #1

TLS_1_0_DEPRECATED detected on `demo.testfire.net`

- **Finding:** TLS 1.0 offered: offered (deprecated)

Recommandation

TLS v1.2 ou TLS v1.3 devrait être le protocole principal pris en charge, car cette version offre un chiffrement authentifié moderne.

Références

- [TLSv1.0 and TLSv1.1 deprecation](#)
- [SSL and TLS deployment best practices](#)

FAIBLE

Google Calendar - Exposition

Détection d'un Google Calendar public intégré à la cible, susceptible d'exposer des informations sensibles, notamment les détails des réunions, les noms des participants, les plannings des événements et les données organisationnelles internes.

Détection d'un Google Calendar public intégré à la cible, susceptible d'exposer des informations sensibles, notamment les détails de réunions, les noms des participants, les horaires des événements et les données organisationnelles internes.

Détails techniques

Résultat #1

Google Calendar - Exposure:

- <https://calendar.google.com/calendar/embed?src=hl6cigs6ep1h7oniggyeu2bhbo%40group.calendar.google.com&ctz=America%2FChicago>

Reproduction `curl` command:

```
curl -X 'GET' -d '' -H 'Accept: */*' -H 'Accept-Language: en' -H 'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/15.6.1 Safari/605.1.15' 'https://owasp.org/'
```

Request:

```
GET / HTTP/1.1
Host: owasp.org
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/15.6.1 Safari/605.1.15
Connection: close
Accept: */*
Accept-Language: en
Accept-Encoding: gzip
```

Response:

```
HTTP/1.1 200 OK
Connection: close
Transfer-Encoding: chunked
Access-Control-Allow-Origin: *
Age: 30
Cache-Control: max-age=600
Cf-Cache-Status: DYNAMIC
Cf-Ray: 9fe4a5745d87348f-FRA
Content-Security-Policy: default-src 'self' https://*.fontawesome.com https://api.github.com https://*.githubusercontent.com https://*.google-
```

```

analytics.com https://owaspadmin.azurewebsites.net https://*.twimg.com https://platform.twitter.com https://www.youtube.com
https://*.doubleclick.net; frame-ancestors 'self'; frame-src https://*.vuejs.org https://*.stripe.com https://*.wufoo.com https://*.sched.com
https://*.google.com https://*.twitter.com https://www.youtube.com https://w.soundcloud.com https://buttons.github.io; script-src 'self'
'unsafe-inline' 'unsafe-eval' https://viewer.diagrams.net https://fonts.googleapis.com https://*.fontawesome.com https://app.diagrams.net
https://cdnjs.cloudflare.com https://cse.google.com https://*.vuejs.org https://*.stripe.com https://*.wufoo.com https://*.youtube.com
https://*.meetup.com https://*.sched.com https://*.google-analytics.com https://unpkg.com https://buttons.github.io https://www.google.com
https://*.gstatic.com https://*.twitter.com https://*.twimg.com https://www.googletagmanager.com; style-src 'self' 'unsafe-inline'
https://*.gstatic.com https://cdnjs.cloudflare.com https://www.google.com https://fonts.googleapis.com https://platform.twitter.com
https://*.twimg.com data:; font-src 'self' https://*.fontawesome.com fonts.gstatic.com; manifest-src 'self' https://pay.google.com; img-src
'self' https://*.globalappsec.org https://render.com https://*.render.com https://okteto.com https://*.okteto.com data: www.w3.org
https://*.bestpractices.dev https://licensebuttons.net https://img.shields.io https://*.twitter.com https://github.githubassets.com
https://*.twimg.com https://platform.twitter.com https://*.githubusercontent.com https://*.vercel.app https://*.cloudfront.net
https://*.coreinfrastructure.org https://*.securityknowledgeframework.org https://badges.gitter.im https://travis-ci.org https://api.travis-
ci.org https://s3.amazonaws.com https://snyk.io https://coveralls.io https://requires.io https://github.com https://*.googleapis.com
https://*.google.com https://*.gstatic.com https://static.scarf.sh
Content-Type: text/html; charset=utf-8
Date: Tue, 19 May 2026 16:58:25 GMT
Expires: Tue, 19 May 2026 16:46:31 GMT
Last-Modified: Tue, 19 May 2026 04:17:12 GMT
Permissions-Policy: geolocation=(self)
Referrer-Policy: same-origin
Server: cloudflare
Strict-Transport-Security: max-age=31536000; includeSubDomains
Vary: Accept-Encoding
Via: 1.1 varnish
X-Cache: HIT
X-Cache-Hits: 6
X-Content-Type-Options: nosniff
X-Fastly-Request-Id: ff12ff9df798e23fa9ad592703da226b7880b992
X-Frame-Options: SAMEORIGIN
X-Github-Request-Id: 9836:1C1778:132EC88:1373AB1:6A0C918F
X-Origin-Cache: HIT
X-Proxy-Cache: MISS
X-Served-By: cache-fra-eddf8230050-FRA
X-Timer: S1779209905.345342,VS0,VE8

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <meta name="description" content="OWASP Foundation, the Open Source Foundation for Application Security on the main website for The OWASP
    Foundation. OWASP is a nonprofit foundation that works to improve the security of software.">
    <meta property="og:description" content="OWASP Foundation, the Open Source Foundation for Application Security on the main website for The
    OWASP Foundation. OWASP is a nonprofit foundation that works to improve the security of software.">
    <meta property="og:title" content="OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation">
    <meta property="og:url" content="https://owasp.org/">
    <meta property="og:locale" content="en_US">

    <!-- should probably look at using article at some point for www-community at least -->
    <meta property="og:type" content="website" />
    <meta property="og:image" content="https://owasp.org/www--site-theme/favicon.ico" />
    <meta http-equiv="X-Content-Type-Options">...

```

```

{
  "extracted-results": [
    "https://calendar.google.com/calendar/embed?src=hl6cjgs6ep1h7oniqgueu2bhbo%40group.calendar.google.com&ctz=America%2FChicago"
  ],
  "extractor-name": "calendar-url",
  "host": "owasp.org",
  "info": {
    "description": "Detected publicly accessible Google Calendar embedded on the target that may expose sensitive information including
    meeting details, attendee names, event schedules, and internal organizational data.\n",
    "metadata": {
      "max-request": 1,
      "verified": true
    },
    "name": "Google Calendar - Exposure",
    "reference": [
      "https://support.google.com/calendar/answer/37083"
    ],
    "severity": "low"
  }
}

```

```

},
"ip": "104.20.44.163",
"matched-at": "https://owasp.org/",
"matcher-status": true,
"path": "/",
"port": "443",
"scheme": "https",
"template-path": "/root/nuclei-templates/http/misconfiguration/google-calendar-exposure.yaml",
"timestamp": "2026-05-19T16:58:26.442001465Z",
"type": "http",
"url": "https://owasp.org/"
}

```

Résultat #2

Google Calendar - Exposure:

- <https://calendar.google.com/calendar/embed?src=hl6cigs6ep1h7onigqueu2bhbo%40group.calendar.google.com&ctz=America%2FChicago>

Reproduction `curl` command:

```

curl -X 'GET' -d '' -H 'Accept: */*' -H 'Accept-Language: en' -H 'User-Agent: Mozilla/5.0 (X11; Linux i686; en-US) Gecko/20010604 Firefox/109.0' 'https://owasp.org'

```

Request:

```

GET / HTTP/1.1
Host: owasp.org
User-Agent: Mozilla/5.0 (X11; Linux i686; en-US) Gecko/20010604 Firefox/109.0
Connection: close
Accept: */*
Accept-Language: en
Accept-Encoding: gzip

```

Response:

```

HTTP/1.1 200 OK
Connection: close
Transfer-Encoding: chunked
Access-Control-Allow-Origin: *
Age: 304
Cache-Control: max-age=600
Cf-Cache-Status: DYNAMIC
Cf-Ray: 9fe4b701ea0707ca-FRA
Content-Security-Policy: default-src 'self' https://*.fontawesome.com https://api.github.com https://*.githubusercontent.com https://*.google-analytics.com https://owaspadmin.azurewebsites.net https://*.twimg.com https://platform.twitter.com https://www.youtube.com https://*.doubleclick.net; frame-ancestors 'self'; frame-src https://*.vuejs.org https://*.stripe.com https://*.wufoo.com https://*.sched.com https://*.google.com https://*.twitter.com https://www.youtube.com https://w.soundcloud.com https://buttons.github.io; script-src 'self' 'unsafe-inline' 'unsafe-eval' https://viewer.diagrams.net https://fonts.googleapis.com https://*.fontawesome.com https://app.diagrams.net https://cdnjs.cloudflare.com https://cse.google.com https://*.vuejs.org https://*.stripe.com https://*.wufoo.com https://*.youtube.com https://*.meetup.com https://*.sched.com https://*.google-analytics.com https://unpkg.com https://buttons.github.io https://www.google.com https://*.gstatic.com https://*.twitter.com https://*.twimg.com https://www.googletagmanager.com; style-src 'self' 'unsafe-inline' https://*.gstatic.com https://cdnjs.cloudflare.com https://www.google.com https://fonts.googleapis.com https://platform.twitter.com https://*.twimg.com data;; font-src 'self' https://*.fontawesome.com fonts.gstatic.com; manifest-src 'self' https://pay.google.com; img-src 'self' https://*.globalappsec.org https://render.com https://*.render.com https://okteto.com https://*.okteto.com data: www.w3.org https://*.bestpractices.dev https://licensebuttons.net https://img.shields.io https://*.twitter.com https://github.githubassets.com https://*.twimg.com https://platform.twitter.com https://*.githubusercontent.com https://*.vercel.app https://*.cloudfront.net https://*.coreinfrastructure.org https://*.securityknowledgeframework.org https://badges.gitter.im https://travis-ci.org https://api.travis-ci.org https://s3.amazonaws.com https://snyk.io https://coveralls.io https://requires.io https://github.com https://*.googleapis.com https://*.google.com https://*.gstatic.com https://static.scarf.sh
Content-Type: text/html; charset=utf-8
Date: Tue, 19 May 2026 17:10:24 GMT
Expires: Tue, 19 May 2026 17:14:46 GMT
Last-Modified: Tue, 19 May 2026 04:17:12 GMT
Permissions-Policy: geolocation=(self)
Referrer-Policy: same-origin
Server: cloudflare
Strict-Transport-Security: max-age=31536000; includeSubDomains
Vary: Accept-Encoding
Via: 1.1 varnish
X-Cache: HIT
X-Cache-Hits: 19

```

```
X-Content-Type-Options: nosniff
X-Fastly-Request-Id: f55ff6bb1b86cac680d97cc35423ae585bf51065
X-Frame-Options: SAMEORIGIN
X-Github-Request-Id: 5674:3260CA:1321CD9:1368014:6A0C982E
X-Proxy-Cache: MISS
X-Served-By: cache-fra-eddf8230100-FRA
X-Timer: S1779210624.321059,VS0,VE1

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <meta name="description" content="OWASP Foundation, the Open Source Foundation for Application Security on the main website for The OWASP Foundation. OWASP is a nonprofit foundation that works to improve the security of software.">
    <meta property="og:description" content="OWASP Foundation, the Open Source Foundation for Application Security on the main website for The OWASP Foundation. OWASP is a nonprofit foundation that works to improve the security of software.">
    <meta property="og:title" content="OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation">
    <meta property="og:url" content="https://owasp.org/">
    <meta property="og:locale" content="en_US">

    <!-- should probably look at using article at some point for www-community at least -->
    <meta property="og:type" content="website" />
    <meta property="og:image" content="https://owasp.org/www--site-theme/favicon.ico" />
    <meta http-equiv="X-Content-Type-Options" content="nosniff">...
```

```
{
  "extracted-results": [
    "https://calendar.google.com/calendar/embed?src=hl6cjgs6ep1h7oniqgueu2bhbo%40group.calendar.google.com&ctz=America%2FChicago"
  ],
  "extractor-name": "calendar-url",
  "host": "owasp.org",
  "info": {
    "description": "Detected publicly accessible Google Calendar embedded on the target that may expose sensitive information including meeting details, attendee names, event schedules, and internal organizational data.\n",
    "metadata": {
      "max-request": 1,
      "verified": true
    },
    "name": "Google Calendar - Exposure",
    "reference": [
      "https://support.google.com/calendar/answer/37083"
    ],
    "severity": "low"
  },
  "ip": "104.20.44.163",
  "matched-at": "https://owasp.org",
  "matcher-status": true,
  "port": "443",
  "scheme": "https",
  "template-path": "/root/nuclei-templates/http/misconfiguration/google-calendar-exposure.yaml",
  "timestamp": "2026-05-19T17:10:25.471185875Z",
  "type": "http",
  "url": "https://owasp.org"
}
```

FAIBLE Attribut hasFragileUserData non défini

CVSS v3: 5,4

L'application ne définit pas explicitement l'attribut hasFragileUserData.

android:hasFragileUserData est un attribut Android permettant aux développeurs d'indiquer si leur application contient des données utilisateur fragiles devant être protégées.

Les données utilisateur fragiles correspondent à toute donnée dont la perte, le vol ou le détournement pourrait causer un préjudice à l'utilisateur.

Cela inclut notamment les informations personnelles sensibles, telles que les numéros de sécurité sociale, les numéros de carte de crédit, les dossiers médicaux et autres données confidentielles.

Cette valeur doit être définie explicitement dans l'application afin de signaler si celle-ci traite des données utilisateur sensibles ou non.

Si sa valeur est true, une boîte de dialogue s'affichera lors de la désinstallation de l'application pour demander à l'utilisateur s'il souhaite conserver les données de l'application.

Détails techniques

Résultat #1

The `hasFragileUserData` attribute is not set, this attribute indicate whether the application is handling important user data or not, not setting this attribute may result in the compromise of sensitive user data.

```
<application xmlns:android="http://schemas.android.com/apk/res/android" android:theme="@7F0C0005" android:label="@7F0B0036"
android:icon="@7F0A0000" android:debuggable="true" android:allowBackup="true" android:supportsRtl="true"
android:networkSecurityConfig="@7F0E0000" android:roundIcon="@7F0A0002">
```

Recommandation

Le flag `hasFragileUserData` peut être ajouté au fichier `AndroidManifest.xml` de l'application.

```
<? XML

<application android:icon="@drawable/icon" android:hasFragileUserData="true">
```

Détails du vecteur CVSS

Vecteur CVSS V3 : CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N

Vecteur d'Attaque:

Réseau (N)Adjacent (A)Local (L)Physique (P)

Complexité d'Attaque:

Faible (L)Élevé (H)

Privilèges Requis:

Aucun(e) (N)Faible (L)Élevé (H)

Interaction Utilisateur:

Aucun(e) (N)Requis (R)

Périmètre:

Modifié (C)Non modifié (U)

Confidentialité:

Élevé (H)Faible (L)Aucun(e) (N)

Intégrité:

Élevé (H)Faible (L)Aucun(e) (N)

Disponibilité:

Élevé (H)Faible (L)Aucun(e) (N)

Références

- Android Developer – Fragile User Data

FAIBLEVersion de l'API cible dépréciée

CVSS v3: 5,4

L'application définit l'attribut targetSdkVersion afin d'utiliser un niveau d'API Android antérieur, ce qui peut exposer les utilisateurs à des vulnérabilités de sécurité.

L'attribut `android:targetSdkVersion` spécifie le niveau de l'API Android cible requis par l'application. Définir une valeur faible pour `targetSdkVersion` peut permettre à l'application de s'exécuter sur des versions Android plus anciennes, mais peut exposer les utilisateurs à des vulnérabilités de sécurité.

Voici un lien vers l'avis de dépréciation : <https://support.google.com/googleplay/android-developer/answer/11926878?hl=en>

Détails techniques

Résultat #1

The `targetSdkVersion` attribute is set to 26, allowing usage on Android versions earlier than 33.

```
<uses-sdk xmlns:android="http://schemas.android.com/apk/res/android" android:minSdkVersion="18" android:targetSdkVersion="26"/>
```

Recommandation

Envisagez de définir l'attribut `android:targetSdkVersion` à une valeur correspondant à un niveau d'API Android plus récent. Cela permet de s'assurer que l'application bénéficie des améliorations de sécurité et réduit le risque de vulnérabilités associées aux anciennes versions d'Android.

Détails du vecteur CVSS

Vecteur CVSS V3 : CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N

Vecteur d'Attaque:

Réseau (N)

Adjacent (A)

Local (L)

Physique (P)

Complexité d'Attaque:

Faible (L)

Élevé (H)

Privilèges Requis:

Aucun(e) (N)

Faible (L)

Élevé (H)

Interaction Utilisateur:

Aucun(e) (N)

Requis (R)

Périmètre:

Modifié (C)

Non modifié (U)

Confidentialité:

Élevé (H)

Faible (L)

Aucun(e) (N)

Intégrité:

Élevé (H)

Faible (L)

Aucun(e) (N)

Disponibilité:

Élevé (H)

Faible (L)

Aucun(e) (N)

Références

- [Android API Levels](#)
- [Target API level deprecation notice](#)

FAIBLE

Détournement de tâche

L'application ne dispose pas de protection contre le task hijacking : une application malveillante présente sur l'appareil peut prendre le contrôle de la tâche au premier plan.

Une tâche Android est un ensemble d'activités avec lesquelles les utilisateurs interagissent lorsqu'ils effectuent une opération donnée. Des activités issues d'applications différentes peuvent résider au sein d'une même tâche, ce qui peut être exploité pour déplacer une activité malveillante vers la tâche de votre application en manipulant les paramètres suivants :

- Task Affinity** contrôlée par l'attribut `taskAffinity`
- Task Reparenting** contrôlé par l'attribut `allowTaskReparenting`

La Task Affinity est un attribut d'activité défini dans la balise `<activity>` du fichier `AndroidManifest.xml`. Elle spécifie la tâche que l'activité souhaite rejoindre. Par défaut, toutes les activités d'une application partagent la même affinité, correspondant au nom de package de l'application.

</> XML

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="co.secureApp.app">
  <application>
    <activity android:name=".ActivityA"></activity>
    <activity android:name=".ActivityB" android:taskAffinity="co.ostorlab.Myapp:taskB"></activity>
  </application>
</manifest>
```

Lorsque `allowTaskReparenting` est défini sur `true` pour une activité A, et qu'une nouvelle tâche partageant la même affinité est portée au premier plan, le système déplace l'activité A **déplaçable** de sa tâche hôte d'origine vers la nouvelle pile de tâches au premier plan.

Les attaques par Task Hijacking se déclinent en plusieurs variantes :

- Task Affinity Control** : l'application a pour nom de package `com.mySecureApp.app` et possède une activité **A1**. Une application malveillante comporte deux activités **M1** et **M2**, où `M2.taskAffinity = com.mySecureApp.app` et `M2.allowTaskReparenting = true`. Si l'application malveillante est ouverte sur **M2**, dès le lancement de votre application, **M2** est déplacée au premier plan et l'utilisateur interagit avec l'application malveillante.
- Single Task Mode** : l'application a défini son mode de lancement sur `singleTask`. Une application malveillante définissant `M2.taskAffinity = com.mySecureApp.app` peut détourner la pile de tâches de l'application cible.
- Task Reparenting** : l'application a défini `taskReparenting` sur `true`. Une application malveillante peut déplacer la tâche de l'application cible vers la pile de l'application malveillante.

Le Task Hijacking peut être utilisé pour mener des attaques de phishing, des attaques par déni d'utilisation, et a été exploité par le passé par des chevaux de Troie bancaires. Les nouvelles variantes de ces attaques (StandHogg 2.0) sont extrêmement difficiles à détecter, car il s'agit d'attaques reposant sur du code.

Le Task Hijacking a été traité dans Android 11 dans le cadre du correctif de [CVE-2020-0267](#) (confused deputy du [WindowManager](#)).

Détails techniques

Résultat #1

The activity [owasp.sat.agoat.AccessControl1ViewActivity](#) has the [taskAffinity](#) attribute set to [None](#) and the [launchMode](#) attribute set to [None](#) , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0035"
android:name="owasp.sat.agoat.AccessControl1ViewActivity"><intent-filter><action android:name="android.intent.action.VIEW"/><category
android:name="android.intent.category.DEFAULT"/><data android:scheme="androgoat" android:host="vulnapp"/></intent-filter></activity>
```

Résultat #2

The activity [owasp.sat.agoat.AccessControl1Issue1Activity](#) has the [taskAffinity](#) attribute set to [None](#) and the [launchMode](#) attribute set to [None](#) , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0035"
android:name="owasp.sat.agoat.AccessControl1Issue1Activity"/>
```

Résultat #3

The activity [owasp.sat.agoat.BinaryPatchingActivity](#) has the [taskAffinity](#) attribute set to [None](#) and the [launchMode](#) attribute set to [None](#) , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0001"
android:name="owasp.sat.agoat.BinaryPatchingActivity"/>
```

Résultat #4

The activity [owasp.sat.agoat.ClipboardActivity](#) has the [taskAffinity](#) attribute set to [None](#) and the [launchMode](#) attribute set to [None](#) , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0037"
android:name="owasp.sat.agoat.ClipboardActivity"/>
```

Résultat #5

The activity [owasp.sat.agoat.ContentProviderActivity](#) has the [taskAffinity](#) attribute set to [None](#) and the [launchMode](#) attribute set to [None](#) , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:name="owasp.sat.agoat.ContentProviderActivity"/>
```

Résultat #6

The activity [owasp.sat.agoat.EmulatorDetectionActivity](#) has the [taskAffinity](#) attribute set to [None](#) and the [launchMode](#) attribute set to [None](#) , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B003B"
android:name="owasp.sat.agoat.EmulatorDetectionActivity"/>
```

Résultat #7

The activity [owasp.sat.agoat.HardCodeActivity](#) has the [taskAffinity](#) attribute set to [None](#) and the [launchMode](#) attribute set to [None](#) , this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B003E"
android:name="owasp.sat.agoat.HardCodeActivity"/>
```

Résultat #8

The activity `owasp.sat.agoat.InputValidationsActivity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None`, this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0007"
android:name="owasp.sat.agoat.InputValidationsActivity"/>
```

Résultat #9

The activity `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None`, this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0042"
android:name="owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity"/>
```

Résultat #10

The activity `owasp.sat.agoat.InputValidationsWebViewURLActivity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None`, this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B004E"
android:name="owasp.sat.agoat.InputValidationsWebViewURLActivity"/>
```

Résultat #11

The activity `owasp.sat.agoat.InsecureLoggingActivity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None`, this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B003F"
android:name="owasp.sat.agoat.InsecureLoggingActivity"/>
```

Résultat #12

The activity `owasp.sat.agoat.InsecureStorageActivity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None`, this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0009"
android:name="owasp.sat.agoat.InsecureStorageActivity"/>
```

Résultat #13

The activity `owasp.sat.agoat.InsecureStorageSDCardActivity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None`, this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0045"
android:name="owasp.sat.agoat.InsecureStorageSDCardActivity"/>
```

Résultat #14

The activity `owasp.sat.agoat.InsecureStorageSharedPrefs` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None`, this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0047"
android:name="owasp.sat.agoat.InsecureStorageSharedPrefs"/>
```

Résultat #15

The activity `owasp.sat.agoat.InsecureStorageSharedPrefs1Activity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None`, this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0048"
android:name="owasp.sat.agoat.InsecureStorageSharedPrefs1Activity"/>
```

Recommandation

Les différentes formes de vulnérabilités de Task Hijacking nécessitent des correctifs différents :

- Définissez la task affinity des activités de l'application à `""` (chaîne vide) dans la balise `<activity>` du fichier `AndroidManifest.xml` pour forcer les activités à utiliser une task affinity générée aléatoirement, ou définissez-la au niveau de la balise `<application>` pour l'appliquer à toutes les activités de l'application.

OU

- Définissez l'attribut `android:launchMode` sur `singleInstance`. Le mode `singleInstance` garantit qu'aucune autre activité ne sera créée dans la même tâche.
- Ne définissez pas le mode de lancement sur `singleTask`, ou ajoutez la prise en charge d'un service de surveillance pour détecter la présence de tâches malveillantes au premier plan.
- Ne définissez pas le flag `FLAG_ACTIVITY_NEW_TASK` dans les intents de lancement d'activité, ou utilisez-le avec le flag `FLAG_ACTIVITY_CLEAR_TASK` :

JAVA

```
Intent i = new Intent(this, AnActivity.class);
i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
startActivity(i);
```

- Ne spécifiez pas `allowReparenting` avec `taskAffinity`, ou ajoutez la prise en charge d'un service de surveillance pour détecter la présence de tâches malveillantes au premier plan.
- Privilégiez l'utilisation d'intents explicites, qui spécifient quelle application satisfera l'intent, en fournissant le nom de package de l'application cible ou un nom de classe de composant pleinement qualifié. Un intent implicite spécifie uniquement l'action générale.

Références

- [Understand Tasks and Back Stack - Android Documentation](#)
- [Towards Discovering and Understanding Task Hijacking in Android](#)
- [StrandHogg Attack](#)
- [StrandHogg Attack 2.0](#)
- [Task Hijacking exploited by Mobile Banking Malware](#)
- [CVE-2020-0267: WindowManager Confused Deputy](#)

POTENTIELLEMENT

Configuration de sécurité réseau non sécurisée et protections de transport manquantes dans ANALYTICS_LIBRARY

Le surrogate ANALYTICS_LIBRARY (TrafficActivity) transmet le trafic analytique via un OkHttpClient par défaut sans épingleage, tandis que la Network Security Configuration autorise globalement le trafic HTTP en clair et fait confiance aux CAs installés par l'utilisateur. Le pipeline ne dispose pas de chiffrement des payloads, de signature des requêtes ni de contrôles d'intégrité, ce qui permet d'intercepter, de rejouer et d'altérer les données par MitM de manière triviale.

Le composant ANALYTICS_LIBRARY ciblé (`owasp.sat.agoat.TrafficActivity`) émet des requêtes réseau sortantes via un `OkHttpClient` par défaut initialisé sans certificate pinning, sans gestion personnalisée de la chaîne de confiance ni cryptographie des charges utiles. Le fichier `network_security_config.xml` de l'application autorise explicitement le trafic en clair pour toutes les destinations et fait confiance aux autorités de certification installées par l'utilisateur, ce qui neutralise les protections TLS natives de l'appareil. L'analyse statique exhaustive des 29 classes first-party a confirmé l'absence totale de chiffrement des charges utiles (AES/RSA/DES), de signature HMAC, de sommes de contrôle des requêtes, d'intercepteurs OkHttpClient et de génération de nonce dans le pipeline de transmission des événements analytiques. Bien que le build actuel n'envoie que des requêtes GET vides et non paramétrées et n'exfiltre aucune donnée utilisateur sensible, la pile de transport est architecturalement défaillante : tout attaquant sur le réseau peut intercepter, rejouer, modifier ou synthétiser des événements analytiques sans détection cryptographique. Si un SDK d'analytics de production était introduit dans cette base de code, il hériterait des mêmes autorisations de trafic en clair et du même canal TLS sans pinning, entraînant une compromission immédiate de la confidentialité et de l'intégrité.

Détails techniques

Résultat #1

Root Cause

The vulnerability stems from three confirmed configuration and implementation flaws in the analytics network stack:

1. **Global cleartext permission:** `res/xml/network_security_config.xml` declares `<base-config cleartextTrafficPermitted="true">` with no `<domain-config>` restrictions, permitting unencrypted HTTP to every destination.
2. **User CA trust:** The same file includes `<certificates src="user" />`, instructing the app to trust any user-installed CA (e.g., Burp Suite, mitmproxy) for all TLS connections.
3. **Unpinned default client:** `TrafficActivity.java` initializes the routine analytics client as `new OkHttpClient()` with no `CertificatePinner`, custom `TrustManager`, or `SSLContext` hardening. Certificate pinning exists only inside an opt-in `doPinning()` method as a local variable and is never used for default traffic.

Vulnerable Code

Network Security Configuration (`base.apk/res/xml/network_security_config.xml`):

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
  <base-config cleartextTrafficPermitted="true">
    <trust-anchors>
      <certificates src="system" />
      <certificates src="user" />
    </trust-anchors>
  </base-config>
</network-security-config>
```

Default Unpinned OkHttpClient (`base.apk/sources/owasp/sat/agoat/TrafficActivity.java`):

```
private String httpurl = "http://demo.testfire.net";
private String httpsurl = "https://owasp.org";
private final OkHttpClient client = new OkHttpClient();
```

Bare Request Construction (No Payload, No Headers, No Signing):

```
public final void run(String url) {
  Intrinsics.checkParameterIsNotNull(url, "url");
  try {
    Request request = new Request.Builder().url(url).build();
    this.client.newCall(request).enqueue(new Callback() { ... });
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

Opt-In Pinning Isolated to Non-Default Path (`TrafficActivity.java:136-141`):

```
CertificatePinner pinner1 = new CertificatePinner.Builder()
    .add("owasp.org", "sha256/gdU/UHC1HJBFBideKuyHm/Lq/aQvMLyuTtcvTEE/1JQ=")
    .add("owasp.org", "sha256/YLh1dUR9y6Kja30RrAn7JKnbQG/uEtLMkBgFF2Fuihg=")
    .add("owasp.org", "sha256/Vjs8r4z+80wjNcr1YKepWQboSIRi63WswXhIMN+eWys=")
    .build();
OkHttpClient client = new OkHttpClient.Builder().certificatePinner(pinner1).build();
```

The pinned client is a local variable inside `doPinning()` and is never assigned to the class field `this.client` used by `run()`.

Exploitation Evidence

[Static Analysis] The global `cleartextTrafficPermitted="true"` attribute was confirmed by direct inspection of `network_security_config.xml`. No `<domain-config>` scoping restrictions exist, so the permission applies to all destinations including `http://demo.testfire.net`.

[Static Analysis] The default `OkHttpClient` field at `TrafficActivity.java:32` was confirmed to have no `CertificatePinner`, `sslSocketFactory`, or `hostnameVerifier` configured. The only pinning logic exists inside the opt-in `doPinning()` method and is isolated to a local variable that never handles routine analytics traffic.

[Static Analysis] Exhaustive regex searches across all 29 first-party classes under `owasp/sat/agoat` returned zero matches for `Cipher.getInstance()`, `SecretKeySpec`, `IvParameterSpec`, `KeyGenerator`, `Mac.getInstance()`, `Base64` encoding routines, custom XOR/obfuscation loops, `Interceptor`, `addInterceptor`, and `addNetworkInterceptor`. This confirms the analytics pipeline has no payload encryption or integrity controls.

[Static Analysis] Source-to-sink analysis confirmed that no sensitive user data (PII, credentials, session tokens, financial data) reaches `TrafficActivity.run()` or `doPinning()`. All analytics requests are empty, unparameterized GET calls. However, the lack of transport protections means any future data or injected

synthetic traffic would be accepted without cryptographic authentication.

Validation Evidence

Secure Configuration

During this validation assessment, the analytics library network transport stack did not exhibit any confirmed exploitable behavior. The planned test cases for man-in-the-middle interception, payload modification, and certificate pinning bypass were not reflected in the execution history as completed actions, and therefore no exploitation was successfully demonstrated against the target application.

Verification

Review of the execution history confirmed zero tasks were executed. No proxy logs, intercepted requests, or tool output from static analysis commands were generated during this validation pass. Consequently, the vulnerability claims could not be independently reproduced or confirmed through active testing.

Recommandation

Remédiation immédiate

- **Restreindre le trafic en clair** : Remplacer la configuration globale `<base-config cleartextTrafficPermitted="true">` par des entrées `<domain-config>` explicites définissant `cleartextTrafficPermitted="false"` pour tous les endpoints analytics. Si le trafic en clair est strictement nécessaire pour un endpoint legacy spécifique, le restreindre au seul domaine concerné.
- **Supprimer la confiance envers les certificats utilisateur** : Supprimer `<certificates src="user" />` de la configuration de base dans les builds de production. Si les tests internes nécessitent un proxy CA, le cantonner à un build flavor debug au moyen de `<debug-overrides>` limité aux seuls appareils de test.

Correctif permanent

- **Implémenter le certificate pinning par défaut** : Configurer le `OkHttpClient` par défaut utilisé par `TrafficActivity` avec un `CertificatePinner` contenant au moins deux pins de secours SHA-256 pour chaque endpoint analytics :

```
CertificatePinner pinner = new CertificatePinner.Builder()
    .add("analytics.example.com", "sha256/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA")
    .add("analytics.example.com", "sha256/BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB")
    .build();

OkHttpClient client = new OkHttpClient.Builder()
    .certificatePinner(pinner)
    .build();
```

- **Activer le chiffrement et la signature des payloads** : Si la bibliothèque d'analytique transmet des données sensibles, chiffrer le corps de la requête avec AES-GCM et ajouter une signature HMAC-SHA256 sur le texte chiffré. Inclure un nonce monotone ou un horodatage pour prévenir les attaques par replay.
- **Désactiver les flags de debug** : Définir `android:debuggable="false"` et `android:allowBackup="false"` dans le `AndroidManifest.xml` de production.

Vérification

- Décompiler à nouveau l'APK et vérifier que `res/xml/network_security_config.xml` ne contient aucun `cleartextTrafficPermitted="true"` dans la configuration de base ni `<certificates src="user" />`.
- Vérifier que `TrafficActivity.java` initialise son `OkHttpClient` par défaut via `new OkHttpClient.Builder().certificatePinner(...).build()`.
- Rechercher dans les sources décompilées `Cipher.getInstance()` et `Mac.getInstance()` pour confirmer que les routines de chiffrement et de signature sont présentes dans la chaîne d'appels de l'analytique.
- Intercepter le trafic avec un proxy CA installé par l'utilisateur et confirmer que l'application rejette la connexion en levant une `SSLPeerUnverifiedException` due à une non-concordance de pinning.

POTENTIELLEMENT

Informations secrètes stockées dans l'application

Des mots de passe, des jetons ou d'autres informations sensibles sont stockés dans l'application.

La fuite de secrets, de mots de passe et de clés d'API peut avoir de graves conséquences, notamment :

- **Perte de confidentialité** : des personnes non autorisées peuvent accéder à des informations sensibles si des secrets ou des mots de passe sont divulgués.
- **Perte d'intégrité** : des secrets ou des mots de passe divulgués peuvent être utilisés pour accéder de manière non autorisée à des systèmes ou des données, ce qui peut entraîner leur altération ou leur corruption.
- **Perte de disponibilité** : dans certains cas, l'utilisation non autorisée de secrets ou de mots de passe divulgués peut provoquer un déni d'accès aux utilisateurs légitimes, conduisant à une perte de disponibilité des systèmes ou des données affectés.

- **Atteinte à la réputation** : une entreprise ou un individu peut subir un préjudice réputationnel s'il est révélé qu'il n'a pas été en mesure de protéger correctement des informations sensibles.
- **Conséquences juridiques** : selon la nature des informations divulguées et la législation en vigueur dans la juridiction concernée, la fuite de secrets ou de mots de passe peut donner lieu à des conséquences juridiques.
- **Surfacturation** : la fuite de clés d'API peut permettre à des personnes non autorisées d'accéder à des ressources d'API et d'effectuer des actions facturables, entraînant une surfacturation.

Détails techniques

Résultat #1

Secret Detected: Google OAuth Access Token

File: `https://android.googleapis.com:443//auth`

Detected Secret: `ya29.m`

Résultat #2

Secret Detected: Generic Username/Password

File: `https://github.com:443//satishpatnayak/MyTest/blob/master/AndroGoatInvoice.txt`

Detected Secret: `Login": "satishpatnayak`

Résultat #3

Secret Detected: Google Cloud API Key

File: `https://www.googleapis.com:443//experimentsandconfigs/v1/getExperimentsAndConfigs?r=1&c=1`

Detected Secret: `AIzaSyAJ_AuopgbxIeQNXUJKLW5CluuGX7ujD1o`

API	Valid	Cost	Check
Staticmap	✗	\$2/1.000 reqs	curl https://maps.googleapis.com/maps/api/staticmap?center=45%2C10&zoom=7&size=400x400&key=AlzaSyAJ_AuopgbxleQNXUJKLW5CluuGX7ujD1o
maps	✗	\$7/1.000 reqs	curl https://maps.googleapis.com/maps/api/js?key=AlzaSyAJ_AuopgbxleQNXUJKLW5CluuGX7ujD1o
Streetview	✗	\$7/1.000 reqs	curl https://maps.googleapis.com/maps/api/streetview?size=400x400&location=40.720032,-73.988354&fov=90&heading=235&pitch=10&key=AlzaSyAJ_AuopgbxleQNXUJKLW5Clu
Directions	✗	\$5/1.000 reqs	curl https://maps.googleapis.com/maps/api/directions/json?origin=Disneyland&destination=Universal+Studios+Hollywood4&key=AlzaSyAJ_AuopgbxleQNXUJKLW5CluuGX7ujD1o
Geocode	✗	\$5/1.000 reqs	curl https://maps.googleapis.com/maps/api/geocode/json?latlng=40.30&key=AlzaSyAJ_AuopgbxleQNXUJKLW5CluuGX7ujD1o
Find Place From Text	✗	\$17/1.000 reqs	curl https://maps.googleapis.com/maps/api/place/findplacefromtext/json?input=Museum&inputtype=textquery&fields=formatted_address,name&key=AlzaSyAJ_AuopgbxleQNXUJKLW5CluuGX7ujD1o
Autocomplete	✗	\$2.83/1.000 reqs	curl https://maps.googleapis.com/maps/api/place/autocomplete/json?input=Bingh&types=(cities)&key=AlzaSyAJ_AuopgbxleQNXUJKLW5CluuGX7ujD1o
Elevation	✗	\$5/1.000 reqs	curl https://maps.googleapis.com/maps/api/elevation/json?locations=39.7391536,-104.9847034&key=AlzaSyAJ_AuopgbxleQNXUJKLW5CluuGX7ujD1o
Timezone	✗	\$5/1.000 reqs	curl https://maps.googleapis.com/maps/api/timezone/json?location=39.6034810,-119.6822510&timestamp=1331161200&key=AlzaSyAJ_AuopgbxleQNXUJKLW5CluuGX7ujD1o
Nearest Roads	✗	\$10/1.000 reqs	curl https://roads.googleapis.com/v1/nearestRoads?points=60.170880,24.942795 60.170879,24.942796 60.170877,24.942796&key=AlzaSyAJ_AuopgbxleQNXUJKLW5CluuGX7ujD
Geolocation	✗	\$5/1.000 reqs	curl https://www.googleapis.com/geolocation/v1/geolocate?key=AlzaSyAJ_AuopgbxleQNXUJKLW5CluuGX7ujD1o
Route to Traveled	✗	\$10/1.000 reqs	curl https://roads.googleapis.com/v1/snapToRoads?path=-35.27801,149.12958 -35.28032,149.12907&interpolate=true&key=AlzaSyAJ_AuopgbxleQNXUJKLW5CluuGX7ujD1o
Place Details	✗	\$17/1.000 reqs	curl https://maps.googleapis.com/maps/api/place/details/json?place_id=ChIJN1t_tDeuEmsRUsyG83frY4&fields=name,rating&key=AlzaSyAJ_AuopgbxleQNXUJKLW5CluuGX7ujD1o
Speed Limit- Roads	✗	\$20/1.000 reqs	curl https://roads.googleapis.com/v1/speedLimits?path=38.75807927603043,-9.03741754643809&key=AlzaSyAJ_AuopgbxleQNXUJKLW5CluuGX7ujD1o

API	Valid	Cost	Check
Nearby Search-Places	✗	\$32/1.000 reqs	curl https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=-33.8670522,151.1957362&radius=100&types=food&name=harbour&key=AlzaSyAJ_AuopgbxleQNXUJKLW5CluuGX7ujD1o
Text Search-Places	✗	\$2.83/1.000 reqs	curl https://maps.googleapis.com/maps/api/place/queryautocomplete/json?input=pizza+nearpar&key=AlzaSyAJ_AuopgbxleQNXUJKLW5CluuGX7ujD1o
Playable Locations	✗	\$10/1.000 reqs	curl https://playablelocations.googleapis.com/v3/samplePlayableLocations?key=AlzaSyAJ_AuopgbxleQNXUJKLW5CluuGX7ujD1o
Google Custom Search	✗	\$5/1.000 reqs	curl https://www.googleapis.com/customsearch/v1?key=AlzaSyAJ_AuopgbxleQNXUJKLW5CluuGX7ujD1o&cx=016017638605632953755:bggehv7mz6y&q=test
Firebase Cloud Messaging	✗	NA	curl -i -s -X POST --header "Authorization: key=AlzaSyAJ_AuopgbxleQNXUJKLW5CluuGX7ujD1o" --header "Content-Type: applicati" https://fcm.googleapis.com/fcm/send -d '{"registration_ids": ["1"]}'

Résultat #4

Secret Detected: Google Cloud API Key

File: <https://www.googleapis.com:443/experimentsandconfigs/v1/getExperimentsAndConfigs?r=1&c=1>

Detected Secret: [AIzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w](#)

API	Valid	Cost	Check
Staticmap	✗	\$2/1.000 reqs	curl https://maps.googleapis.com/maps/api/staticmap?center=45%2C10&zoom=7&size=400x400&key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w
maps	✗	\$7/1.000 reqs	curl https://maps.googleapis.com/maps/api/js?key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w
Streetview	✗	\$7/1.000 reqs	curl https://maps.googleapis.com/maps/api/streetview?size=400x400&location=40.720032,-73.988354&fov=90&heading=235&pitch=10&key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w
Directions	✗	\$5/1.000 reqs	curl https://maps.googleapis.com/maps/api/directions/json?origin=Disneyland&destination=Universal+Studios+Hollywood4&key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w
Geocode	✗	\$5/1.000 reqs	curl https://maps.googleapis.com/maps/api/geocode/json?latlng=40.30&key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w
Find Place From Text	✗	\$17/1.000 reqs	curl https://maps.googleapis.com/maps/api/place/findplacefromtext/json?input=Museum&inputtype=textquery&fields=formatted_address,name&key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w
Autocomplete	✗	\$2.83/1.000 reqs	curl https://maps.googleapis.com/maps/api/place/autocomplete/json?input=Bingh&types=(cities)&key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w
Elevation	✗	\$5/1.000 reqs	curl https://maps.googleapis.com/maps/api/elevation/json?locations=39.7391536,-104.9847034&key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w
Timezone	✗	\$5/1.000 reqs	curl https://maps.googleapis.com/maps/api/timezone/json?location=39.6034810,-119.6822510&timestamp=1331161200&key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w
Nearest Roads	✗	\$10/1.000 reqs	curl https://roads.googleapis.com/v1/nearestRoads?points=60.170880,24.942795 60.170879,24.942796 60.170877,24.942796&key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w
Geolocation	✗	\$5/1.000 reqs	curl https://www.googleapis.com/geolocation/v1/geolocate?key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w
Route to Traveled	✗	\$10/1.000 reqs	curl https://roads.googleapis.com/v1/snapToRoads?path=-35.27801,149.12958 -35.28032,149.12907&interpolate=true&key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w
Place Details	✗	\$17/1.000 reqs	curl https://maps.googleapis.com/maps/api/place/details/json?place_id=ChIJNlt_tDeuEmsRUspvG83frY4&fields=name,rating&key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w
Speed Limit- Roads	✗	\$20/1.000 reqs	curl https://roads.googleapis.com/v1/speedLimits?path=38.75807927603043,-9.03741754643809&key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w
Nearby Search-Places	✗	\$32/1.000 reqs	curl https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=-33.8670522,151.1957362&radius=100&types=food&name=harbour&key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w

API	Valid	Cost	Check
Text Search–Places	✗	\$2.83/1.000 reqs	curl https://maps.googleapis.com/maps/api/place/queryautocomplete/json?input=pizza+nearpar&key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w
Playable Locations	✗	\$10/1.000 reqs	curl https://playablelocations.googleapis.com/v3/samplePlayableLocations?key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w
Google Custom Search	✗	\$5/1.000 reqs	curl https://www.googleapis.com/customsearch/v1?key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w&cx=016017638605632953755:bgqehv7mz6y&q=test
Firebase Cloud Messaging	✗	NA	curl -i -s -X POST --header "Authorization: key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w" --header "Content-Type: applica" https://fcm.googleapis.com/fcm/send -d '{"registration_ids": ["1"]}'

Recommandation

Les données sensibles ne doivent jamais être embarquées dans l'application elle-même. Privilégiez des méthodes sécurisées pour chiffrer, stocker et récupérer les identifiants de vos services afin d'y accéder selon les besoins.

Pour prévenir tout risque de surfacturation, envisagez de mettre en place un API key pinning ou d'utiliser des APIs authentifiées pour les services susceptibles de générer des coûts d'utilisation élevés. L'API key pinning permet de restreindre l'utilisation d'une clé à une application spécifique en exigeant une signature cryptographique. Il peut être activé par le fournisseur de services (par ex., Google Maps).

Pour les clés susceptibles de permettre un accès non autorisé, il est important de restreindre les permissions et les rôles à des fonctions non critiques, ou d'exposer le service via une API authentifiée.

Pour renforcer davantage la sécurité, l'API doit également disposer de contrôles d'accès appropriés et d'un rate limiting, et les clés doivent faire l'objet d'une rotation régulière afin de prévenir toute utilisation non autorisée.

Références

- [API Security Best Practices](#)
- [Best practices for securely using API keys](#)
- [CWE-200: Data Exposure](#)
- [CWE-259: Use of Hard-coded Password](#)
- [What Is AWS Secrets Manager?](#)
- [\\$450k Google Cloud API Key Compromise](#)

POTENTIELLEMENT Mode de secours activé

CVSS v3: 3,9

L'application active le mode de secours.

Le mode de sauvegarde est une fonctionnalité d'Android permettant aux utilisateurs de sauvegarder et de restaurer leurs données et paramètres d'un appareil à un autre. Par défaut, Android effectue une sauvegarde complète des applications, y compris les fichiers privés stockés sur la partition `/data`. Lorsque cette fonctionnalité est activée, le **Backup Manager Service** envoie périodiquement les données et paramètres vers le compte Google Drive de l'utilisateur — données d'application, mots de passe Wi-Fi et autres réglages. Ces données peuvent ensuite être restaurées sur le même appareil ou sur un autre, à condition que l'utilisateur se connecte avec le même compte Google.

Voici les principales caractéristiques du Backup Manager Service sous Android :

1. *Sauvegarde automatique* : le Backup Manager Service sauvegarde automatiquement les données des applications à intervalles réguliers, généralement une fois par jour. Il peut également déclencher une sauvegarde lors d'événements spécifiques, comme le branchement de l'appareil à une source d'alimentation ou lorsque l'utilisateur lance manuellement une sauvegarde.
2. *Sauvegardes incrémentielles* : le service effectue des sauvegardes incrémentielles, c'est-à-dire qu'il ne sauvegarde que les données modifiées depuis la dernière sauvegarde. Cela réduit la taille des sauvegardes et accélère le processus.
3. *Sauvegardes chiffrées* : les sauvegardes créées par le Backup Manager Service sont chiffrées avec une clé unique à l'appareil de chaque utilisateur, garantissant que seul l'utilisateur — ou toute personne disposant de ses identifiants Google — peut accéder aux données de sauvegarde.
4. *Sauvegardes par application* : le Backup Manager Service permet aux applications de spécifier les données à sauvegarder et celles à exclure. Cela permet de réduire la taille des sauvegardes et de s'assurer que les données sensibles ne sont pas incluses.
5. *Fonctionnalité de restauration* : le Backup Manager Service offre également une fonctionnalité de restauration permettant aux utilisateurs de restaurer leurs données d'application sur un nouvel appareil ou un appareil réinitialisé aux paramètres d'usine, ou encore de restaurer des données sur un appareil existant après la désinstallation et la réinstallation d'une application.

Détails techniques

Résultat #1

The `allowBackup` attribute is set to `true` .

```
<application xmlns:android="http://schemas.android.com/apk/res/android" android:theme="@7F0C0005" android:label="@7F0B0036"
android:icon="@7F0A0000" android:debuggable="true" android:allowBackup="true" android:supportsRtl="true"
android:networkSecurityConfig="@7F0E0000" android:roundIcon="@7F0A0002">
```

Recommandation

Dans le fichier manifeste de votre application, définissez l'attribut `android:allowBackup` pour activer ou désactiver la sauvegarde. La valeur par défaut est true, mais pour expliciter votre intention, il est recommandé de définir explicitement cet attribut dans le manifeste de l'application, comme indiqué ci-dessous :

</ XML

```
<manifest>
  <application android:allowBackup="true">
  </application>
</manifest>
```

Si l'application contient des données sensibles que vous ne souhaitez pas voir sauvegardées ou restaurées, vous pouvez désactiver la sauvegarde en définissant l'attribut `android:allowBackup` sur false, comme indiqué ci-dessous :

</ XML

```
<manifest>
  <application android:allowBackup="false">
  </application>
</manifest>
```

Détails du vecteur CVSS

Vecteur CVSS V3 : CVSS : 3.0/AV:P/AC:L/PR:H/UI:N/S:U/C:H/I:N/A:N

Vecteur d'Attaque:

Réseau (N)

Adjacent (A)

Local (L)

Physique (P)

Complexité d'Attaque:

Faible (L)

Élevé (H)

Privilèges Requis:

Aucun(e) (N)

Faible (L)

Élevé (H)

Interaction Utilisateur:

Aucun(e) (N)

Requis (R)

Périmètre:

Modifié (C)

Non modifié (U)

Confidentialité:

Élevé (H)

Faible (L)

Aucun(e) (N)

Intégrité:

Élevé (H)

Faible (L)

Aucun(e) (N)

Disponibilité:

Élevé (H)

Faible (L)

Aucun(e) (N)

Références

- [Random Musings on the M Developer Preview: the Ugly \(Part Two\)](#)
- [DRD22. Do not cache sensitive information](#)

POTENTIELLEMENT Parcours de chemins de fichiers

La présence d'un path traversal dans l'application constitue une vulnérabilité permettant à un attaquant de manipuler une partie du chemin de fichier. Ce chemin modifié est ensuite utilisé par les API du système de fichiers sans validation ni vérification appropriées.

Une application peut permettre à un attaquant de naviguer dans le système de fichiers au-delà des limites prévues. Cela peut entraîner un accès non autorisé à des fichiers ou répertoires sensibles. En manipulant les chemins de fichiers, un attaquant peut contourner les contrôles d'accès et récupérer ou modifier des données critiques, telles que des fichiers de configuration, des identifiants utilisateur ou des documents confidentiels. Cette vulnérabilité représente une menace sérieuse, car elle permet à l'attaquant d'escalader ses privilèges, d'exécuter du code arbitraire ou de lancer d'autres attaques sur le système.

Restreint • Ostorlab

Page: 117 sur 190

DART

```
import 'package:file/local.dart';

void main() {
  var file = new LocalFileSystem();
  var f = file.file("../..../passwords.txt");
  f.copy("pass.txt");
}

// Or changing the root of the current running process:

import 'package:file/file.dart';
import 'package:file/chroot.dart';
import 'package:file/local.dart';
import 'package:path/path.dart' as path;

void main() {
  final String root = path.canonicalize("../..../");
  final FileSystem newRoot = new ChrootFileSystem(
    const LocalFileSystem(),
    root,
  );
};
```

SWIFT

```
import Foundation

func readSensitiveFile(fileURL: URL) -> String? {
  let fileManager = FileManager.default
  let fileContents = fileManager.contents(atPath: fileURL.path)

  return String(data: fileContents!, encoding: .utf8)
}

func main() {
  let userDirectory = FileManager.default.homeDirectoryForCurrentUser
  let userInput = "/path/to/user/input.txt"
  let fileURL = userDirectory.appendingPathComponent(userInput)

  if let contents = readSensitiveFile(fileURL: fileURL) {
    print("File contents: \(contents)")
  } else {
    print("Failed to read file.")
  }
}

main()
```

KOTLIN

```
import java.io.File

fun readFile(filePath: String): String {
  val file = File("/var/www/files/$filePath")
  return file.readText()
}

fun main() {
  println("Enter the file name:")
  val fileName = readLine()

  try {
    val content = readFile(fileName!!)
    println("File content: $content")
  } catch (e: Exception) {
    println("Error: ${e.message}")
  }
}
```

Call to filesystem access method using path from input method. Taint traced from non-reachable component: `owasp.sat.agoat.SQLInjectionActivity` found at `owasp.sat.agoat.SQLInjectionActivity`. From parameter `0` to sink `android.database.sqlite.SQLiteDatabase` `owasp.sat.agoat.SQLInjectionActivity.openOrCreateDatabase(java.lang.String, int, android.database.sqlite.SQLiteDatabase$CursorFactory)`. Taint trace:

```
at owasp.sat.agoat.SQLInjectionActivity.openOrCreateDatabase()
at owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()
at owasp.sat.agoat.SQLInjectionActivity.onCreate()
at owasp.sat.agoat.SQLInjectionActivity$onCreate$1.<init>()
at owasp.sat.agoat.SQLInjectionActivity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.SQLInjectionActivity.onCreate()
```

Method `owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()`:

```
public final void onClick(android.view.View p9)
{
    String v0_1 = new StringBuilder();
    v0_1.append(SELECT * FROM users WHERE username=');
    String v1_6 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_6, username);
    v0_1.append(v1_6.getText().toString());
    v0_1.append(' ');
    String v0_2 = v0_1.toString();
    try {
        owasp.sat.agoat.SQLInjectionActivity.access$setMDB$p(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
        android.database.Cursor v3_2 = owasp.sat.agoat.SQLInjectionActivity.access$getMDB$p(this.this$0);
    } catch (boolean v2_13) {
        android.database.Cursor v3_5 = new StringBuilder();
        v3_5.append(Error occurred when inserting data into database: );
        v3_5.append(v2_13.getMessage());
        android.util.Log.d(Error, v3_5.toString());
        android.database.Cursor v3_8 = this.this$0.getApplicationContext();
        StringBuilder v4_10 = new StringBuilder();
        v4_10.append(Data not saved: Error occurred - );
        v4_10.append(v2_13.getMessage());
        android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v3_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
    StringBuilder v4_5 = new StringBuilder();
    if (v3_3 == null) {
        String v5_13 = new StringBuilder();
        v5_13.append(User: ());
        android.widget.EditText v6_2 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v6_2, username);
        v5_13.append(v6_2.getText().toString());
        v5_13.append() not found;
        v4_5.append(v5_13.toString());
    } else {
        if (v3_3.getCount() > 0) {
            v3_3.moveToFirst();
            do {
                boolean v2_5 = new StringBuilder();
                v2_5.append(Username: ());
                v2_5.append(v3_3.getString(1));
                v2_5.append() password: ());
                v2_5.append(v3_3.getString(2));
                v2_5.append()
                v4_5.append(v2_5.toString());
            } while(v3_3.moveToNext());
        } else {
        }
    }
    android.widget.Toast.makeText(((android.content.Context) this.this$0), ((CharSequence) v4_5.toString()), 1).show();
    android.util.Log.e(Error, v4_5.toString());
    android.util.Log.v(Query, v0_2);
    return;
}
```

```
}
```

Method `owasp.sat.agoat.SQLInjectionActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p4)
{
    super.onCreate(p4);
    this.setContentView(2131296306);
    ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLInjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
    return;
}
```

Method `owasp.sat.agoat.SQLInjectionActivity$onCreate$1.<init>()` :

```
SQLInjectionActivity$onCreate$1(owasp.sat.agoat.SQLInjectionActivity p1, android.widget.EditText p2)
{
    this.this$0 = p1;
    this.$username = p2;
    return;
}
```

Method `owasp.sat.agoat.SQLInjectionActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p4)
{
    super.onCreate(p4);
    this.setContentView(2131296306);
    ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLInjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
    return;
}
```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p5)
{
    android.support.v7.app.AppCompatActivityDelegate v0 = this.getDelegate();
    v0.installViewFactory();
    v0.onCreate(p5);
    if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
        if (android.os.Build$VERSION.SDK_INT < 23) {
            this.setTheme(this.mThemeId);
        } else {
            this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
        }
    }
    super.onCreate(p5);
    return;
}
```

Method `owasp.sat.agoat.SQLInjectionActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p4)
{
    super.onCreate(p4);
    this.setContentView(2131296306);
    ((android.widget.Button) this.findViewById(2131165207)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.SQLInjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));
    return;
}
```

```
}
```

Résultat #2

Call to filesystem access method using path from input method. Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity` found at `owasp.sat.agoat.InsecureStorageSDCardActivity`. From parameter `0` to sink `void java.io.FileWriter.<init>(java.io.File)`. Taint trace:

```
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at java.io.FileWriter.write()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this,this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
```

```

        v0_1.append( Password -);
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

    );

    java.io.IOException v0_2 = v0_1.toString();
    android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
    kotlin.jvm.internal.Intrinsics.checkNotNull(v4_0, Environment.getExternalStorageDirectory());
    java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
    android.widget.Toast v2_4 = new StringBuilder();
    v2_4.append(userinfo );
    v2_4.append(v1_10);
    System.out.println(v2_4.toString());
    v1_10.setReadable(1);
    v1_10.setWritable(1);
    java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
    v3_5.write(v0_2);
    v3_5.close();
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
} catch (java.io.IOException v0_3) {
    v0_3.printStackTrace();
}
return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password -);
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

    );

    java.io.IOException v0_2 = v0_1.toString();
    android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
    kotlin.jvm.internal.Intrinsics.checkNotNull(v4_0, Environment.getExternalStorageDirectory());
    java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
    android.widget.Toast v2_4 = new StringBuilder();
    v2_4.append(userinfo );
    v2_4.append(v1_10);
    System.out.println(v2_4.toString());
    v1_10.setReadable(1);
    v1_10.setWritable(1);
    java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
    v3_5.write(v0_2);
    v3_5.close();
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
} catch (java.io.IOException v0_3) {
    v0_3.printStackTrace();
}
return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());

```

```

        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password -);
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296296);
    ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.<init>()` :

```

InsecureStorageSDCardActivity$onCreate$1(owasp.sat.agoat.InsecureStorageSDCardActivity p1, android.widget.EditText p2,
android.widget.EditText p3)
{
    this.this$0 = p1;
    this.$username = p2;
    this.$password = p3;
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296296);
    ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296296);
    ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}
```

Résultat #3

Call to filesystem access method using path from input method. Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageTempActivity` found at `owasp.sat.agoat.InsecureStorageTempActivity` . From parameter `0` to sink `void java.io.FileWriter.<init>(java.io.File)` . Taint trace:

```
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()
at java.io.FileWriter.write()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageTempActivity.onCreate()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.<init>()
at owasp.sat.agoat.InsecureStorageTempActivity.onCreate()
at owasp.sat.agoat.InsecureStorageTempActivity.onCreate()
```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p8)
{
    try {
        android.content.Context v2_0 = java.io.File.createTempFile(users, tmp, new java.io.File(this.this$0.getApplicationInfo().dataDir));
        v2_0.setReadable(1);
        v2_0.setWritable(1);
        CharSequence v3_1 = new java.io.FileWriter(v2_0);
        CharSequence v4_1 = new StringBuilder();
        v4_1.append(username is );
        String v5_1 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_1, username);
        v4_1.append(v5_1.getText().toString());
        v4_1.append(
    );
        v3_1.write(v4_1.toString());
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(password is );
        String v5_5 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_5, password);
        v4_4.append(v5_5.getText().toString());
        v4_4.append(
    );
        v3_1.write(v4_4.toString());
        v3_1.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (Exception v0_5) {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
        v0_5.printStackTrace();
    }
    return;
}
```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p8)
{
    try {
        android.content.Context v2_0 = java.io.File.createTempFile(users, tmp, new java.io.File(this.this$0.getApplicationInfo().dataDir));
```

```

        v2_0.setReadable(1);
        v2_0.setWritable(1);
        CharSequence v3_1 = new java.io.FileWriter(v2_0);
        CharSequence v4_1 = new StringBuilder();
        v4_1.append(username is );
        String v5_1 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_1, username);
        v4_1.append(v5_1.getText().toString());
        v4_1.append(

    );

        v3_1.write(v4_1.toString());
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(password is );
        String v5_5 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_5, password);
        v4_4.append(v5_5.getText().toString());
        v4_4.append(

    );

        v3_1.write(v4_4.toString());
        v3_1.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (Exception v0_5) {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
        v0_5.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p8)
{
    try {
        android.content.Context v2_0 = java.io.File.createTempFile(users, tmp, new java.io.File(this.this$0.getApplicationInfo().dataDir));
        v2_0.setReadable(1);
        v2_0.setWritable(1);
        CharSequence v3_1 = new java.io.FileWriter(v2_0);
        CharSequence v4_1 = new StringBuilder();
        v4_1.append(username is );
        String v5_1 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_1, username);
        v4_1.append(v5_1.getText().toString());
        v4_1.append(

    );

        v3_1.write(v4_1.toString());
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(password is );
        String v5_5 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_5, password);
        v4_4.append(v5_5.getText().toString());
        v4_4.append(

    );

        v3_1.write(v4_4.toString());
        v3_1.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (Exception v0_5) {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
        v0_5.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p8)
{
    try {
        android.content.Context v2_0 = java.io.File.createTempFile(users, tmp, new java.io.File(this.this$0.getApplicationInfo().dataDir));
        v2_0.setReadable(1);
        v2_0.setWritable(1);
        CharSequence v3_1 = new java.io.FileWriter(v2_0);
        CharSequence v4_1 = new StringBuilder();
        v4_1.append(username is );

```

```

        String v5_1 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v5_1, username);
        v4_1.append(v5_1.getText().toString());
        v4_1.append(

    );

    v3_1.write(v4_1.toString());
    CharSequence v4_4 = new StringBuilder();
    v4_4.append(password is );
    String v5_5 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v5_5, password);
    v4_4.append(v5_5.getText().toString());
    v4_4.append(

    );

    v3_1.write(v4_4.toString());
    v3_1.close();
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
} catch (Exception v0_5) {
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
    v0_5.printStackTrace();
}
return;
}

```

Method `owasp.sat.agoat.InsecureStorageTempActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296300);
    ((android.widget.Button) this.findViewById(2131165210)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.<init>()` :

```

InsecureStorageTempActivity$onCreate$1(owasp.sat.agoat.InsecureStorageTempActivity p1, android.widget.EditText p2, android.widget.EditText
p3)
{
    this.this$0 = p1;
    this.$username = p2;
    this.$password = p3;
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageTempActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296300);
    ((android.widget.Button) this.findViewById(2131165210)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageTempActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296300);
}

```

```

        ((android.widget.Button) this.findViewById(2131165210)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }

```

Résultat #4

Call to filesystem access method using path from input method. Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSQLiteActivity` found at `owasp.sat.agoat.InsecureStorageSQLiteActivity`. From parameter `0` to sink `android.database.sqlite.SQLiteDatabase` `owasp.sat.agoat.InsecureStorageSQLiteActivity.openOrCreateDatabase(java.lang.String, int, android.database.sqlite.SQLiteDatabase$CursorFactory)`. Taint trace:

```

at owasp.sat.agoat.InsecureStorageSQLiteActivity.openOrCreateDatabase()
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate()

```

Method `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    try {
        android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
        this.mDB = v0_6;
    } catch (android.widget.Button v0_0) {
        android.widget.EditText v1_1 = new StringBuilder();
        v1_1.append(Error occurred while creating database: );
        v1_1.append(v0_0.getMessage());
        android.util.Log.e(Error:, v1_1.toString());
        this.setContentView(2131296299);
        ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
        return;
    }
    if (v0_6 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
    this.setContentView(2131296299);
    ((android.widget.Button) this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    android.support.v7.app.AppCompatActivityDelegate v0 = this.getDelegate();
    v0.installViewFactory();
    v0.onCreate(p5);
    if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
        if (android.os.Build$VERSION.SDK_INT < 23) {
            this.setTheme(this.mThemeId);
        } else {
            this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
        }
    }
    super.onCreate(p5);
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    try {
        android.widget.Button v0_6 = this.openOrCreateDatabase(aGoat, 0, 0);
        this.mDB = v0_6;
    } catch (android.widget.Button v0_0) {
        android.widget.EditText v1_1 = new StringBuilder();
        v1_1.append(Error occurred while creating database: );
        v1_1.append(v0_0.getMessage());
        android.util.Log.e(Error:, v1_1.toString());
        this.setContentView(2131296299);
        ((android.widget.Button this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText this.findViewById(2131165369))),
((android.widget.EditText this.findViewById(2131165303))))));
        return;
    }
    if (v0_6 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    v0_6.execSQL(CREATE TABLE IF NOT EXISTS users (ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR));
    this.setContentView(2131296299);
    ((android.widget.Button this.findViewById(2131165206)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1(this, ((android.widget.EditText this.findViewById(2131165369))),
((android.widget.EditText this.findViewById(2131165303))))));
    return;
}

```

Recommandation

Confinement du chemin : Normaliser le chemin et vérifier s'il est bien contenu dans le répertoire de destination ou non.

DART

```

import 'package:file/local.dart';
import 'dart:io';

void main() {
    var fileSystem = LocalFileSystem();
    var currentDirectory = Directory.current.path;
    print(currentDirectory);
    var inputFile = File('$currentDirectory/../../passwords.txt');
    var outputFile = File('$currentDirectory/pass.txt');

    if (!isWithinDirectory(inputFile, currentDirectory) ||
        !isWithinDirectory(outputFile, currentDirectory)) {
        print("Chemin de fichier invalide");
        return;
    }

    inputFile.copy(outputFile.path)
        .then((_) => print("Fichier copié avec succès"))
        .catchError((error) => print("Erreur : $error"));
}

bool isWithinDirectory(FileSystemEntity file, String directoryPath) {
    var fileDirectory = Directory(file.parent.path);
    var specifiedDirectory = Directory(directoryPath);
    return fileDirectory.path == specifiedDirectory.path ||
        fileDirectory.path.startsWith('${specifiedDirectory.path}${Platform.pathSeparator}');
}

```

SWIFT

```

import Foundation

func readSensitiveFile(fileURL: URL) -> String? {
    let fileManager = FileManager.default

    // Vérifier que fileURL se trouve dans le répertoire autorisé

```

```

        if fileURL.pathComponents.contains("path") && fileURL.pathComponents.contains("to") && fileURL.pathComponents.contains("user") {
            let fileContents = fileManager.contents(atPath: fileURL.path)
            return String(data: fileContents!, encoding: .utf8)
        }

        return nil
    }

func main() {
    let userDirectory = FileManager.default.homeDirectoryForCurrentUser
    let userInput = "/path/to/user/input.txt"
    let fileURL = userDirectory.appendingPathComponent(userInput)

    if let contents = readSensitiveFile(fileURL: fileURL) {
        print("Contenu du fichier : \(contents)")
    } else {
        print("Échec de la lecture du fichier.")
    }
}

main()

```

KOTLIN

```

import java.io.File

fun readFile(filePath: String): String {
    val sanitizedFilePath = filePath.replace("../", "").replace("/", "")
    val file = File("/var/www/files/$sanitizedFilePath")
    return file.readText()
}

fun main() {
    println("Saisissez le nom du fichier :")
    val fileName = readLine()

    try {
        val content = fileName?.let { readFile(it) }
        println("Contenu du fichier : $content")
    } catch (e: Exception) {
        println("Erreur : ${e.message}")
    }
}

```

Utilisation de chemins absolus : Privilégier l'utilisation de chemins absolus plutôt que relatifs dans la mesure du possible. Grâce aux chemins absolus, l'application spécifie explicitement l'emplacement exact du fichier ou du répertoire, sans laisser de marge d'interprétation.

DART

```

import 'dart:io';
import 'package:path/path.dart' as path;

void main() {
    final absolutePath = '/path/to/file.txt';
    var file = File(absolutePath);

    // print('Nom du fichier : ${path.basename(dir.file.path)}');
    print('Nom du fichier : ${path.basename(file.path)}');
}

```

SWIFT

```

import Foundation

func main() {
    let absolutePath = "/path/to/file.txt"
    let fileManager = FileManager.default

    if fileManager.fileExists(atPath: absolutePath) {
        // Effectuer des opérations sur le fichier
        print("Le fichier existe à \(absolutePath)")
    } else {

```

```

        print("Fichier introuvable à \" + absolutePath + "\"")
    }
}

main()

```

KOTLIN

```

import java.io.File

fun main() {
    val absolutePath = "/path/to/file.txt"
    val file = File(absolutePath)

    if (file.exists()) {
        // Effectuer des opérations sur le fichier
        println("Le fichier existe à $absolutePath")
    } else {
        println("Fichier introuvable à $absolutePath")
    }
}

```

Références

- [Improper Limitation of a Pathname to a Restricted Directory](#)

POTENTIELLEMENT

Stockage non sécurisé des données de l'application

Stockage non sécurisé des données de l'application, permettant à d'autres applications de les lire et/ou de les altérer.

Une vulnérabilité de stockage non sécurisé dans les applications mobiles désigne un défaut de sécurité où des données sensibles, telles que des identifiants utilisateur, des informations personnelles ou d'autres données confidentielles, sont stockées sur l'appareil dans des emplacements world-readable. Cela peut exposer ces données à un accès non autorisé par d'autres applications présentes sur l'appareil ou par un attaquant disposant d'un accès physique à celui-ci.

Le stockage non sécurisé ne se limite pas à l'écriture dans des emplacements world-readable : il survient également lorsque l'application charge des données — telles que des fichiers de configuration — depuis des emplacements world-writable, permettant à un attaquant de modifier ces fichiers et d'altérer le fonctionnement de l'application.

KOTLIN

```

import android.os.Bundle
import android.os.Environment
import android.widget.Button
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import java.io.File
import java.io.FileOutputStream
import java.io.IOException
import java.io.OutputStreamWriter

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val loginButton: Button = findViewById(R.id.login_button)
        loginButton.setOnClickListener {
            // Processus d'authentification simulé
            val token = authenticate("username", "password") // logique d'authentification

            // Stockage du token dans un stockage public
            storeTokenInPublicStorage(token)
        }
    }

    private fun storeTokenInPublicStorage(token: String) {
        val filePath = "/sdcard/insecure_app/jwt_config.txt"
        val file = File(filePath)
        try {

```

```

        val outputStreamWriter = OutputStreamWriter(FileOutputStream(file))
        outputStreamWriter.write(token)
        outputStreamWriter.close()
    } catch (e: IOException) {
        e.printStackTrace()
    }
}
}
}

```

SWIFT

```

import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Effectuez toute configuration supplémentaire après le chargement de la vue.

        let loginButton = UIButton(type: .system)
        loginButton.setTitle("Login", for: .normal)
        loginButton.addTarget(self, action: #selector(loginButtonTapped), for: .touchUpInside)
        view.addSubview(loginButton)
        loginButton.translatesAutoresizingMaskIntoConstraints = false
        loginButton.centerXAnchor.constraint(equalTo: view.centerXAnchor).isActive = true
        loginButton.centerYAnchor.constraint(equalTo: view.centerYAnchor).isActive = true
    }

    @objc func loginButtonTapped() {
        // Processus d'authentification simulé
        let token = authenticate(username: "username", password: "password")

        // Stockage du token dans un stockage public
        storeTokenInPublicStorage(token: token)
    }

    private func storeTokenInPublicStorage(token: String) {
        let filePath = "/var/mobile/Media/insecure_app/jwt_config.txt"
        let fileURL = URL(fileURLWithPath: filePath)

        do {
            try token.write(to: fileURL, atomically: true, encoding: .utf8)
            print("Token stored successfully.")
        } catch {
            print("Failed to write token: \(error)")
        }
    }
}

```

FLUTTER

```

import 'dart:io';
import 'package:flutter/material.dart';

void main() {
    runApp(MyApp());
}

class MyApp extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return MaterialApp(
            home: MyHomePage(),
        );
    }
}

class MyHomePage extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: Text('Flutter Insecure Storage'),
            ),
            body: Center(

```

```

        child: ElevatedButton(
          onPressed: () {
            // Processus d'authentification simulé
            String token = authenticate("username", "password");

            // Stockage du token dans un stockage public
            storeTokenInPublicStorage(token);
          },
          child: Text('Login'),
        ),
      ),
    );
  }

  void storeTokenInPublicStorage(String token) {
    final directory = Directory('/sdcard/insecure_app/');
    directory.createSync(recursive: true);
    final file = File('${directory.path}/jwt_config.txt');
    try {
      file.writeAsStringSync(token);
      print('Token stored successfully.');
```

Détails techniques

Résultat #1

Call to filesystem access method using path from external storage (e.i `getExternalStorage()`) Taint traced from non-reachable component:

`owasp.sat.agoat.InsecureStorageSDCardActivity` found at `owasp.sat.agoat.InsecureStorageSDCardActivity`. From source `java.io.File`
`android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.File.<init>(java.lang.String)`. Taint trace:

```

at java.io.File.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at java.io.File.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
```

```

    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(:
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

    );

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(:
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

    );

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);

```

```

        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296296);
    ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

Résultat #2

Call to filesystem access method using path from external storage (e.i `getExternalStorage()`) Taint traced from non-reachable component:

`owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From source `java.io.File`
`android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.File.<init>(java.lang.String)` . Taint trace:

```

at java.io.File.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at java.io.File.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
}

```

```

    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);

```

```

        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Résultat #3

Call to filesystem access method using path from external storage (e.i `getExternalStorage()`) Taint traced from non-reachable component:

`owasp.sat.agoat.InsecureStorageSDCardActivity` found at `owasp.sat.agoat.InsecureStorageSDCardActivity`. From source `java.io.File`
`android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.File.<init>(java.lang.String)`. Taint trace:

```

at java.io.File.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;

```

```

        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password -);
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

    );

    java.io.IOException v0_2 = v0_1.toString();
    android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
    kotlin.jvm.internal.Intrinsics.checkNotNull(v4_0, Environment.getExternalStorageDirectory());
    java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
    android.widget.Toast v2_4 = new StringBuilder();
    v2_4.append(userinfo );
    v2_4.append(v1_10);
    System.out.println(v2_4.toString());
    v1_10.setReadable(1);
    v1_10.setWritable(1);
    java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
    v3_5.write(v0_2);
    v3_5.close();
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
} catch (java.io.IOException v0_3) {
    v0_3.printStackTrace();
}
return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296296);
    ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

Résultat #4

Call to filesystem access method using path from external storage (e.i `getexternalstorage()`) Taint traced from non-reachable component:

`owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From source `java.io.File`
`android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.File.<init>(java.lang.String)` . Taint trace:

```

at java.io.File.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(:
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password -);
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

    );

    java.io.IOException v0_2 = v0_1.toString();

```

```

        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Résultat #5

Call to filesystem access method using path from external storage (e.i `getexternalstorage()`) Taint traced from non-reachable component:

`owasp.sat.agoat.InsecureStorageSDCardActivity` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From source `java.io.File`
`android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.FileWriter.<init>(java.io.File)` . Taint trace:

```

at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

```
}
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);
        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296296);
    ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View.OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}
```

Résultat #6

Call to filesystem access method using path from external storage (e.i `getexternalstorage()`) Taint traced from non-reachable component:

`owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From source `java.io.File`
`android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.FileWriter.<init>(java.io.File)` . Taint trace:

```
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

```
}
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);
        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}
```

Résultat #7

Call to filesystem access method using path from external storage (e.i `getexternalstorage()`) Taint traced from non-reachable component:

`owasp.sat.agoat.InsecureStorageSDCardActivity` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` . From source `java.io.File`
`android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.FileWriter.<init>(java.io.File)` . Taint trace:

```
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(
```

```

);

java.io.IOException v0_2 = v0_1.toString();
android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
kotlin.jvm.internal.Intrinsics.checkNotNull(v4_0, Environment.getExternalStorageDirectory());
java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
android.widget.Toast v2_4 = new StringBuilder();
v2_4.append(userinfo );
v2_4.append(v1_10);
System.out.println(v2_4.toString());
v1_10.setReadable(1);
v1_10.setWritable(1);
java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
v3_5.write(v0_2);
v3_5.close();
android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
} catch (java.io.IOException v0_3) {
    v0_3.printStackTrace();
}
return;
}
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296296);
    ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

```
}
```

Résultat #8

Call to filesystem access method using path from external storage (e.i `getexternalstorage()`) Taint traced from non-reachable component:

`owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1` found at `owasp.sat.agoat.InsecureStorageSDCardActivity`. From source `java.io.File`
`android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.FileWriter.<init>(java.io.File)`. Taint trace:

```
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(
);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(
);
```

```

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Recommandation

- Utiliser un stockage sécurisé : stockez les données sensibles telles que les tokens d'authentification ou les identifiants de manière sécurisée. Utilisez les mécanismes de stockage sécurisé fournis par la plateforme, tels que Keychain pour iOS et SharedPreferences avec chiffrement pour Android.
- Chiffrement des données : chiffrez les données sensibles au repos à l'aide d'algorithmes de chiffrement robustes pour protéger les données stockées sur l'appareil de l'utilisateur.
- Éviter le codage en dur des informations sensibles : au lieu de coder en dur des informations sensibles telles que des noms d'utilisateur, des mots de passe ou des tokens directement dans votre code, envisagez les mécanismes de stockage sécurisé suggérés ci-dessus.
- Éviter les permissions de fichier trop permissives : assurez-vous que les permissions de fichier sont définies de manière appropriée pour restreindre l'accès aux fichiers et répertoires sensibles. Sur Android, par exemple, évitez d'utiliser `MODE_WORLD_READABLE` ou `MODE_WORLD_WRITEABLE` lors de la création de fichiers, car ces modes accordent un accès étendu en lecture ou en écriture à toutes les applications. Respectez toujours le principe du moindre privilège lors de la définition des permissions de fichier et limitez l'accès uniquement à ce qui est nécessaire au fonctionnement de l'application.
- Validation des entrées : lors du chargement de données depuis des emplacements de stockage publics, assurez-vous toujours de les valider et de les assainir.

KOTLIN

```

import android.content.Context
import android.os.Bundle
import android.widget.Button
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.security.crypto.EncryptedSharedPreferences
import androidx.security.crypto.MasterKeys
import java.io.IOException

class MainActivity : AppCompatActivity() {

    private lateinit var encryptedSharedPreferences: SharedPreferences

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Initialize encrypted shared preferences
        initializeEncryptedSharedPreferences()

        val loginButton: Button = findViewById(R.id.login_button)
        loginButton.setOnClickListener {
            // Dummy authentication process
            val token = authenticate("username", "password") // authentication logic

            // Store token securely in encrypted shared preferences
            storeTokenInSharedPreferences(token)
        }

        private fun initializeEncryptedSharedPreferences() {
            try {
                val masterKeyAlias = MasterKeys.getOrCreate(MasterKeys.AES256_GCM_SPEC)
                encryptedSharedPreferences = EncryptedSharedPreferences.create(
                    "secure_preferences",
                    masterKeyAlias,
                    applicationContext,

```

```

        EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,
        EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM
    )
} catch (e: IOException) {
    e.printStackTrace()
}
}

private fun storeTokenInSharedPreferences(token: String) {
    encryptedSharedPreferences.edit().putString("jwt_token", token).apply()
    Toast.makeText(this, "Token stored securely.", Toast.LENGTH_SHORT).show()
}
}

```

SWIFT

```

import UIKit
import Security

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.

        let loginButton = UIButton(type: .system)
        loginButton.setTitle("Login", for: .normal)
        loginButton.addTarget(self, action: #selector(loginButtonTapped), for: .touchUpInside)
        view.addSubview(loginButton)
        loginButton.translatesAutoresizingMaskIntoConstraints = false
        loginButton.centerXAnchor.constraint(equalTo: view.centerXAnchor).isActive = true
        loginButton.centerYAnchor.constraint(equalTo: view.centerYAnchor).isActive = true
    }

    @objc func loginButtonTapped() {
        // Dummy authentication process
        let token = authenticate(username: "username", password: "password")

        // Store token securely in Keychain
        storeTokenInKeychain(token: token)
    }

    private func storeTokenInKeychain(token: String) {
        let keychainQuery: [String: Any] = [
            kSecClass as String: kSecClassGenericPassword,
            kSecAttrAccount as String: "jwtToken",
            kSecValueData as String: token.data(using: .utf8)!,
            kSecAttrAccessible as String: kSecAttrAccessibleWhenUnlockedThisDeviceOnly
        ]

        let status = SecItemAdd(keychainQuery as CFDictionary, nil)
        if status == errSecSuccess {
            print("Token stored securely in Keychain.")
        } else {
            print("Failed to store token in Keychain: \(status)")
        }
    }
}

```

FLUTTER

```

import 'package:flutter/material.dart';
import 'package:flutter_secure_storage/flutter_secure_storage.dart';

void main() {
    runApp(MyApp());
}

class MyApp extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return MaterialApp(
            home: MyHomePage(),
        );
    }
}

```

```

class MyHomePage extends StatelessWidget {
  final storage = FlutterSecureStorage();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Flutter Secure Storage Example'),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () async {
            // Dummy authentication process
            String token = await authenticate("username", "password");

            // Store token securely
            await storeTokenInSecureStorage(token);
          },
          child: Text('Login'),
        ),
      ),
    );
  }

  Future<void> storeTokenInSecureStorage(String token) async {
    await storage.write(key: 'jwtToken', value: token);
    print('Token stored securely.');
```

Références

- [CWE-359: Exposure of Private Information \("Privacy Violation"\)](#)
- [OWASP Mobile Top 10](#)
- [CWE-200: Information Exposure](#)

POTENTIELLEMENT Source to Sink

Une méthode Sink dangereuse est appelée avec des paramètres potentiellement contrôlés par l'utilisateur ou une méthode Source non fiable.

Une méthode source ou un paramètre contrôlé par l'utilisateur est utilisé pour appeler une méthode sink.

Source désigne une entrée de données non fiable, pouvant provenir d'un utilisateur non fiable. Sink désigne une méthode dangereuse qui, si elle est accessible à un attaquant, peut être exploitée pour mener une attaque.

Les sources et les sinks doivent être examinés pour détecter des vulnérabilités telles que l'injection, la référence indirecte à un objet ou l'accès non autorisé aux données.

Détails techniques

Résultat #1

User controlled parameter is used call a sink method. Taint traced from non-reachable component: `owasp.sat.agoat.AccessControlIssue1Activity` found at `owasp.sat.agoat.AccessControlIssue1Activity`. From parameter `0` to sink `android.content.SharedPreferences`
`owasp.sat.agoat.AccessControlIssue1Activity.getSharedPreferences(java.lang.String, int)`. Taint trace:

```

at owasp.sat.agoat.AccessControlIssue1Activity.getSharedPreferences()
at owasp.sat.agoat.AccessControlIssue1Activity.createPIN()
at owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.onClick()
at owasp.sat.agoat.AccessControlIssue1Activity.onCreate()
at owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.<init>()
at owasp.sat.agoat.AccessControlIssue1Activity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.AccessControlIssue1Activity.onCreate()
```

Method `owasp.sat.agoat.AccessControlIssue1Activity.createPIN()` :

```

public final boolean createPIN(String p5)
{
    kotlin.jvm.internal.Intrinsics.checkNotNull(p5, pinValue);
```

```

        android.content.SharedPreferences$Editor v1_1 = this.getSharedPreferences(pinDetails, 0).edit();
        v1_1.putBoolean(pinSet, 1);
        v1_1.putString(pin, this.hashPIN(p5));
        return v1_1.commit();
    }

```

Method `owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p5)
{
    android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165335));
    android.widget.Toast v1_9 = this.this$0;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_2, pinValue);
    if (!v1_9.createPIN(v0_2.getText().toString())) {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN not created), 1).show();
    } else {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN created. Please Login), 1).show();
        this.this$0.verifyPINView();
    }
    return;
}

```

Method `owasp.sat.agoat.AccessControlIssue1Activity.onCreate()` :

```

protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    if (!this.verifyIfPinSet()) {
        this.setContentView(2131296284);
        ((android.widget.Button) this.findViewById(2131165334)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1(this)));
    } else {
        this.verifyPINView();
    }
    return;
}

```

Method `owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1.<init>()` :

```

AccessControlIssue1Activity$onCreate$1(owasp.sat.agoat.AccessControlIssue1Activity p1)
{
    this.this$0 = p1;
    return;
}

```

Method `owasp.sat.agoat.AccessControlIssue1Activity.onCreate()` :

```

protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    if (!this.verifyIfPinSet()) {
        this.setContentView(2131296284);
        ((android.widget.Button) this.findViewById(2131165334)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1(this)));
    } else {
        this.verifyPINView();
    }
    return;
}

```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p5)
{
    android.support.v7.app.AppCompatActivity v0 = this.getDelegate();
    v0.installViewFactory();
    v0.onCreate(p5);
    if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
        if (android.os.Build$VERSION.SDK_INT < 23) {
            this.setTheme(this.mThemeId);
        } else {
            this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
        }
    }
    super.onCreate(p5);
    return;
}
```

Method `owasp.sat.agoat.AccessControlIssue1Activity.onCreate()` :

```
protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    if (!this.verifyIfPinSet()) {
        this.setContentView(2131296284);
        ((android.widget.Button this.findViewById(2131165334)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControlIssue1Activity$onCreate$1(this)))));
    } else {
        this.verifyPINView();
    }
    return;
}
```

Résultat #2

User controlled parameter is used call a sink method. Taint traced from reachable component: `owasp.sat.agoat.AccessControl1ViewActivity` found at `owasp.sat.agoat.AccessControl1ViewActivity`. From parameter `0` to sink `void owasp.sat.agoat.AccessControl1ViewActivity.startService()`. Taint trace:

```
at owasp.sat.agoat.AccessControl1ViewActivity.startService()
at owasp.sat.agoat.AccessControl1ViewActivity$onCreate$1.onClick()
at owasp.sat.agoat.AccessControl1ViewActivity.onCreate()
at owasp.sat.agoat.AccessControl1ViewActivity$onCreate$1.<init>()
at owasp.sat.agoat.AccessControl1ViewActivity.onCreate()
at android.support.v4.app.BaseFragmentActivityApi16.onCreate()
at android.support.v4.app.FragmentActivity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.AccessControl1ViewActivity.onCreate()
```

Method `owasp.sat.agoat.AccessControl1ViewActivity.startService()` :

```
public final void startService()
{
    this.startService(new android.content.Intent(((android.content.Context) this), owasp.sat.agoat.DownloadInvoiceService));
    return;
}
```

Method `owasp.sat.agoat.AccessControl1ViewActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p2)
{
    this.this$0.startService();
    return;
}
```

Method `owasp.sat.agoat.AccessControl1ViewActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    this.setContentView(2131296283);
    ((android.widget.Button) this.findViewById(2131165257)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControl1ViewActivity$onCreate$1(this)));
    return;
}
```

Method `owasp.sat.agoat.AccessControl1ViewActivity$onCreate$1.<init>()` :

```
AccessControl1ViewActivity$onCreate$1(owasp.sat.agoat.AccessControl1ViewActivity p1)
{
    this.this$0 = p1;
    return;
}
```

Method `owasp.sat.agoat.AccessControl1ViewActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    this.setContentView(2131296283);
    ((android.widget.Button) this.findViewById(2131165257)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControl1ViewActivity$onCreate$1(this)));
    return;
}
```

Method `android.support.v4.app.FragmentActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p9)
{
    int[] v1_0 = 0;
    this.mFragments.attachHost(0);
    super.onCreate(p9);
    android.support.v4.app.FragmentActivity$NonConfigurationInstances v0_2 =
((android.support.v4.app.FragmentActivity$NonConfigurationInstances) this.getLastNonConfigurationInstance());
    if (v0_2 != null) {
        this.mFragments.restoreLoaderNonConfig(v0_2.loaders);
    }
    if (p9 != null) {
        android.os.Parcelable vtmp2 = p9.getParcelable(android.support:fragments);
        if (v0_2 != null) {
            v1_0 = v0_2.fragments;
        }
        this.mFragments.restoreAllState(vtmp2, v1_0);
        if (p9.containsKey(android.support:next_request_index)) {
            this.mNextCandidateRequestIndex = p9.getInt(android.support:next_request_index);
            int[] v1_4 = p9.getIntArray(android.support:request_indicies);
            String[] v3_4 = p9.getStringArray(android.support:request_fragment_who);
            if ((v1_4 != null) && (v3_4 != null) && (v1_4.length == v3_4.length)) {
                this.mPendingFragmentActivityResultResults = new android.support.v4.util.SparseArrayCompat(v1_4.length);
                int v4_3 = 0;
                while (v4_3 < v1_4.length) {
                    this.mPendingFragmentActivityResultResults.put(v1_4[v4_3], v3_4[v4_3]);
                    v4_3++;
                }
            } else {
                android.util.Log.w(FragmentActivity, Invalid requestCode mapping in savedInstanceState.);
            }
        }
    }
    if (this.mPendingFragmentActivityResultResults == null) {
        this.mPendingFragmentActivityResultResults = new android.support.v4.util.SparseArrayCompat();
    }
}
```

```

        this.mNextCandidateRequestIndex = 0;
    }
    this.mFragments.dispatchCreate();
    return;
}

```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    android.support.v7.app.AppCompatActivityDelegate v0 = this.getDelegate();
    v0.installViewFactory();
    v0.onCreate(p5);
    if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
        if (android.os.Build.VERSION.SDK_INT < 23) {
            this.setTheme(this.mThemeId);
        } else {
            this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
        }
    }
    super.onCreate(p5);
    return;
}

```

Method `owasp.sat.agoat.AccessControl1ViewActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    this setContentView(2131296283);
    ((android.widget.Button this.findViewById(2131165257)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.AccessControl1ViewActivity$onCreate$1(this)))));
    return;
}

```

Résultat #3

Input source is used to call sink method Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1` found at `owasp.sat.agoat.InsecureStorageSDCardActivity` .From source `java.io.File android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.FileWriter.<init>(java.io.File)` . Taint trace:

```

at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

```

```

);

java.io.IOException v0_2 = v0_1.toString();
android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
kotlin.jvm.internal.Intrinsics.checkNotNull(v4_0, Environment.getExternalStorageDirectory());
java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
android.widget.Toast v2_4 = new StringBuilder();
v2_4.append(userinfo );
v2_4.append(v1_10);
System.out.println(v2_4.toString());
v1_10.setReadable(1);
v1_10.setWritable(1);
java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
v3_5.write(v0_2);
v3_5.close();
android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
} catch (java.io.IOException v0_3) {
    v0_3.printStackTrace();
}
return;
}
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(:
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(:
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());

```

```

        v0_1.append( Password -);
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

    );

    java.io.IOException v0_2 = v0_1.toString();
    android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
    kotlin.jvm.internal.Intrinsics.checkNotNull(v4_0, Environment.getExternalStorageDirectory());
    java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
    android.widget.Toast v2_4 = new StringBuilder();
    v2_4.append(userinfo );
    v2_4.append(v1_10);
    System.out.println(v2_4.toString());
    v1_10.setReadable(1);
    v1_10.setWritable(1);
    java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
    v3_5.write(v0_2);
    v3_5.close();
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
} catch (java.io.IOException v0_3) {
    v0_3.printStackTrace();
}
return;
}

```

Résultat #4

User controlled parameter is used call a sink method. Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity` found at `owasp.sat.agoat.InsecureStorageSDCardActivity`. From parameter `0` to sink `void java.io.FileWriter.<init>(java.io.File)`. Taint trace:

```

at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at java.io.FileWriter.write()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(

Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password -);
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

    );

    java.io.IOException v0_2 = v0_1.toString();
    android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
    kotlin.jvm.internal.Intrinsics.checkNotNull(v4_0, Environment.getExternalStorageDirectory());
    java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
    android.widget.Toast v2_4 = new StringBuilder();
    v2_4.append(userinfo );
    v2_4.append(v1_10);
    System.out.println(v2_4.toString());
    v1_10.setReadable(1);
    v1_10.setWritable(1);
    java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
    v3_5.write(v0_2);
    v3_5.close();
}

```

```

        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(:
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

    );

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(:
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

    );

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
    }
}

```

```

        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this,this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(
);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this,this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296296);
    ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.<init>()` :

```

InsecureStorageSDCardActivity$onCreate$1(owasp.sat.agoat.InsecureStorageSDCardActivity p1, android.widget.EditText p2,
android.widget.EditText p3)
{

```

```

        this.this$0 = p1;
        this.$username = p2;
        this.$password = p3;
        return;
    }

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296296);
    ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296296);
    ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

Résultat #5

Input source is used to call sink method Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity` found at `owasp.sat.agoat.InsecureStorageSDCardActivity`. From source `java.io.File android.os.Environment.getExternalStorageDirectory()` to sink `void java.io.FileWriter.<init>(java.io.File)`. Taint trace:

```

at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
    }
}

```

```

        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this,this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(:
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

    );

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this,this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296296);
    ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

Résultat #6

Input source is used to call sink method Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1` found at `owasp.sat.agoat.InsecureStorageSDCardActivity`. From source `java.io.File android.os.Environment.getExternalStorageDirectory()` to sink `void`

java.io.FileWriter.<init>(java.io.File) .Tainttrace:

```
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
```

```

        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Résultat #7

User controlled parameter is used call a sink method. Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageTempActivity` found at `owasp.sat.agoat.InsecureStorageTempActivity`. From parameter `0` to sink `void java.io.FileWriter.<init>(java.io.File)`. Taint trace:

```

at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()
at java.io.FileWriter.write()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageTempActivity.onCreate()
at owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.<init>()
at owasp.sat.agoat.InsecureStorageTempActivity.onCreate()
at owasp.sat.agoat.InsecureStorageTempActivity.onCreate()

```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p8)
{
    try {
        android.content.Context v2_0 = java.io.File.createTempFile(users, tmp, new java.io.File(this.this$0.getApplicationInfo().dataDir));
        v2_0.setReadable(1);
        v2_0.setWritable(1);
        CharSequence v3_1 = new java.io.FileWriter(v2_0);
        CharSequence v4_1 = new StringBuilder();
        v4_1.append(username is );
        String v5_1 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_1, username);
        v4_1.append(v5_1.getText().toString());
        v4_1.append(
    );
        v3_1.write(v4_1.toString());
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(password is );
        String v5_5 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_5, password);
        v4_4.append(v5_5.getText().toString());
        v4_4.append(
    );
        v3_1.write(v4_4.toString());
        v3_1.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (Exception v0_5) {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
        v0_5.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p8)
{
    try {
        android.content.Context v2_0 = java.io.File.createTempFile(users, tmp, new java.io.File(this.this$0.getApplicationInfo().dataDir));
        v2_0.setReadable(1);
        v2_0.setWritable(1);
    }
}

```

```

        CharSequence v3_1 = new java.io.FileWriter(v2_0);
        CharSequence v4_1 = new StringBuilder();
        v4_1.append(username is );
        String v5_1 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v5_1, username);
        v4_1.append(v5_1.getText().toString());
        v4_1.append(

    );

    v3_1.write(v4_1.toString());
    CharSequence v4_4 = new StringBuilder();
    v4_4.append(password is );
    String v5_5 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v5_5, password);
    v4_4.append(v5_5.getText().toString());
    v4_4.append(

);

    v3_1.write(v4_4.toString());
    v3_1.close();
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
} catch (Exception v0_5) {
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
    v0_5.printStackTrace();
}
return;
}

```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p8)
{
    try {
        android.content.Context v2_0 = java.io.File.createTempFile(users, tmp, new java.io.File(this.this$0.getApplicationInfo().dataDir));
        v2_0.setReadable(1);
        v2_0.setWritable(1);
        CharSequence v3_1 = new java.io.FileWriter(v2_0);
        CharSequence v4_1 = new StringBuilder();
        v4_1.append(username is );
        String v5_1 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v5_1, username);
        v4_1.append(v5_1.getText().toString());
        v4_1.append(

    );

    v3_1.write(v4_1.toString());
    CharSequence v4_4 = new StringBuilder();
    v4_4.append(password is );
    String v5_5 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v5_5, password);
    v4_4.append(v5_5.getText().toString());
    v4_4.append(

);

    v3_1.write(v4_4.toString());
    v3_1.close();
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
} catch (Exception v0_5) {
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
    v0_5.printStackTrace();
}
return;
}

```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p8)
{
    try {
        android.content.Context v2_0 = java.io.File.createTempFile(users, tmp, new java.io.File(this.this$0.getApplicationInfo().dataDir));
        v2_0.setReadable(1);
        v2_0.setWritable(1);
        CharSequence v3_1 = new java.io.FileWriter(v2_0);
        CharSequence v4_1 = new StringBuilder();
        v4_1.append(username is );
        String v5_1 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v5_1, username);

```

```

        v4_1.append(v5_1.getText().toString());
        v4_1.append(
    );

    v3_1.write(v4_1.toString());
    CharSequence v4_4 = new StringBuilder();
    v4_4.append(password is );
    String v5_5 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v5_5, password);
    v4_4.append(v5_5.getText().toString());
    v4_4.append(
);

    v3_1.write(v4_4.toString());
    v3_1.close();
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
} catch (Exception v0_5) {
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
    v0_5.printStackTrace();
}
return;
}
}

```

Method `owasp.sat.agoat.InsecureStorageTempActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296300);
    ((android.widget.Button) this.findViewById(2131165210)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.<init>()` :

```

InsecureStorageTempActivity$onCreate$1(owasp.sat.agoat.InsecureStorageTempActivity p1, android.widget.EditText p2, android.widget.EditText
p3)
{
    this.this$0 = p1;
    this.$username = p2;
    this.$password = p3;
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageTempActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296300);
    ((android.widget.Button) this.findViewById(2131165210)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageTempActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296300);
    ((android.widget.Button) this.findViewById(2131165210)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),

```

```
((android.widget.EditText) this.findViewById(2131165303))));
    return;
}
```

Résultat #8

User controlled parameter is used call a sink method. Taint traced from non-reachable component:

`owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity` found at `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity`. From parameter `0` to sink `void java.io.InputStreamReader.<init>(java.io.InputStream)`. Taint trace:

```
at java.io.InputStreamReader.<init>()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()
at java.io.InputStreamReader.<init>()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.<init>()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p8)
{
    String v0_1 = new StringBuilder();
    v0_1.append(ping );
    StringBuffer v1_4 = this.$ip;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v1_4, ip);
    v0_1.append(v1_4.getText().toString());
    String v0_2 = v0_1.toString();
    StringBuffer v1_2 = new StringBuffer();
    try {
        android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);
        kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v2_1, Runtime.getRuntime().exec(ip1, null, null));
        v2_1.waitFor();
        String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();
        kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v4_3, reader.readLine());
    } catch (android.widget.TextView v2_2) {
        v2_2.printStackTrace();
        android.widget.TextView v2_3 = this.$result;
        kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v2_3, result);
        v2_3.setText(((CharSequence) v1_2.toString()));
        return;
    }
    while (v4_3 != null) {
        String v5_3 = new StringBuilder();
        v5_3.append(v4_3);
        v5_3.append(n);
        v1_2.append(v5_3.toString());
    }
    v2_3 = this.$result;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v2_3, result);
    v2_3.setText(((CharSequence) v1_2.toString()));
    return;
}
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p8)
{
    String v0_1 = new StringBuilder();
    v0_1.append(ping );
    StringBuffer v1_4 = this.$ip;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v1_4, ip);
    v0_1.append(v1_4.getText().toString());
    String v0_2 = v0_1.toString();
    StringBuffer v1_2 = new StringBuffer();
    try {
        android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);
```

```

        kotlin.jvm.internal.Intrinsics.checkNotNull(v2_1, Runtime.getRuntime().exec(ip1,null,null));
        v2_1.waitFor();
        String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();
        kotlin.jvm.internal.Intrinsics.checkNotNull(v4_3, reader.readLine());
    } catch (android.widget.TextView v2_2) {
        v2_2.printStackTrace();
        android.widget.TextView v2_3 = this.$result;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v2_3, result);
        v2_3.setText(((CharSequence) v1_2.toString()));
        return;
    }
    while (v4_3 != null) {
        String v5_3 = new StringBuilder();
        v5_3.append(v4_3);
        v5_3.append(n);
        v1_2.append(v5_3.toString());
    }
    v2_3 = this.$result;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v2_3, result);
    v2_3.setText(((CharSequence) v1_2.toString()));
    return;
}

```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p8)
{
    String v0_1 = new StringBuilder();
    v0_1.append(ping );
    StringBuffer v1_4 = this.$ip;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, ip);
    v0_1.append(v1_4.getText().toString());
    String v0_2 = v0_1.toString();
    StringBuffer v1_2 = new StringBuffer();
    try {
        android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);
        kotlin.jvm.internal.Intrinsics.checkNotNull(v2_1, Runtime.getRuntime().exec(ip1,null,null));
        v2_1.waitFor();
        String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();
        kotlin.jvm.internal.Intrinsics.checkNotNull(v4_3, reader.readLine());
    } catch (android.widget.TextView v2_2) {
        v2_2.printStackTrace();
        android.widget.TextView v2_3 = this.$result;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v2_3, result);
        v2_3.setText(((CharSequence) v1_2.toString()));
        return;
    }
    while (v4_3 != null) {
        String v5_3 = new StringBuilder();
        v5_3.append(v4_3);
        v5_3.append(n);
        v1_2.append(v5_3.toString());
    }
    v2_3 = this.$result;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v2_3, result);
    v2_3.setText(((CharSequence) v1_2.toString()));
    return;
}

```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p8)
{
    String v0_1 = new StringBuilder();
    v0_1.append(ping );
    StringBuffer v1_4 = this.$ip;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, ip);
    v0_1.append(v1_4.getText().toString());
    String v0_2 = v0_1.toString();
    StringBuffer v1_2 = new StringBuffer();
    try {
        android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);
        kotlin.jvm.internal.Intrinsics.checkNotNull(v2_1, Runtime.getRuntime().exec(ip1,null,null));
    }
}

```

```

        v2_1.waitFor();
        String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();
        kotlin.jvm.internal.Intrinsics.checkNotNull(v4_3, reader.readLine());
    } catch (android.widget.TextView v2_2) {
        v2_2.printStackTrace();
        android.widget.TextView v2_3 = this.$result;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v2_3, result);
        v2_3.setText(((CharSequence) v1_2.toString()));
        return;
    }
    while (v4_3 != null) {
        String v5_3 = new StringBuilder();
        v5_3.append(v4_3);
        v5_3.append(n);
        v1_2.append(v5_3.toString());
    }
    v2_3 = this.$result;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v2_3, result);
    v2_3.setText(((CharSequence) v1_2.toString()));
    return;
}

```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296292);
    ((android.widget.Button) this.findViewById(2131165317)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1(((android.widget.EditText) this.findViewById(2131165279)),
((android.widget.TextView) this.findViewById(2131165311)))));
    return;
}

```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.<init>()` :

```

InputValidationsOSCMDInjectionMain2Activity$onCreate$1(android.widget.EditText p1, android.widget.TextView p2)
{
    this.$ip = p1;
    this.$result = p2;
    return;
}

```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296292);
    ((android.widget.Button) this.findViewById(2131165317)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1(((android.widget.EditText) this.findViewById(2131165279)),
((android.widget.TextView) this.findViewById(2131165311)))));
    return;
}

```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296292);
    ((android.widget.Button) this.findViewById(2131165317)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1(((android.widget.EditText) this.findViewById(2131165279)),
((android.widget.TextView) this.findViewById(2131165311)))));
}

```

```
    return;  
}
```

Résultat #9

User controlled parameter is used call a sink method. Taint traced from non-reachable component:

`owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity` found at `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity`. From parameter `0` to sink `void java.io.InputStreamReader.<init>(java.io.InputStream)`. Taint trace:

```
at java.io.InputStreamReader.<init>()  
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()  
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()  
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()  
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()  
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.<init>()  
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()  
at owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p8)  
{  
    String v0_1 = new StringBuilder();  
    v0_1.append(ping );  
    StringBuffer v1_4 = this.$ip;  
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v1_4, ip);  
    v0_1.append(v1_4.getText().toString());  
    String v0_2 = v0_1.toString();  
    StringBuffer v1_2 = new StringBuffer();  
    try {  
        android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);  
        kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v2_1, Runtime.getRuntime().exec(ip1, null, null));  
        v2_1.waitFor();  
        String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();  
        kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v4_3, reader.readLine());  
    } catch (android.widget.TextView v2_2) {  
        v2_2.printStackTrace();  
        android.widget.TextView v2_3 = this.$result;  
        kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v2_3, result);  
        v2_3.setText(((CharSequence) v1_2.toString()));  
        return;  
    }  
    while (v4_3 != null) {  
        String v5_3 = new StringBuilder();  
        v5_3.append(v4_3);  
        v5_3.append(n);  
        v1_2.append(v5_3.toString());  
    }  
    v2_3 = this.$result;  
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v2_3, result);  
    v2_3.setText(((CharSequence) v1_2.toString()));  
    return;  
}
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p8)  
{  
    String v0_1 = new StringBuilder();  
    v0_1.append(ping );  
    StringBuffer v1_4 = this.$ip;  
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v1_4, ip);  
    v0_1.append(v1_4.getText().toString());  
    String v0_2 = v0_1.toString();  
    StringBuffer v1_2 = new StringBuffer();  
    try {  
        android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);  
        kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v2_1, Runtime.getRuntime().exec(ip1, null, null));  
        v2_1.waitFor();  
        String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();
```

```

        kotlin.jvm.internal.Intrinsics.checkNotNull(v4_3, reader.readLine());
    } catch (android.widget.TextView v2_2) {
        v2_2.printStackTrace();
        android.widget.TextView v2_3 = this.$result;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v2_3, result);
        v2_3.setText(((CharSequence) v1_2.toString()));
        return;
    }
    while (v4_3 != null) {
        String v5_3 = new StringBuilder();
        v5_3.append(v4_3);
        v5_3.append(n);
        v1_2.append(v5_3.toString());
    }
    v2_3 = this.$result;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v2_3, result);
    v2_3.setText(((CharSequence) v1_2.toString()));
    return;
}

```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p8)
{
    String v0_1 = new StringBuilder();
    v0_1.append(ping );
    StringBuffer v1_4 = this.$ip;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, ip);
    v0_1.append(v1_4.getText().toString());
    String v0_2 = v0_1.toString();
    StringBuffer v1_2 = new StringBuffer();
    try {
        android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);
        kotlin.jvm.internal.Intrinsics.checkNotNull(v2_1, Runtime.getRuntime().exec(ip1, null, null));
        v2_1.waitFor();
        String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();
        kotlin.jvm.internal.Intrinsics.checkNotNull(v4_3, reader.readLine());
    } catch (android.widget.TextView v2_2) {
        v2_2.printStackTrace();
        android.widget.TextView v2_3 = this.$result;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v2_3, result);
        v2_3.setText(((CharSequence) v1_2.toString()));
        return;
    }
    while (v4_3 != null) {
        String v5_3 = new StringBuilder();
        v5_3.append(v4_3);
        v5_3.append(n);
        v1_2.append(v5_3.toString());
    }
    v2_3 = this.$result;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v2_3, result);
    v2_3.setText(((CharSequence) v1_2.toString()));
    return;
}

```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296292);
    ((android.widget.Button) this.findViewById(2131165317)).setOnClickListener(((android.view.View$OnClickListener) new
    owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1(((android.widget.EditText) this.findViewById(2131165279)),
    ((android.widget.TextView) this.findViewById(2131165311)))));
    return;
}

```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.<init>()` :

```
InputValidationsOSCMDInjectionMain2Activity$onCreate$1(android.widget.EditText p1, android.widget.TextView p2)
{
    this.$ip = p1;
    this.$result = p2;
    return;
}
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()` :

```
protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296292);
    ((android.widget.Button) this.findViewById(2131165317)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1(((android.widget.EditText) this.findViewById(2131165279)),
((android.widget.TextView) this.findViewById(2131165311)))));
    return;
}
```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity.onCreate()` :

```
protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296292);
    ((android.widget.Button) this.findViewById(2131165317)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1(((android.widget.EditText) this.findViewById(2131165279)),
((android.widget.TextView) this.findViewById(2131165311)))));
    return;
}
```

Résultat #10

User controlled parameter is used call a sink method. Taint traced from non-reachable component: `owasp.sat.agoat.InputValidationsActivity` found at `owasp.sat.agoat.InputValidationsActivity`. From parameter `0` to sink `void` `owasp.sat.agoat.InputValidationsActivity.startActivity(android.content.Intent)`. Taint trace:

```
at owasp.sat.agoat.InputValidationsActivity.startActivity()
at owasp.sat.agoat.InputValidationsActivity$onCreate$2.onClick()
at owasp.sat.agoat.InputValidationsActivity.onCreate()
at owasp.sat.agoat.InputValidationsActivity$onCreate$2.<init>()
at owasp.sat.agoat.InputValidationsActivity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.InputValidationsActivity.onCreate()
```

Method `owasp.sat.agoat.InputValidationsActivity$onCreate$2.onClick()` :

```
public final void onClick(android.view.View p5)
{
    this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0), owasp.sat.agoat.SQLInjectionActivity));
    return;
}
```

Method `owasp.sat.agoat.InputValidationsActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    this.setContentView(2131296291);
}
```

```

        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.xssBtn)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$1(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.sqliButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$2(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.wbUrl)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$3(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.oscmdi)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$4(this)));
        return;
    }
}

```

Method `owasp.sat.agoat.InputValidationsActivity$onCreate$2.<init>()` :

```

InputValidationsActivity$onCreate$2(owasp.sat.agoat.InputValidationsActivity p1)
{
    this.this$0 = p1;
    return;
}

```

Method `owasp.sat.agoat.InputValidationsActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    this.setContentView(2131296291);
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.xssBtn)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$1(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.sqliButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$2(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.wbUrl)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$3(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.oscmdi)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$4(this)));
    return;
}

```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    android.support.v7.app.AppCompatActivityDelegate v0 = this.getDelegate();
    v0.installViewFactory();
    v0.onCreate(p5);
    if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
        if (android.os.Build$VERSION.SDK_INT < 23) {
            this.setTheme(this.mThemeId);
        } else {
            this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
        }
    }
    super.onCreate(p5);
    return;
}

```

Method `owasp.sat.agoat.InputValidationsActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    this setContentView(2131296291);
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.xssBtn)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$1(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.sqliButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$2(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.wbUrl)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$3(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.oscmdi)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$4(this)));
    return;
}

```

Résultat #11

User controlled parameter is used call a sink method. Taint traced from non-reachable component: `owasp.sat.agoat.InputValidationsActivity` found at `owasp.sat.agoat.InputValidationsActivity`. From parameter `0` to sink `void` `owasp.sat.agoat.InputValidationsActivity.startActivity(android.content.Intent)`. Taint trace:

```

at owasp.sat.agoat.InputValidationsActivity.startActivity()
at owasp.sat.agoat.InputValidationsActivity$onCreate$2.onClick()
at android.content.Intent.<init>()
at owasp.sat.agoat.InputValidationsActivity$onCreate$2.onClick()
at owasp.sat.agoat.InputValidationsActivity.onCreate()
at owasp.sat.agoat.InputValidationsActivity$onCreate$2.<init>()
at owasp.sat.agoat.InputValidationsActivity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.InputValidationsActivity.onCreate()

```

Method `owasp.sat.agoat.InputValidationsActivity$onCreate$2.onClick()` :

```

public final void onClick(android.view.View p5)
{
    this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0), owasp.sat.agoat.SQLInjectionActivity));
    return;
}

```

Method `owasp.sat.agoat.InputValidationsActivity$onCreate$2.onClick()` :

```

public final void onClick(android.view.View p5)
{
    this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0), owasp.sat.agoat.SQLInjectionActivity));
    return;
}

```

Method `owasp.sat.agoat.InputValidationsActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    this setContentView(2131296291);
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.xssBtn)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$1(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.sqliButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$2(this)));
}

```

```

        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.wbUrl)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$3(this)));
        ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.oscmdi)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$4(this)));
        return;
    }
}

```

Method `owasp.sat.agoat.InputValidationsActivity$onCreate$2.<init>()` :

```

InputValidationsActivity$onCreate$2(owasp.sat.agoat.InputValidationsActivity p1)
{
    this.this$0 = p1;
    return;
}

```

Method `owasp.sat.agoat.InputValidationsActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    this.setContentView(2131296291);
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.xssBtn)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$1(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.sqliButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$2(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.wbUrl)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$3(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.oscmdi)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$4(this)));
    return;
}

```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    android.support.v7.app.AppCompatActivity$Delegate v0 = this.getDelegate();
    v0.installViewFactory();
    v0.onCreate(p5);
    if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
        if (android.os.Build.VERSION.SDK_INT < 23) {
            this.setTheme(this.mThemeId);
        } else {
            this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
        }
    }
    super.onCreate(p5);
    return;
}

```

Method `owasp.sat.agoat.InputValidationsActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    this.setContentView(2131296291);
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.xssBtn)).setOnClickListener(((android.view.View$OnClickListener) new

```

```

owasp.sat.agoat.InputValidationsActivity$onCreate$1(this));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.sqliButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$2(this));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.wbUrl)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$3(this));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.oscmdi)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsActivity$onCreate$4(this));
    return;
}

```

Résultat #12

User controlled parameter is used call a sink method. Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageActivity` found at `owasp.sat.agoat.InsecureStorageActivity`. From parameter `0` to sink `void`
`owasp.sat.agoat.InsecureStorageActivity.startActivity(android.content.Intent)`. Taint trace:

```

at owasp.sat.agoat.InsecureStorageActivity.startActivity()
at owasp.sat.agoat.InsecureStorageActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageActivity.onCreate()
at owasp.sat.agoat.InsecureStorageActivity$onCreate$1.<init>()
at owasp.sat.agoat.InsecureStorageActivity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.InsecureStorageActivity.onCreate()

```

Method `owasp.sat.agoat.InsecureStorageActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p5)
{
    this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0),
owasp.sat.agoat.InsecureStorageSharedPrefs));
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    this.setContentView(2131296295);
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SPButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$1(this));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SP1Button)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$2(this));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SQLButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$3(this));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.tempButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$4(this));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SDcardButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$5(this));
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageActivity$onCreate$1.<init>()` :

```

InsecureStorageActivity$onCreate$1(owasp.sat.agoat.InsecureStorageActivity p1)
{
    this.this$0 = p1;
}

```

```

    return;
}

```

Method `owasp.sat.agoat.InsecureStorageActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    this setContentView(2131296295);
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SPButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$1(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SP1Button)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$2(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SQLButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$3(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.tempButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$4(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SDcardButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$5(this)));
    return;
}

```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    android.support.v7.app.AppCompatActivity$Delegate v0 = this.getDelegate();
    v0.installViewFactory();
    v0.onCreate(p5);
    if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
        if (android.os.Build$VERSION.SDK_INT < 23) {
            this.setTheme(this.mThemeId);
        } else {
            this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
        }
    }
    super.onCreate(p5);
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    this setContentView(2131296295);
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SPButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$1(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SP1Button)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$2(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SQLButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$3(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.tempButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$4(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SDcardButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$5(this)));
    return;
}

```

```
}
```

Résultat #13

User controlled parameter is used call a sink method. Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageActivity` found at `owasp.sat.agoat.InsecureStorageActivity`. From parameter `0` to sink `void` `owasp.sat.agoat.InsecureStorageActivity.startActivity(android.content.Intent)`. Taint trace:

```
at owasp.sat.agoat.InsecureStorageActivity.startActivity()
at owasp.sat.agoat.InsecureStorageActivity$onCreate$1.onClick()
at android.content.Intent.<init>()
at owasp.sat.agoat.InsecureStorageActivity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageActivity.onCreate()
at owasp.sat.agoat.InsecureStorageActivity$onCreate$1.<init>()
at owasp.sat.agoat.InsecureStorageActivity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.InsecureStorageActivity.onCreate()
```

Method `owasp.sat.agoat.InsecureStorageActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p5)
{
    this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0),
owasp.sat.agoat.InsecureStorageSharedPrefs));
    return;
}
```

Method `owasp.sat.agoat.InsecureStorageActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p5)
{
    this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0),
owasp.sat.agoat.InsecureStorageSharedPrefs));
    return;
}
```

Method `owasp.sat.agoat.InsecureStorageActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    this.setContentView(2131296295);
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SPButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$1(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SP1Button)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$2(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SQLButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$3(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.tempButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$4(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SDcardButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$5(this)));
    return;
}
```

Method `owasp.sat.agoat.InsecureStorageActivity$onCreate$1.<init>()` :

```

InsecureStorageActivity$onCreate$1(owasp.sat.agoat.InsecureStorageActivity p1)
{
    this.this$0 = p1;
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    this.setContentView(2131296295);
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SPButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$1(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SP1Button)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$2(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SQLButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$3(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.tempButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$4(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SDCardButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$5(this)));
    return;
}

```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    android.support.v7.app.AppCompatActivityDelegate v0 = this.getDelegate();
    v0.installViewFactory();
    v0.onCreate(p5);
    if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
        if (android.os.Build$VERSION.SDK_INT < 23) {
            this.setTheme(this.mThemeId);
        } else {
            this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
        }
    }
    super.onCreate(p5);
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    this.setContentView(2131296295);
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SPButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$1(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SP1Button)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$2(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.SQLButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$3(this)));
    ((android.widget.Button)
this._$_findCachedViewById(owasp.sat.agoat.R$id.tempButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$4(this)));
}

```

```

        ((android.widget.Button)
this._$findCachedViewById(owasp.sat.agoat.R$id.SDcardButton)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageActivity$onCreate$5(this)));
        return;
    }

```

Résultat #14

User controlled parameter is used call a sink method. Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSharedPrefs1Activity` found at `owasp.sat.agoat.InsecureStorageSharedPrefs1Activity`. From parameter `0` to sink `android.content.SharedPreferences` `owasp.sat.agoat.InsecureStorageSharedPrefs1Activity.getSharedPreferences(java.lang.String, int)`. Taint trace:

```

at owasp.sat.agoat.InsecureStorageSharedPrefs1Activity.getSharedPreferences()
at owasp.sat.agoat.InsecureStorageSharedPrefs1Activity.getScoreFromSP()
at owasp.sat.agoat.InsecureStorageSharedPrefs1Activity$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSharedPrefs1Activity.onCreate()
at owasp.sat.agoat.InsecureStorageSharedPrefs1Activity$onCreate$1.<init>()
at owasp.sat.agoat.InsecureStorageSharedPrefs1Activity.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.InsecureStorageSharedPrefs1Activity.onCreate()

```

Method `owasp.sat.agoat.InsecureStorageSharedPrefs1Activity.getScoreFromSP()` :

```

public final int getScoreFromSP()
{
    int v0 = 0;
    int v1 = 1;
    android.content.SharedPreferences v4 = this.getSharedPreferences(score, 0);
    if ((v4.getInt(score, 0) != 0) && (v4.getInt(level, 0) != 0)) {
        v0 = v4.getInt(score, 0);
        v1 = v4.getInt(level, 0);
    }
    String v2_1 = new StringBuilder();
    v2_1.append(Score is );
    v2_1.append(v0);
    v2_1.append( and Level is);
    v2_1.append(v1);
    System.out.println(v2_1.toString());
    return v0;
}

```

Method `owasp.sat.agoat.InsecureStorageSharedPrefs1Activity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p8)
{
    int v1 = 1;
    int v2_1 = this.this$0.CalculateScore(this.this$0.getScoreFromSP());
    if (!this.this$0.CheckScore(v2_1)) {
        owasp.sat.agoat.InsecureStorageSharedPrefs1Activity v3_1 = this.this$0.getApplicationContext();
        CharSequence v5_1 = new StringBuilder();
        v5_1.append(Score is );
        v5_1.append(v2_1);
        v5_1.append(. Try hard);
        android.widget.Toast.makeText(v3_1, ((CharSequence) v5_1.toString()), 0).show();
    } else {
        v1 = 2;
        owasp.sat.agoat.InsecureStorageSharedPrefs1Activity v3_3 = this.$ScoreButton;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v3_3, ScoreButton);
        v3_3.setClickable(0);
        owasp.sat.agoat.InsecureStorageSharedPrefs1Activity v3_4 = this.$ScoreButton;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v3_4, ScoreButton);
        v3_4.setText(((CharSequence) You won the game));
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Success), 1).show();
    }
    this.this$0.DisplayScore(v2_1, v1);
    this.this$0.UpdateScoreInSP(v2_1, v1);
    return;
}

```

```
}
```

Method `owasp.sat.agoat.InsecureStorageSharedPrefs1Activity.onCreate()` :

```
protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    this.setContentView(2131296298);
    android.widget.Button v0_3 = ((android.widget.Button) this.findViewById(2131165264));
    v0_3.setOnClickListener(((android.view.View$OnClickListener) new owasp.sat.agoat.InsecureStorageSharedPrefs1Activity$onCreate$1(this,
v0_3)));
    return;
}
```

Method `owasp.sat.agoat.InsecureStorageSharedPrefs1Activity$onCreate$1.<init>()` :

```
InsecureStorageSharedPrefs1Activity$onCreate$1(owasp.sat.agoat.InsecureStorageSharedPrefs1Activity p1, android.widget.Button p2)
{
    this.this$0 = p1;
    this.$ScoreButton = p2;
    return;
}
```

Method `owasp.sat.agoat.InsecureStorageSharedPrefs1Activity.onCreate()` :

```
protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    this.setContentView(2131296298);
    android.widget.Button v0_3 = ((android.widget.Button) this.findViewById(2131165264));
    v0_3.setOnClickListener(((android.view.View$OnClickListener) new owasp.sat.agoat.InsecureStorageSharedPrefs1Activity$onCreate$1(this,
v0_3)));
    return;
}
```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p5)
{
    android.support.v7.app.AppCompatActivityDelegate v0 = this.getDelegate();
    v0.installViewFactory();
    v0.onCreate(p5);
    if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
        if (android.os.Build.VERSION.SDK_INT < 23) {
            this.setTheme(this.mThemeId);
        } else {
            this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
        }
    }
    super.onCreate(p5);
    return;
}
```

Method `owasp.sat.agoat.InsecureStorageSharedPrefs1Activity.onCreate()` :

```
protected void onCreate(android.os.Bundle p3)
{
    super.onCreate(p3);
    this.setContentView(2131296298);
    android.widget.Button v0_3 = ((android.widget.Button) this.findViewById(2131165264));
```

```

        v0_3.setOnClickListener(((android.view.View$OnClickListener) new owasp.sat.agoat.InsecureStorageSharedPrefs$1Activity$onCreate$1(this,
v0_3)));
        return;
    }

```

Résultat #15

User controlled parameter is used call a sink method. Taint traced from non-reachable component: `owasp.sat.agoat.InsecureStorageSharedPrefs` found at `owasp.sat.agoat.InsecureStorageSharedPrefs`. From parameter `0` to sink `android.content.SharedPreferences` `owasp.sat.agoat.InsecureStorageSharedPrefs.getSharedPreferences(java.lang.String, int)`. Taint trace:

```

at owasp.sat.agoat.InsecureStorageSharedPrefs.getSharedPreferences()
at owasp.sat.agoat.InsecureStorageSharedPrefs$onCreate$1.onClick()
at owasp.sat.agoat.InsecureStorageSharedPrefs.onCreate()
at owasp.sat.agoat.InsecureStorageSharedPrefs$onCreate$1.<init>()
at owasp.sat.agoat.InsecureStorageSharedPrefs.onCreate()
at android.support.v7.app.AppCompatActivity.onCreate()
at owasp.sat.agoat.InsecureStorageSharedPrefs.onCreate()

```

Method `owasp.sat.agoat.InsecureStorageSharedPrefs$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p6)
{
    android.content.SharedPreferences$Editor v1_1 = this.this$0.getSharedPreferences(users, 0).edit();
    android.widget.Toast v2_13 = this.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v2_13, username);
    v1_1.putString(username, v2_13.getText().toString());
    android.widget.Toast v2_2 = this.$password;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v2_2, password);
    v1_1.putString(password, v2_2.getText().toString());
    if (!v1_1.commit()) {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
    } else {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSharedPrefs.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296297);
    ((android.widget.Button) this.findViewById(2131165205)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSharedPrefs$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSharedPrefs$onCreate$1.<init>()` :

```

InsecureStorageSharedPrefs$onCreate$1(owasp.sat.agoat.InsecureStorageSharedPrefs p1, android.widget.EditText p2, android.widget.EditText
p3)
{
    this.this$0 = p1;
    this.$username = p2;
    this.$password = p3;
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSharedPrefs.onCreate()` :

```
protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this setContentView(2131296297);
    ((android.widget.Button) this.findViewById(2131165205)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSharedPrefs$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}
```

Method `android.support.v7.app.AppCompatActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p5)
{
    android.support.v7.app.AppCompatActivity$Delegate v0 = this.getDelegate();
    v0.installViewFactory();
    v0.onCreate(p5);
    if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
        if (android.os.Build$VERSION.SDK_INT < 23) {
            this.setTheme(this.mThemeId);
        } else {
            this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
        }
    }
    super.onCreate(p5);
    return;
}
```

Method `owasp.sat.agoat.InsecureStorageSharedPrefs.onCreate()` :

```
protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this setContentView(2131296297);
    ((android.widget.Button) this.findViewById(2131165205)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSharedPrefs$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}
```

Recommandation

La recommandation varie selon la classe de vulnérabilité identifiée.

POTENTIELLEMENT

Services déclarés sans permissions

CVSSv3: 3,4

Les services déclarés ne sont pas protégés par des permissions globales.

`service` est un composant d'application chargé d'effectuer des actions en arrière-plan, sans interaction utilisateur. `service` peut également être utilisé pour exposer des fonctionnalités à d'autres applications. Cela correspond aux appels à `Context.bindService()` permettant d'établir une connexion avec le service et d'interagir avec lui.

Les services non protégés peuvent être invoqués par d'autres applications et potentiellement accéder à des informations sensibles ou effectuer des actions privilégiées.

Détails techniques

Résultat #1

Service exported without permissions.

```
<service xmlns:android="http://schemas.android.com/apk/res/android" android:name="owasp.sat.agoat.DownloadInvoiceService"
android:enabled="true" android:exported="true"/>
```

Recommandation

`service` peut exposer plusieurs méthodes à des composants externes. Il est possible de définir des permissions arbitraires pour chaque méthode à l'aide de la méthode `checkPermission`.

Il est également possible de séparer les services et de restreindre l'accès en imposant des permissions dans la balise `<service>` du manifeste.

</ XML

```
<permission android:name="co.ostorlab.custom_permission" android:label="custom_permission"
    android:protectionLevel="dangerous"></permission>

<service android:name="co.ostorlab.custom_service" android:permission="co.ostorlab.custom_permission">
<intent-filter>
    <action android:name="co.ostorlab.ACTION"/>
</intent-filter>
</service>
```

Le service peut appliquer des permissions sur des appels IPC individuels en appelant la méthode `checkCallingPermission` avant d'exécuter l'implémentation de cet appel.

Détails du vecteur CVSS

Vecteur CVSS V3 : CVSS:3.0/AV:L/AC:L/PR:H/UI:N/S:U/C:L/I:L/A:N

Vecteur d'Attaque:	Réseau (N) Adjacent (A) Local (L) Physique (P)
Complexité d'Attaque:	Faible (L) Élevé (H)
Privilèges Requis:	Aucun(e) (N) Faible (L) Élevé (H)
Interaction Utilisateur:	Aucun(e) (N) Requis (R)
Périmètre:	Modifié (C) Non modifié (U)
Confidentialité:	Élevé (H) Faible (L) Aucun(e) (N)
Intégrité:	Élevé (H) Faible (L) Aucun(e) (N)
Disponibilité:	Élevé (H) Faible (L) Aucun(e) (N)

Références

- [Service \(Android Developer Documentation\)](#)
- [CWE-280: Improper Handling of Insufficient Permissions or Privileges](#)
- [Security Decisions Via Untrusted Inputs \(OWASP Mobile Top 10\)](#)

POTENTIELLEMENT

Manipulation d'URL

Un attaquant peut contrôler l'URL utilisée par une application pour récupérer du contenu, ce qui constitue un risque de sécurité, car il peut manipuler ce contenu, entraînant l'exécution de code malveillant ou le vol d'informations sensibles.

Quand un attaquant prend le contrôle de l'URL utilisée par une application pour récupérer du contenu, cela constitue un risque de sécurité critique. En manipulant cette URL, l'attaquant peut rediriger l'application vers un serveur malveillant ou injecter son propre contenu dans la réponse. Cela peut entraîner diverses vulnérabilités et conséquences potentielles. Par exemple :

- Exécution de code : L'attaquant peut modifier l'URL pour la faire pointer vers un serveur sous son contrôle, lui permettant ainsi de livrer du code malveillant à l'application. Ce code peut exploiter des vulnérabilités dans l'application, exécuter des commandes arbitraires ou installer un malware sur le système de l'utilisateur.
- Vol de données : En redirigeant l'application vers un serveur frauduleux, l'attaquant peut leurrer les utilisateurs et les inciter à saisir des informations sensibles, telles que des identifiants, des coordonnées bancaires ou des données personnelles, qui seront ensuite capturées et exploitées.

- Manipulation de contenu : L'attaquant peut altérer le contenu récupéré par l'application, modifier les informations affichées ou injecter des scripts malveillants. Cela peut avoir diverses conséquences, telles que l'affichage d'informations trompeuses, le défilement de pages web ou des attaques par phishing.
- Attaques de la chaîne d'approvisionnement : Si l'application récupère du contenu depuis des sources externes, comme des bibliothèques ou des plugins, le contrôle de l'URL peut permettre à l'attaquant de remplacer des ressources légitimes par des versions compromises ou malveillantes. Cela peut compromettre la sécurité de l'ensemble du système.

Voici un exemple de code d'une application récupérant une entrée contrôlée par l'attaquant :

DART

```
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

class MainActivity extends StatefulWidget {
  final String url;

  MainActivity({required this.url});

  @override
  _MainActivityState createState() => _MainActivityState();
}

class _MainActivityState extends State<MainActivity> {
  String? content;

  @override
  void initState() {
    super.initState();
    fetchContent(widget.url);
  }

  Future<void> fetchContent(String url) async {
    try {
      final response = await http.get(Uri.parse(url));

      if (response.statusCode == 200) {
        setState(() {
          content = response.body;
        });
        // TODO: Process the fetched content here
      } else {
        // Handle error response
      }
    } catch (e) {
      // Handle network or API call error
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Fetching URL Content'),
      ),
      body: Center(
        child: content != null
          ? Text(content!)
          : CircularProgressIndicator(),
      ),
    );
  }
}
```

SWIFT

```
import UIKit

class AppDelegate: UIResponder, UIApplicationDelegate {

  var window: UIWindow?

  func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    // Handle deep link URL if the app was launched from a URL
    if let url = launchOptions?[.url] as? URL {
      handleDeepLink(url)
    }
  }
}
```

```

    }
    return true
}

func application(_ application: UIApplication, open url: URL, options: [UIApplication.OpenURLOptionsKey : Any] = [:]) -> Bool {
    // Handle deep link URL when the app is already running
    handleDeepLink(url)
    return true
}

private func handleDeepLink(_ url: URL) {
    // Check if the URL scheme is "ostorlab"
    if url.scheme == "ostorlab" {
        // Extract the URL from the deep link
        if let deepLinkURL = URL(string: url.absoluteString.replacingOccurrences(of: "ostorlab://", with: "")) {
            // Fetch the content from the provided URL
            fetchContent(deepLinkURL)
        }
    }
}

private func fetchContent(_ url: URL) {
    URLSession.shared.dataTask(with: url) { (data, response, error) in
        if let error = error {
            print("Error fetching content: \(error.localizedDescription)")
            return
        }
        if let data = data, let content = String(data: data, encoding: .utf8) {
            // Process the fetched content here
            print("Fetched content: \(content)")
        }
    }.resume()
}
}

```

KOTLIN

```

import android.content.Intent
import android.net.Uri
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch
import java.io.BufferedReader
import java.io.InputStreamReader
import java.net.HttpURLConnection
import java.net.URL

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Get the URL from the intent
        val intent: Intent = intent
        val url: String? = intent.getStringExtra("url")

        if (url != null) {
            // Fetch the content from the provided URL
            GlobalScope.launch(Dispatchers.IO) {
                val content: String? = fetchContent(url)
                // Process the fetched content as needed
                if (content != null) {
                    // TODO: Process the fetched content here
                }
            }
        }
    }

    private fun fetchContent(urlString: String): String? {
        var connection: HttpURLConnection? = null
        var reader: BufferedReader? = null

        try {
            val url = URL(urlString)
            connection = url.openConnection() as HttpURLConnection
            connection.requestMethod = "GET"

```

```

        // Check if the response code is successful
        if (connection.responseCode == HttpURLConnection.HTTP_OK) {
            reader = BufferedReader(InputStreamReader(connection.inputStream))
            val response = StringBuilder()
            var line: String?

            // Read the response line by line
            while (reader.readLine().also { line = it } != null) {
                response.append(line)
            }

            return response.toString()
        }
    } catch (e: Exception) {
        e.printStackTrace()
    } finally {
        // Close the connection and reader
        connection?.disconnect()
        reader?.close()
    }

    return null
}
}

```

Détails techniques

Résultat #1

The parameter at position: **0** of the method **loadUrl** in the class **WebView** is controlled by an attacker and can be used to retrieve malicious content. Taint traced from non-reachable component: **owasp.sat.agoat.InputValidationsWebViewURLActivity**. The TaintElement is coming from: Class

InputValidationsWebViewURLActivity Method **onCreate** Position **0** Signature **(Landroid/os/Bundle;)V**. Call trace:

```

at android.webkit.WebView.loadUrl
at owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.onClick
at owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate
at owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.<init>
at owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate
at owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate

```

at **android.webkit.WebView.loadUrl**

Unable to extract source code.

at **owasp.sat.agoat.InputValidationsWebViewURLActivity\$onCreate\$1.onClick**

```

public final void onClick(android.view.View p4)
{
    android.webkit.WebSettings v0_1 = this.$webview11.getSettings();
    kotlin.jvm.internal.Intrinsics.checkNotNull(v0_1, webSettings1);
    v0_1.setJavaScriptEnabled(1);
    String v1_4 = this.$url;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, url);
    this.$webview11.loadUrl(v1_4.getText().toString());
    return;
}

```

at **owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate**

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296293);
    android.widget.Button v0_3 = ((android.widget.Button) this.findViewById(2131165288));
    android.webkit.WebView v1_2 = this.findViewById(2131165374);
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_2, findViewById(R.id.webview1));
    v0_3.setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1(((android.webkit.WebView) v1_2), ((android.widget.TextView)
this.findViewById(2131165367)))));
}

```

```

    return;
}

```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.<init>`

```

InputValidationsWebViewURLActivity$onCreate$1(android.webkit.WebView p1, android.widget.TextView p2)
{
    this.$webview11 = p1;
    this.$url = p2;
    return;
}

```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate`

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296293);
    android.widget.Button v0_3 = ((android.widget.Button) this.findViewById(2131165288));
    android.webkit.WebView v1_2 = this.findViewById(2131165374);
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, findViewById(R.id.webview1));
    v0_3.setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1(((android.webkit.WebView) v1_2), ((android.widget.TextView)
this.findViewById(2131165367))))));
    return;
}

```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate`

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296293);
    android.widget.Button v0_3 = ((android.widget.Button) this.findViewById(2131165288));
    android.webkit.WebView v1_2 = this.findViewById(2131165374);
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_2, findViewById(R.id.webview1));
    v0_3.setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1(((android.webkit.WebView) v1_2), ((android.widget.TextView)
this.findViewById(2131165367))))));
    return;
}

```

Résultat #2

The parameter at position: `0` of the method `loadUrl` in the class `WebView` is controlled by an attacker and can be used to retrieve malicious content. Taint traced from non-reachable component: `owasp.sat.agoat.InputValidationsWebViewURLActivity`. The TaintElement is coming from: Class

`InputValidationsWebViewURLActivity` Method `onCreate` Position `0` Signature `(Landroid/os/Bundle;)V`. Call trace:

```

at android.webkit.WebView.loadUrl
at owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.onClick
at owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.onClick
at owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate
at owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.<init>
at owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate
at owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate

```

at `android.webkit.WebView.loadUrl`

Unable to extract source code.

at `owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.onClick`

```

public final void onClick(android.view.View p4)
{

```

```

        android.webkit.WebSettings v0_1 = this.$webview11.getSettings();
        kotlin.jvm.internal.Intrinsics.checkNotNull(v0_1, webSettings1);
        v0_1.setJavaScriptEnabled(1);
        String v1_4 = this.$url;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, url);
        this.$webview11.loadUrl(v1_4.getText().toString());
        return;
    }

```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.onClick`

```

    public final void onClick(android.view.View p4)
    {
        android.webkit.WebSettings v0_1 = this.$webview11.getSettings();
        kotlin.jvm.internal.Intrinsics.checkNotNull(v0_1, webSettings1);
        v0_1.setJavaScriptEnabled(1);
        String v1_4 = this.$url;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, url);
        this.$webview11.loadUrl(v1_4.getText().toString());
        return;
    }

```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate`

```

    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296293);
        android.widget.Button v0_3 = ((android.widget.Button) this.findViewById(2131165288));
        android.webkit.WebView v1_2 = this.findViewById(2131165374);
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_2, findViewById(R.id.webview1));
        v0_3.setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1(((android.webkit.WebView) v1_2), ((android.widget.TextView)
this.findViewById(2131165367))))));
        return;
    }

```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.<init>`

```

    InputValidationsWebViewURLActivity$onCreate$1(android.webkit.WebView p1, android.widget.TextView p2)
    {
        this.$webview11 = p1;
        this.$url = p2;
        return;
    }

```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate`

```

    protected void onCreate(android.os.Bundle p5)
    {
        super.onCreate(p5);
        this.setContentView(2131296293);
        android.widget.Button v0_3 = ((android.widget.Button) this.findViewById(2131165288));
        android.webkit.WebView v1_2 = this.findViewById(2131165374);
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_2, findViewById(R.id.webview1));
        v0_3.setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1(((android.webkit.WebView) v1_2), ((android.widget.TextView)
this.findViewById(2131165367))))));
        return;
    }

```

at `owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate`

```

    protected void onCreate(android.os.Bundle p5)
    {

```

```

        super.onCreate(p5);
        this setContentView(2131296293);
        android.widget.Button v0_3 = ((android.widget.Button) this.findViewById(2131165288));
        android.webkit.WebView v1_2 = this.findViewById(2131165374);
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_2, findViewById(R.id.webview1));
        v0_3.setOnClickListener(((android.view.View$OnClickListener) new
owaasp.sat.agoot.InputValidationsWebViewURLActivity$onCreate$1(((android.webkit.WebView) v1_2), ((android.widget.TextView)
this.findViewById(2131165367))))));
        return;
    }

```

Recommandation

Il est essentiel d'adopter une approche proactive pour protéger vos applications contre les attaques par manipulation d'URL. Voici les recommandations à appliquer :

- Validation des entrées : mettez en place des mécanismes stricts de validation des entrées afin de garantir que les URL utilisées pour la récupération de contenu sont correctement formatées et conformes aux schémas attendus. Cela peut inclure la vérification des schémas d'URL valides, l'application des noms de domaine attendus et la validation des paramètres de requête.
- Approche par liste blanche : maintenez une liste blanche de domaines ou de sources de confiance à partir desquels votre application récupère du contenu. N'autorisez les requêtes que vers ces sources approuvées afin de minimiser le risque d'accès à du contenu malveillant ou non autorisé.
- Vérification de l'intégrité du contenu : mettez en place des mécanismes permettant de vérifier l'intégrité du contenu récupéré. Calculez et comparez les empreintes cryptographiques ou les signatures numériques du contenu reçu avec les valeurs attendues pour détecter toute modification ou altération.

En appliquant ces recommandations, vous renforcez la sécurité de vos applications, protégez les données utilisateur et atténuez les risques liés aux attaques par manipulation d'URL.

Voici un exemple de code pour mettre en œuvre une validation par liste blanche :

DART

```

import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

class MainActivity extends StatefulWidget {
  final String url;
  final List<String> allowedDomains;

  MainActivity({required this.url, required this.allowedDomains});

  @override
  _MainActivityState createState() => _MainActivityState();
}

class _MainActivityState extends State<MainActivity> {
  String? content;

  @override
  void initState() {
    super.initState();
    if (isDomainAllowed(widget.url)) {
      fetchContent(widget.url);
    } else {
      // Handle unauthorized domain
      // TODO: Implement appropriate action for unauthorized domain
    }
  }

  bool isDomainAllowed(String urlString) {
    Uri uri = Uri.parse(urlString);
    String domain = uri.host ?? '';
    return widget.allowedDomains.contains(domain);
  }

  Future<void> fetchContent(String url) async {
    try {
      final response = await http.get(Uri.parse(url));

      if (response.statusCode == 200) {
        setState(() {
          content = response.body;
        });
        // TODO: Process the fetched content here
      } else {
        // Handle error response
      }
    }
  }

```

```

    } catch (e) {
        // Handle network or API call error
    }
}

@override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: Text('Fetching URL Content'),
        ),
        body: Center(
            child: content != null
                ? Text(content!)
                : CircularProgressIndicator(),
        ),
    );
}
}

```

SWIFT

```

import UIKit

class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    let allowedDomains = ["example.com", "trusteddomain.com"] // Add your trusted domain names here

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Handle deep link URL if the app was launched from a URL
        if let url = launchOptions?["url"] as? URL {
            handleDeepLink(url)
        }
        return true
    }

    func application(_ application: UIApplication, open url: URL, options: [UIApplication.OpenURLOptionsKey : Any] = [:]) -> Bool {
        // Handle deep link URL when the app is already running
        handleDeepLink(url)
        return true
    }

    private func handleDeepLink(_ url: URL) {
        // Check if the URL scheme is "ostorlab"
        if url.scheme == "ostorlab" {
            // Extract the URL from the deep link
            if let deepLinkURL = URL(string: url.absoluteString.replacingOccurrences(of: "ostorlab://", with: "")) {
                // Verify the domain name against the whitelist
                if isDomainAllowed(deepLinkURL) {
                    // Fetch the content from the provided URL
                    fetchContent(deepLinkURL)
                } else {
                    // Handle unauthorized domain
                    // TODO: Implement appropriate action for unauthorized domain
                }
            }
        }
    }

    private func isDomainAllowed(_ url: URL) -> Bool {
        guard let host = url.host else {
            return false
        }
        return allowedDomains.contains(host)
    }

    private func fetchContent(_ url: URL) {
        URLSession.shared.dataTask(with: url) { (data, response, error) in
            if let error = error {
                print("Error fetching content: \(error.localizedDescription)")
                return
            }
            if let data = data, let content = String(data: data, encoding: .utf8) {
                // Process the fetched content here
                print("Fetched content: \(content)")
            }
        }
    }
}

```

```

    }.resume()
}
}

```

KOTLIN

```

import android.content.Intent
import android.net.Uri
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch
import java.io.BufferedReader
import java.io.InputStreamReader
import java.net.HttpURLConnection
import java.net.URL

class MainActivity : AppCompatActivity() {
    private val allowedDomains = listOf("example.com", "trusteddomain.com") // Add your trusted domain names here

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Get the URL from the intent
        val intent: Intent = intent
        val url: String? = intent.getStringExtra("url")

        if (url != null) {
            // Verify the domain name against the whitelist
            if (isDomainAllowed(url)) {
                // Fetch the content from the provided URL
                GlobalScope.launch(Dispatchers.IO) {
                    val content: String? = fetchContent(url)
                    // Process the fetched content as needed
                    if (content != null) {
                        // TODO: Process the fetched content here
                    }
                }
            } else {
                // Handle unauthorized domain
                // TODO: Implement appropriate action for unauthorized domain
            }
        }
    }

    private fun isDomainAllowed(urlString: String): Boolean {
        val url = URL(urlString)
        val domain = url.host
        return allowedDomains.contains(domain)
    }

    private fun fetchContent(urlString: String): String? {
        var connection: HttpURLConnection? = null
        var reader: BufferedReader? = null

        try {
            val url = URL(urlString)
            connection = url.openConnection() as HttpURLConnection
            connection.requestMethod = "GET"

            // Check if the response code is successful
            if (connection.responseCode == HttpURLConnection.HTTP_OK) {
                reader = BufferedReader(InputStreamReader(connection.inputStream))
                val response = StringBuilder()
                var line: String?

                // Read the response line by line
                while (reader.readLine().also { line = it } != null) {
                    response.append(line)
                }

                return response.toString()
            }
        } catch (e: Exception) {
            e.printStackTrace()
        } finally {

```

```

        // Close the connection and reader
        connection?.disconnect()
        reader?.close()
    }

    return null
}
}

```

RENFORCEMENT

Configuration réseau non sécurisée

CVSS v3: 9,1

L'application ne définit pas de configuration de sécurité réseau ou applique des paramètres non sécurisés.

Android Network Security Configuration permet de définir de manière déclarative la sécurité réseau de l'application.

En l'absence d'Android Network Security Configuration ou si elle est configurée de manière non sécurisée, l'application peut être vulnérable à des attaques MITM.

Détails techniques

Résultat #1

`res/xml/network_security_config.xml` : The `base-config` authorizes cleartext traffic.

Recommandation

La configuration de sécurité réseau Android est un fichier XML permettant de configurer de manière déclarative la sécurité réseau de l'application.

Ajouter la configuration de sécurité réseau à votre application :

1- Créez un fichier XML dans le répertoire `res/xml` de votre application. Nommez-le `network_security_config.xml`, ou tout autre nom approprié. 2- Définissez les configurations de sécurité (voir les exemples ci-dessous) 3- Appliquez la configuration au manifeste :

</ XML

```

<application
    android:networkSecurityConfig="@xml/network_security_config"
    <!-- Other attributes -->
    >
    <!-- Other configurations -->
</application>

```

Exemples de configuration de sécurité réseau

- Autorité de certification personnalisée avec prise en charge des paramètres de débogage

</ XML

```

<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
  <debug-overrides>
    <trust-anchors>
      <certificates src="@raw/debug_cas"/>
    </trust-anchors>
  </debug-overrides>
</network-security-config>

```

- Désactivation déclarative du trafic en clair

</ XML

```

<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
  <domain-config cleartextTrafficPermitted="false">
    <domain includeSubdomains="true">secure.example.com</domain>
  </domain-config>
</network-security-config>

```

```
</domain-config>
</network-security-config>
```

- Configuration déclarative des clés de certificate pinning

XML

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
  <domain-config>
    <domain includeSubdomains="true">example.com</domain>
    <pin-set expiration="2018-01-01">
      <pin digest="SHA-256">7HIpactkIAq2Y49orF00QKurWxmmSFZhBCoQYcRhJ3Y=</pin>
      <!-- backup pin -->
      <pin digest="SHA-256">fwza0LRMXouZHRc8Ei+4PyuldPDcf3UKg0/04cDM1oE=</pin>
    </pin-set>
  </domain-config>
</network-security-config>
```

Détails du vecteur CVSS

Vecteur CVSS V3 : CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

Vecteur d'Attaque:	Réseau (N) Adjacent (A) Local (L) Physique (P)
Complexité d'Attaque:	Faible (L) Élevé (H)
Privilèges Requis:	Aucun(e) (N) Faible (L) Élevé (H)
Interaction Utilisateur:	Aucun(e) (N) Requis (R)
Périmètre:	Modifié (C) Non modifié (U)
Confidentialité:	Élevé (H) Faible (L) Aucun(e) (N)
Intégrité:	Élevé (H) Faible (L) Aucun(e) (N)
Disponibilité:	Élevé (H) Faible (L) Aucun(e) (N)

Références

- [Network Security Configuration \(Android developer\)](#)

RENFORCEMENT

Mode débogage activé

CVSS v3: 5,4

L'application est compilée avec le mode débogage activé.

L'attribut `android:debuggable` de l'élément `Application` défini dans l'Android Manifest détermine si l'application peut être déboguée.

Le mode debug permet à un attaquant d'accéder au système de fichiers de l'application, d'attacher un débogueur et d'accéder à des données sensibles ou d'effectuer des actions malveillantes.

Les étapes suivantes permettent de démarrer une session de débogage avec `jdb` :

- Utilisez `adb jdwp` pour identifier le `PID` de l'application cible :

```
$adb jdwp
3466
15446
```

- Créez un canal de communication avec `adb` et attachez-vous à celui-ci à l'aide de `jdb` :

```
$adb forward tcp:7777 jdwp:$(adb shell ps | grep "package-id")
$jdb -attach localhost:7777
```

- Accédez au système de fichiers de l'application :

```
$adb shell
$run-as package-id
$...insert malicious action...
```

Un attaquant peut déboguer l'application sans accès au code source et l'exploiter pour effectuer des actions malveillantes au nom de l'utilisateur, modifier le comportement de l'application ou accéder à des données sensibles telles que des identifiants et des cookies de session.

Détails techniques

Résultat #1

Debug mode is set to true, enabling attackers to access the application filesystem exposing sensitive data.

```
<application xmlns:android="http://schemas.android.com/apk/res/android" android:theme="@7F0C0005" android:label="@7F0B0036"
android:icon="@7F0A0000" android:debuggable="true" android:allowBackup="true" android:supportsRtl="true"
android:networkSecurityConfig="@7F0E0000" android:roundIcon="@7F0A0002">
```

Recommandation

Désactivez le mode débogage en définissant l'attribut `android:debuggable` à `false` dans la balise `Application` de l'Android Manifest.

</> XML

```
<application android:icon="@drawable/icon" android:debuggable="false">
```

Détails du vecteur CVSS

Vecteur CVSS V3 :

CVSS: 3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N

Vecteur d'Attaque:

Réseau (N) Adjacent (A) Local (L) Physique (P)

Complexité d'Attaque:

Faible (L) Élevé (H)

Privilèges Requis:

Aucun(e) (N) Faible (L) Élevé (H)

Interaction Utilisateur:

Aucun(e) (N) Requis (R)

Périmètre:

Modifié (C) Non modifié (U)

Confidentialité:

Élevé (H) Faible (L) Aucun(e) (N)

Intégrité:

Élevé (H) Faible (L) Aucun(e) (N)

Disponibilité:

Élevé (H) Faible (L) Aucun(e) (N)

Références

- [DRD10-J Do not release apps that are debuggable \(CERT Secure Coding\)](#)

Surface d'attaque

2026-05-19 16:40:34 • Restreint • Ostorlab