



制限付き

スキャンレポート

宛先

Ostorlab Official

総合リスク - 高

日付

2026-10-15 18:15:55

エグゼクティブサマリー

アプリケーション  **PlayStore**([owasp.sat.agoat:1.0 \(1\)](#))には **高 | 9.1** のリスク評価が割り当てられており、現在 **44** 件の脆弱性を抱えており、**0** 件がクリティカルリスク、**12** 件が高リスク、**16** 件が中リスク、**6** 件が低リスクと評価されています。さらに **8** 件の潜在的脆弱性があり、詳細な調査が必要です。アプリケーションのセキュリティをさらに強化するため、**2** 件のハードニング対策の実施を推奨します。

このアプリケーションの攻撃対象領域は「**注目すべき**」レベルであり、**0** パーミッション、うち **0** 件が危険な権限です。また、アプリケーションは **0** 件のアクティビティ、**0** 件のサービス、および **0** 件のブロードキャストレシーバーを公開しています。さらに、**0** 件の機能を使用しています。攻撃対象領域が適切かどうかを確認し、不要な機能や公開を削除してください。

全体として、アプリケーションのカバレッジは「**制限あり**」と評価されています。カバレッジを向上させるために、完全な動的スキャンを実行することをお勧めします。

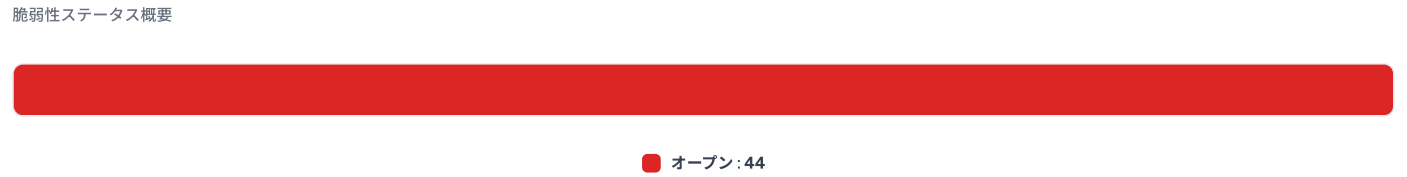
アプリケーション概要

プラットフォーム	 Android
パッケージ	owasp.sat.agoat
バージョン	1.0 (1)
SHA1ハッシュ	ea96f4adc27083696a449b1c6e5cd612a2cf0f5c
サイズ	2 MB
日付	5月 19th 2026, 16:40:34

検出結果サマリー



チケットサマリー

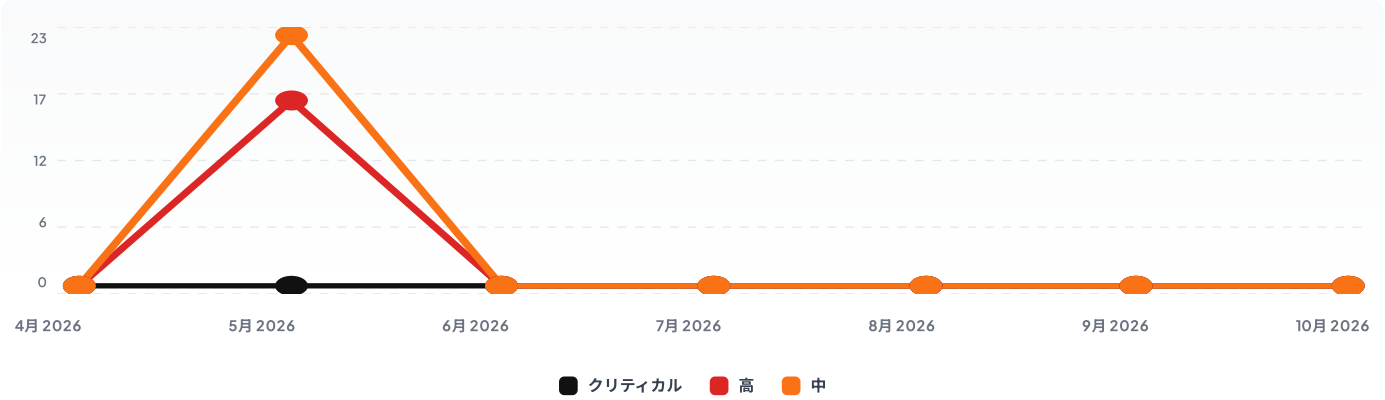


重大度別の脆弱性トップ10

システムクリップボードおよび...	1
SQL Injection via Unsanitized r...	1
暗黙的にエクスポートされた保...	1
Exported Serviceによる未認可...	1
SharedPreferences、SQLite...	1
ソルトなし MD5 ハッシュで保...	1
Confirmed Clickjacking on Se...	1
Android Logcatへの生の認証...	1
InsecureStorageSQLiteActivi...	1
インターネット公開されたFire...	1

アプリケーション脆弱性トレンド

過去6ヶ月間の推移



スキャンサマリー

対象	 Android: owasp.sat.agoat (1.0 (1))
総合リスク	HIGH
脆弱性	合計 34 件 クリティカル 0 件 高 12 件 中 16 件 低 6 件 潜在的 8 件 ハードニング 2 件

検出結果一覧

リスク CVSS	ステータス	タグ	タイトル	概要
高	オープン	タグはありません。	システムクリップボードおよび脆弱なPRNGによる低エントロピーOTPの露出	<code>ClipboardActivity</code> は暗号論的でないPRNGを使用して4桁のOTPを生成し、グローバルクリップボードにコピーしてToastで表示しているため、このトークンは予測可能であり、同じ端末にインストールされた他のアプリやIMEからアクセス可能になっています。
高	オープン	タグはありません。	SQL Injection via Unsanitized.rawQuery in SQLInjectionActivity	SQLInjectionActivityは、EditTextからのユーザー入力をrawQuery()に直接連結してSELECTクエリを構築し、selectionArgsをnullのままにしています。これにより、任意のSQL実行、スキーマ列挙、およびローカル認証バイパスが可能になります。
高	オープン	タグはありません。	暗黙的にエクスポートされた保護対象アクティビティが認証バイパスを可能にする (AccessControlViewActivity)	AccessControlViewActivityはdeep-linkのintent-filterにより暗黙的にexportedとなっており、内部での認証チェックがないため、任意のサードパーティアプリケーションがPINなしで保護されたビューを起動できます。
高	オープン	タグはありません。	Exported Serviceによる未認可ファイルダウンロード (DownloadInvoiceService)	<code>DownloadInvoiceService</code> はマニフェストで明示的にexportedに設定されており、呼び出し元の認可チェックを一切行わずにpublic external storageへファイルをダウンロードします。
高	オープン	タグはありません。	SharedPreferences、SQLite、External Storageにわたる機密認証情報の平文保存	アプリケーションは、ユーザー名、パスワード、およびPINハッシュを複数のローカルストレージメカニズムに永続化しており、暗号化、KeyStore保護、または制限的なファイル権限が設定されていません。
高	オープン	タグはありません。	ソルトなしMD5ハッシュで保護された認証情報 (AccessControlIssue1Activity)	本アプリケーションはPINをソルトなしのMD5ダイジェストとして内部SharedPreferencesに保存しており、オフラインでのブルートフォース攻撃により平文PINを数ミリ秒で容易に復元可能です。
高	オープン	タグはありません。	Confirmed Clickjacking on Sensitive Activities in owasp.sat.agoat	SYSTEM_ALERT_WINDOWを持つ悪意あるアプリは、4つの機密性の高いアクティビティ上にインタラクティブなオーバーレイを描画し、ユーザーのタッチ操作を傍受して不正な請求書のダウンロード、プロモーションの検証、OTPの生成、PINの送信を誘発する可能性があります。
高	オープン	タグはありません。	Android Logcatへの生の認証情報およびSQLクエリの漏洩	複数のアクティビティが <code>Log.e</code> 、 <code>Log.v</code> 、 <code>System.out.println</code> を使用し、生のユーザー名、パスワード、およびSQL文を無条件にAndroidシステムログに書き込んでいるため、ログへのアクセス権を持つ任意のアプリや <code>adb</code> を持つ攻撃者による認証情報の窃取が可能です。
高	オープン	タグはありません。	InsecureStorageSQLiteActivityにおける未サニタイズexecSQLによるSQLインジェクション	InsecureStorageSQLiteActivityは、サニタイズされていない2つのEditTextの値を連結してexecSQL()に渡すことでINSERT文を構築しており、これによりスタックド・クエリ・インジェクション、レコードの任意改変、およびローカルデータベースの完全な整合性侵害が可能となります。
高	オープン	セキュリティ	インターネット公開されたFirebase Database	Firebaseデータベースを公開状態にしていると、不正アクセス、データ改ざん、情報漏洩、悪意ある攻撃、および信用失墜を招く可能性があります。
高	オープン	プライバシー セキュリティ	LOGJAM 共通素数の脆弱性	広く共有されたDiffie-Hellman素数の使用により、インターネットトラフィックの大部分に対する国家レベルの事前計算攻撃が可能となる。
高	オープン	セキュリティ	モバイルSQLインジェクション脆弱性	モバイルSQL Injection脆弱性は、攻撃者がモバイルアプリケーションのデータベースに対して悪意のあるSQLクエリを実行可能にし、不正アクセス、データ漏洩、またはデータ改ざんにつながる可能性があります。

リスク CVSS	ステータス	タグ	タイトル	概要
中	オープン	タグはありません。	クライアントサイドのプール値フラグで管理者権限が決定され、サーバー側での検証が行われていない	BinaryPatchingActivityはローカルのプール値フィールドを使用して管理UI機能へのアクセスを制御していますが、このフィールドはバックエンドAPI、暗号化トークン、再認証チャレンジのいずれでも検証されていません。
中	オープン	タグはありません。	Certificate Pinning in TrafficActivity Silently Fails Due to Broad Exception Handling	<code>TrafficActivity.doPinning()</code> 内の OkHttp CertificatePinner は、広範な catch ブロックによって <code>SSLPeerUnverifiedException</code> が握り潰されているため無効化されており、検知されない中間者攻撃を許容しています。
中	オープン	タグはありません。	不安全な外部ストレージ権限による認証情報漏洩 (InsecureStorageSDCardActivity)	<code>InsecureStorageSDCardActivity</code> は、ユーザーの認証情報を外部ストレージ上の一時ファイルに書き込み、そのファイルを明示的に world-readable および world-writable に設定します。
中	オープン	タグはありません。	FLAG_SECURE未設定により、機密画面で不正な画面キャプチャが許容される	owasp.sat.agoat の 25個すべてのアクティビティに FLAG_SECURE が設定されておらず、PIN の入力や認証情報の入力、OTP の生成など、機密画面のスクリーンショットおよび画面録画が制限なく許可されています。
中	オープン	タグはありません。	不安全なバックアップおよびデバッグフラグによるアプリケーションサンドボックスの露出	アプリケーションマニフェストで <code>android:debuggable="true"</code> および <code>android:allowBackup="true"</code> が有効になっており、USBデバッグアクセス権を持つ任意のユーザーが、アプリのプライベートデータをすべて抽出・変更・復元したり、実行中のプロセスにデバッグをアタッチしたりできる状態です。
中	オープン	タグはありません。	アプリケーションソースコード内にハードコードされた認証情報	複数のハードコードされたシークレット（ユーザー名、パスワード、および認証キーを含む）が、アプリケーションのソースコードに直接埋め込まれており、exported BroadcastReceiverを介して実行時に露出しています。
中	オープン	🔒 セキュリティ	暗号化の脆弱性: 脆弱なハッシュアルゴリズム	クライアントは既知の暗号学的脆弱性を持つ脆弱なハッシュ関数を使用しています。
中 9.1	オープン	🔒 セキュリティ	平文HTTPリクエスト	アプリケーションが暗号化されていないチャネルでデータにアクセスしており、通信中のデータの機密性と完全性を損なっています。
中	オープン	🔒 セキュリティ 🔒 プライバシー	CERTIFICATE_EXPIRED	期限切れのSSL/TLS証明書はブラウザやクライアントから信頼されなくなり、セキュリティ警告やサービス中断、潜在的なセキュリティ脆弱性（中間者攻撃など）を招きます。
中	オープン	🔒 セキュリティ	安全でないHTTPヘッダー設定	アプリケーションのヘッダー設定が未設定であるか、安全でないヘッダー値が設定されています。
中	オープン	🔒 セキュリティ	安全でないHTTPヘッダー設定: HTTP Strict Transport Security (HSTS)	安全でないHTTPヘッダー設定: HTTP Strict Transport Security (HSTS).
中	オープン	🔒 セキュリティ	不安全なTLS暗号スイートのサポート	クライアントは、既知の暗号上の弱点を持つ暗号スイートの組み合わせをサポートしています。
中	オープン	🔒 セキュリティ 🔒 プライバシー	TLS_FALLBACK_SCSV非対応	サーバーは TLS_FALLBACK_SCSV をサポートしておらず、プロトコルダウングレード攻撃に対して脆弱です。
中	オープン	🔒 セキュリティ 🔒 プライバシー	SSL/TLS証明書の暗号鍵および署名アルゴリズムの脆弱性	SSL/TLS証明書は暗号鍵と署名アルゴリズムの両方が脆弱であり、暗号化接続の安全性を損ない、証明書の偽造やその他の暗号攻撃を許容する可能性があります。
中	オープン	🔒 セキュリティ 🔒 プライバシー	モバイルネットワーク接続の要請	アプリケーションはモバイルネットワーク接続の使用を強制しています。

リスク CVSS	ステータス	タグ	タイトル	概要
中	オープン	🔒 セキュリティ	XMLインジェクション	XMLインジェクションは、信頼できないユーザー入力 がXMLデータに組み込まれる前に適切に検証またはサ ニタイズされない場合に発生する脆弱性です。攻撃者 は悪意のあるXMLコンテンツを注入することでこの脆 弱性を悪用でき、不正アクセス、データ改ざん、また はサービス拒否攻撃を引き起こす可能性があります。
低 5.4	オープン	🔒 セキュリティ	usesCleartextTraffic 属性が設定されて います	アプリケーションがusesCleartextTraffic 属性を設定 しています。
低	オープン	🔒 セキュリティ	非推奨のTLS/SSLプロトコルバージョン の使用	アプリケーションが、既知のセキュリティおよびプ ライバシー上の問題を抱える非推奨のSSL/TLS プロト コルを設定しています。
低	オープン	🔒 セキュリティ	Google Calendar - 公開	対象に埋め込まれたGoogle Calendarが公開されてい る状態で検出され、会議の詳細、参加者名、イベント 予定、社内組織データなどの機密情報が漏洩する可 能性があります。
低 5.4	オープン	🔒 セキュリティ	非推奨のTargetAPIバージョン	アプリケーションはtargetSdkVersion属性を古い Android API レベルの使用を許可するように設定して おり、ユーザーをセキュリティ脆弱性にさらす可能性 があります。
低 5.4	オープン	🔒 プライバシー	属性hasFragileUserData が未設定です	アプリケーションがhasFragileUserData 属性を明示 的に設定していません。
低	オープン	🔒 セキュリティ 🔒 マルウェア 🔒 エクスプロイト	タスクハイジャック	アプリケーションはタスクハイジャックに対する保護 がなく、デバイス上の悪意あるアプリケーションがそ のフォアグラウンドタスクを乗っ取る可能性があり ます。
潜在的	オープン	タグはありません。	ANALYTICS_LIBRARYにおけるネット ワークセキュリティ設定の不備およびト ランスポート保護の欠如	ANALYTICS_LIBRARYサロゲート（TrafficActivity） は、デフォルトの未ピン留めOkHttpClientを介して アナリティクストラフィックを送信している。一方、 Network Security Configurationはグローバルに平文 HTTPを許可し、ユーザーインストールのCAを信頼し ている。このパイプラインにはペイロード暗号化、リ クエスト署名、および整合性制御が欠如しており、 MitMによる傍受、リプレイ、および改ざんが容易に 行える。
潜在的	オープン	🔒 セキュリティ	アプリケーションに保存された機密情報	アプリケーション内にパスワード、トークン、その他 の機密情報が保存されています。
潜在的	オープン	🔒 セキュリティ	パストラバーサル	アプリケーション内のパストラバーサルの存在は、攻 撃者がファイルパスの一部を操作できる脆弱性を示し ています。この改変されたパスは、適切な検証や精査 なしにファイルシステムAPIによって利用されます。
潜在的	オープン	🔒 セキュリティ	アプリケーションデータの不安全な保存	アプリケーションデータの安全でない保存により、他 のアプリによるデータの読み取りおよび改ざんが可 能な状態。
潜在的 3.9	オープン	🔒 プライバシー	バックアップモードが有効です	アプリケーションがバックアップモードを有効化して います。
潜在的	オープン	🔒 セキュリティ	ソースからシンク	危険なSinkメソッドが、ユーザが制御しうるパラメー タまたは信頼できないSourceメソッドを用いて呼び 出されています。
潜在的 3.4	オープン	🔒 セキュリティ	パーミッションなしで宣言されたサービ ス	宣言されたサービスがグローバルパーミッションで保 護されていません。
潜在的	オープン	🔒 セキュリティ	URL操作	アプリケーションがコンテンツを取得するために使用 するURLが攻撃者に制御されると、コンテンツの改竄 が可能となり、悪性コードの実行や機密情報の窃取に 至るセキュリティリスクが生じる。
ハードニング 9.1	オープン	🔒 プライバシー 🔒 セキュリティ	不安全なネットワーク構成設定	アプリケーションにネットワークセキュリティ構成が 指定されておらず、または安全でない設定が適用され ています。

リスク CVSS	ステータス	タグ	タイトル	概要
ハードニング 5.4	オープン	🔒 セキュリティ	デバッグモードが有効です	アプリケーションはデバッグモードが有効な状態でコンパイルされています。

詳細検出結果リスト

高

システムクリップボードおよび脆弱なPRNGによる低エントロピーOTPの露出

`ClipboardActivity` は暗号論的でないPRNGを使用して4桁のOTPを生成し、グローバルクリップボードにコピーしてToastで表示しているため、このトークンは予測可能であり、同じ端末にインストールされた他のアプリやIMEからアクセス可能になっています。

OTPは `CollectionsKt.shuffled` を用いてわずか9,000通りの値のプールから生成されており、内部で `java.util.Random` に委譲しています。このLCGベースのジェネレーターは、連続した観測値からシードが復元可能です。生成されたトークンは有効期限やクリア機構がなく、システムクリップボードに配置され、同時にToastにも表示されます。サーバーサイドの検証バックエンドが存在しないため、OTPは純粋なクライアントサイドの産物であり、クリップボードリーダーによる傍受やPRNGシーケンスの観測による予測が可能です。

技術的詳細

検出事項 #1

根本原因

開発者は、 `java.security.SecureRandom` の代わりに `java.util.Random` を内部で使用する簡易な shuffle メソッドを利用しました。生成されたトークンはグローバルクリップボードに格納されたうえ、UI Toastにも表示されていますが、タイムアウト、検証、アンチリプレイのいずれのロジックも存在しません。

脆弱性を含むコード

`ClipboardActivity.java` (63、68〜71行目) :

```
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();
```

悪用の証拠

[静的解析] 静的監査により、 `IntRange(1000, 9999)` のトークン空間が特定され、PRNGがKotlinのデフォルト `Random` インスタンスを介して `java.util.Random` に遡及しました。また、 `ClipboardManager.setPrimaryClip` の呼び出しが70行目で確認されたものの、その後の `clearPrimaryClip` やタイムアウト処理は存在しません。本アクティビティにはネットワークおよび検証のロジックが一切含まれていません。

検証根拠

根本原因

`owasp.sat.agoat.ClipboardActivity` における OTP 生成は `CollectionsKt.shuffled(new IntRange(1000, 9999))` に依存しており、これはKotlin標準ライブラリを介して `java.util.Collections.shuffle(List)` に委譲されます。このJava APIは、システム時刻からシードを取得するデフォルトの `java.util.Random` インスタンスを生成し、 `java.security.SecureRandom` は使用されません。その後、アクティビティは生成されたトークンを2つの不安全なチャネルで露出します。ひとつは `ClipboardManager.setPrimaryClip` によるグローバルAndroidクリップボードであり、もうひとつは `Toast.makeText` によるUI Toastです。本アクティビティのいかなる箇所にも、有効期限、クリア処理、またはサーバーサイド検証のロジックは存在しません。

脆弱性を含むコード

`ClipboardActivity.java` (63、68〜71行目) :

```
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();
```

Kotlin標準ライブラリから `java.util.Random` への委譲 (OTPCLIP-003 トレース) :

```
// 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);
        ...
    }
}
```

悪用の証拠

[静的解析] 包括的な静的検証（OTPCLIP-001からOTPCLIP-003まで）により、以下が確認されました。

- 正確なソース位置：** `owasp/sat/agoat/ClipboardActivity.java`（`/workspace/source_code_analysis_ffd42589f4eb4cf4a51427480ebe2d69/base.apk/sources/owasp/sat/agoat/ClipboardActivity.java`）、計79行。
- 63行目でOTP生成が確認：** `CollectionsKt.shuffled(new IntRange(1000, 9999))` は正確に9,000通りの可能値（1000から9999まで含む）を生成し、およびその13.14ビットのエントロピー、すなわち `log2(9000)` を生み出します。
- PRNGチェーンを完全に追跡：** 引数なしの `CollectionsKt.shuffled` オーバーロードは `Collections.shuffle(List)` を呼び出し、これは時刻由来のシードを持つデフォルトの `java.util.Random` をインスタンス化します。ファイル `CollectionsKt_MutableCollectionsJVMKt.java`（56～61行目）および `CollectionsKt_MutableCollectionsKt.java`（296～305行目）は、この委譲チェーンを証明しています。`ClipboardActivity.java` には `SecureRandom` のインポートまたは使用は一切存在しません。
- 68～70行目でクリップボード露出が確認：** `ClipData.newPlainText("CC Card", String.valueOf(otp))` を伴って `ClipboardManager.setPrimaryClip(clip)` が呼び出されています。後続の `clearPrimaryClip()` や `Handler.postDelayed()` およびライフサイクルクリーンアップによるクリップの削除はありません。
- 71行目でToast露出が確認：** `Toast.makeText(ClipboardActivity.this, "OTP Generated and Copied: " + otp, 1).show()` は生のOTP整数値を画面に描画し、これによりアクセシビリティサービスや画面収録からアクセス可能となっています。
- 検証ロジックなし：** アクティビティおよびインポートを横断的に検索した結果、`HttpURLConnection` や `OkHttp` や `Retrofit` や `Volley` といったネットワークI/O APIや検証キーワードは一切検出されませんでした。OTPは純粋にクライアントサイドで生成されており、サーバー検証、有効期限、アンチリプレイ保護のいずれも存在しません。
- 攻撃表面の制約：** `AndroidManifest.xml`（39行目）によれば、本アクティビティは **intent-filterを持たず**、`android:exported="true"` も設定されていません。つまり、外部アプリが直接OTP生成をトリガーすることはできません。ただし、ユーザーがアプリ内UIと一度インタラクトすると、そのOTPはあらゆるクリップボード読み取りアプリやアクセシビリティサービスに即座に利用可能になります。
- 代替となる安全なPRNGなし：** `ClipboardActivity.java` のインポートには `java.util.*` コレクションおよびKotlin標準ライブラリの型のみが含まれており、`java.security.SecureRandom` や `kotlin.random.Random` や `Math.random()` はいずれも存在しません。

推奨事項

即時対応

脆弱なジェネレータを `SecureRandom` に置き換え、OTPの長さを6桁以上にする：

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

OTPをシステムクリップボードに配置しないこと。やむを得ない場合は `Handler` を用いて短時間後にクリアすること。

恒久対策

サーバーサイドにOTP生成および検証エンドポイントを実装すること。クライアント側でリカバリートークンの生成や検証を行ってはならない。利用は単回のみとし、有効期限を約5分間に設定し、アカウントレベルのレート制限を適用すること。

検証

パッチ適用済みAPKを逆コンパイルし、OTPフロー内に `CollectionsKt.shuffled` および `ClipboardManager.setPrimaryClip` が存在しないことを確認すること。コードベース内を `java.util.Random` で検索し、`java.security.SecureRandom` に置き換えられていることを確認すること。

高 SQL Injection via Unsanitized rawQuery in SQLInjectionActivity

SQLInjectionActivity は、EditText からのユーザー入力を rawQuery() に直接連結して SELECT クエリを構築し、selectionArgs を null のままにしています。これにより、任意の SQL 実行、スキーマ列挙、およびローカル認証バイパスが可能になります。

SQLInjectionActivity は、ローカルの `aGoat` SQLite データベースに対してユーザー名の読み取り専用ルックアップを実行しています。クエリ文字列が `StringBuilder` による連結で構築される際、サニタイズされていない `EditText` の入力値がそのまま連結されて `rawQuery(qry, null)` に渡されているため、攻撃者は `WHERE` 句に任意の SQL 構文を注入できます。注入に成功するとクエリのセマンティクスが改変され、無条件にすべての行が返されるようになり、アプリケーションは `cursor.getCount() > 0` というロジックによりこれを有効なログインと解釈します。これにより `users` テーブルに保存されているすべての平文ユーザー名およびパスワードが露出し、`UNION SELECT` によるスキーマ抽出も可能になります。さらに完全なクエリ文字列が Logcat の `VERBOSE` レベルでログ出力されており、注入されたペイロードがデバイスログに残存する可能性があります。

技術の詳細

検出事項 #1

根本原因

`owasp/sat/agoat/SQLInjectionActivity.java` の 69 行目で、アプリケーションはサニタイズされていないユーザー入力をクエリ文字列に直接連結することにより、SQL `SELECT` ステートメントを動的に構築しています。`rawQuery()` メソッドは `selectionArgs` パラメータに `null` を指定して呼び出されており、SQLite のパラメータ化を利用する余地を完全に排除しています。

脆弱なコード

静的解析により特定された脆弱なクエリ構築パターンは以下のとおりです：

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

`username` 変数は `R.id.userName` から取得されており、検証、許可リスト、サニタイズが行われていない自由形式の文字列入力を受け付ける `EditText` ウィジェットです。

悪用の証拠

[Simulated] Android エミュレータや `adb` ツールが使用できなかったため、悪用の検証は `aGoat` のスキーマ（`users` テーブル、`ID INTEGER PRIMARY KEY AUTOINCREMENT`、`username VARCHAR`、`password VARCHAR` を持つ）を正確に再現したローカル環境のシミュレートされた SQLite データベースに対して実施されました。再構成したクエリは攻撃者が制御する入力値で実行されました：

- 構文破壊ペイロード：** `'` をクエリに注入した入力により `unrecognized token` エラーが発生し、入力が文字列リテラルの文脈から脱出したことを確認しました。
- ブーリアンバイパスペイロード：** 入力 `' OR '1'='1' --` はクエリを `SELECT * FROM users WHERE username=' ' OR '1'='1' --'` に変換しました。シミュレーションでは `users` テーブルの **すべての行** が返され、無条件での行取得が実証されました。
- UNION ベースのスキーマ抽出：** 入力 `' UNION SELECT null, sql, null FROM sqlite_master --` により `sqlite_master` の CREATE ステートメントが正常にダンプされ、スキーマの列挙が確認されました。

アプリケーションのロジックでは `cursor.getCount() > 0` をチェックして成功判定を行っています。ブーリアンバイパスによりカーソルに行が含まれる状態が強制されるため、アプリケーションは成功の分岐に進み、一致したすべてのクレデンシャルを Toast に表示します。

検証エビデンス

検証ステータス

今回の検証では対象 APK に対する動的テストタスクは一切実行されませんでした。元の静的解析結果を超える新たな悪用の証拠は生成されていません。

根本原因

[Static Analysis] 逆コンパイルされた `owasp/sat/agoat/SQLInjectionActivity.java` の 69 行目では、サニタイズされていないユーザー入力をクエリ文字列に直接連結することで SQL `SELECT` ステートメントを動的に構築しています。`rawQuery()` メソッドは `selectionArgs` パラメータに `null` を指定して呼び出されており、SQLite のパラメータ化を利用する余地を完全に排除しています。

脆弱なコード

[Static Analysis] 逆コンパイルされた APK 内で以下の安全でないクエリ構築パターンが特定されました：

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

`username` 値は `R.id.userName` から取得されており、検証、許可リスト、サニタイズが行われていない自由形式の文字列入力を受け付ける `EditText` ウィジェットです。

悪用の証拠

[Static Analysis] 脆弱なコードパターンにより、攻撃者が制御する入力が `WHERE` 句内のシングルクォートで囲まれた文字列リテラルの文脈から脱出し、クエリのセマンティクスを変更可能となっています。[Simulated] `aGoat` スキーマを再現した環境での事前シミュレーションにより、以下が実証されました：

- `'` を入力すると SQL 構文が破壊され、`unrecognized token` エラーが発生する
- `' OR '1'='1' --` を入力するとクエリが変更され、`users` テーブルのすべての行が無条件に返される
- `' UNION SELECT null, sql, null FROM sqlite_master --` を入力すると `sqlite_master` からスキーマ情報が抽出される

アプリケーションのロジックでは認証成功の判定に `cursor.getCount() > 0` を使用しているため、すべての行が返されるとこのチェックを迂回し、`users` テーブルに格納された平文のクレデンシャルが漏洩します。

推奨事項

即時の軽減策

`AndroidManifest.xml` から `android:debuggable="true"` を削除し、ユーザー入力を Logcat に出力する詳細な SQL クエリログを無効化してください。

恒久的な修正

`SQLInjectionActivity.java` 内の動的文字列連結を、`?` プレースホルダーと `selectionArgs` 配列を使用したパラメータ化された `rawQuery()` に置き換えてください：

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

これにより、ユーザー入力は厳密にリテラル文字列値として扱われ、クエリ構造を変更できなくなります。

検証

- 静的解析を再実行し、ユーザー入力が存在する箇所で `selectionArgs` が `null` の `rawQuery()` 呼び出しがないことを確認してください。
- `' OR '1'='1' --` などのペイロードを入力し、クエリがすべての行を返すのではなくゼロ行を返すことを確認してください。
- Logcat 出力を確認し、ユーザー入力を含む完全な SQL 文字列が `VERBOSE` レベルで出力されていないことを確認してください。

高 暗黙的にエクスポートされた保護対象アクティビティが認証バイパスを可能にする (AccessControl1ViewActivity)

`AccessControl1ViewActivity` は deep-link の intent-filter により暗黙的に exported となっており、内部での認証チェックがないため、任意のサードパーティアプリケーションが PIN なしで保護されたビューを起動できます。

この脆弱性は、`AccessControl1Issue1Activity` に実装された PIN 認証ゲートを完全に無効化します。攻撃者は物理的な ADB アクセスや同端末にインストールされた悪意のあるアプリケーションを利用し、認証情報なしで保護された請求書ダウンロードダッシュボードにアクセスできます。本番環境では、機密なユーザー機能を保護する主要な機密性および完全性の制御が回避されます。

技術的詳細

検出事項 #1

根本原因

`AccessControl1ViewActivity` は、カスタム URL スキーム `androgoat://vulnapp` を持つ `<intent-filter>` とともに `AndroidManifest.xml` に宣言されているが、明示的な `android:exported="false"` 属性が欠落している。`targetSdkVersion="26"` は Android 12 以前の動作を示しており、intent-filter の存在により Activity が暗黙的にエクスポートされる。Activity のソースコードを静的解析した結果、`onCreate()` 内に認証ロジックが一切存在しないことを確認した。

脆弱なコード

`AndroidManifest.xml` (暗黙のエクスポート宣言)：

```
<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 (45〜54行目) : 認証ゲートなし

```
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();
        }
    });
}
```

悪用の証拠

【静的解析】 マニフェストおよびソースコードにより、任意のサードパーティアプリまたは `adb` コマンドからこのActivityを直接起動できることが確認される。通常のアプリケーションフローでは `AccessControlIssue1Activity` を介してPINの検証が必要だが、そのゲートはディープリnkを直接呼び出すことで完全に迂回される。

概念実証コマンド:

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

想定されるアクティビティスタックの確認:

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

検証証拠

根本原因

デコンパイルしたAPKの静的検証により、 `AccessControl1ViewActivity` が `base.apk/AndroidManifest.xml` に `<intent-filter>` とともに宣言されており、カスタムディープリnkスキーム `androgoat://vulnapp` を露出していることが確認された。 `<activity>` タグには `android:exported="false"` および `android:permission` 属性のいずれも欠落している。 `base.apk/apktool.yml` の監査により `targetSdkVersion: 26` が確認された。Android12以前の動作では、この欠落によりコンポーネントが暗黙的にエクスポートされる。

脆弱なコード

AndroidManifest.xml (検証済みの正確な宣言) :

```
<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` 38〜48行目) : 無条件で保護対象のUIを描画

```
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();
        }
    });
}
```

このクラスは `android.support.v7.app.AppCompatActivity` を継承しており、認証を強制するカスタム基底クラスは存在しない。この54行のファイル全体に `getSharedPreferences` の呼び出し、MD5の比較、PINの再入力要求、およびセッションの検証は一切含まれていない。

比較証拠：PINゲートの分離 すべての認証ロジックは `AccessControlIssue1Activity.java`（概ね88〜124行目）にのみ存在し、PINのMD5ハッシュを `pinDetails` `SharedPreferences` に保存し、検証成功後にのみ `AccessControl1ViewActivity` を起動する。

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

悪用の証拠

【静的解析】マニフェストおよびソースコードにより、任意のサードパーティアプリまたはADBシェルからActivityを直接起動できることが確認される。ディープリンクのintentは暗黙的にエクスポートされたコンポーネントに解決される。

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

検証により、**2つ目の未認証シンク**が発見された。`DownloadInvoiceService` は明示的にエクスポートされており（`android:exported="true"`、パーミッションなし）、Activityの起動すら迂回して直接呼び出すことができる。

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

いずれの経路でも、アプリケーションはPINチャレンジを提示することなく請求書ダウンロード機能に到達する。

推奨事項

Immediate

`AndroidManifest.xml` 内の `AccessControl1ViewActivity` に `android:exported="false"` を追加するか、ディープリンクが不要な場合は `<intent-filter>` を削除してください。

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

Permanent Fix

ディープリンク機能を維持する必要がある場合は、保護対象のビューをインフレートする前に `AccessControl1ViewActivity.onCreate()` 内で実行時の認証チェックを実装してください：

```
@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
}
```

Verification

- `adb shell am start -d "androgoat://vulnapp"` でActivityを起動しようと試み、起動がブロックされるか、アプリが保護対象のビューに遷移しないことを確認してください。
- マニフェストを検査し、`android:exported="false"` が設定されていることを確認してください。
- `drozer` または `am` を実行し、ActivityがサードパーティのUIDから到達不可能になったことを確認してください。

高 Exported Serviceによる未認可ファイルダウンロード (DownloadInvoiceService)

`DownloadInvoiceService` はマニフェストで明示的に `exported` に設定されており、呼び出し元の認可チェックを一切行わずに `public external storage` へファイルをダウンロードします。

この脆弱性により、`exportedService` は未認証のファイルダウンロードゲートウェイと化す。保存先が `public external storage` なので `READ_EXTERNAL_STORAGE` を持つ任意のアプリがダウンロードしたコンテンツを読み取ることができる。請求書に PII や財務データが含まれる本番環境では、これは直接的な機密性侵害に該当する。

技術的詳細

検出事項 #1

根本原因

`DownloadInvoiceService` は `android:exported="true"` で宣言されており、`onStartCommand()` は呼び出し元の ID や権限、認証状態を検証せずに直ちに `downloadFile()` を呼び出しています。このため、任意のサードパーティアプリケーションがネットワークダウンロードをトリガーし、公開外部ストレージにファイルを書き込むことが可能です。

脆弱なコード

AndroidManifest.xml :

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

DownloadInvoiceService.java (48-83 行目) :

```
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);
}
```

悪用エビデンス

[静的解析] Manifest は Service を明示的にエクスポートしており、ソースコードはダウンロードをキューに入れる前に認可ロジックが存在しないことを確認しています。攻撃者は `AccessControl1ViewActivity` とやり取りすることなく、PIN の入力なしに直接この Service を起動できます。

PoC コマンド :

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

影響 : ダウンロードされたファイルは `/sdcard/Download/<timestamp>` という公開ディレクトリに書き込まれます。タイムスタンプベースのファイル名は予測可能であり、`allowScanningByMediaScanner()` によりファイルは端末のメディアライブラリに表示されます。繰り返しの呼び出しによりストレージや帯域幅を消耗させる可能性があり、ローカル DoS の原因となり得ます。

検証エビデンス

根本原因

`DownloadInvoiceService` は `AndroidManifest.xml` 内で `android:exported="true"` かつ `android:permission` 属性なしで明示的にエクスポートされています。ライフサイクルメソッド `onStartCommand()` (36-42 行目) は、呼び出し元パッケージや権限、Intent の内容を確認せずに `downloadFile()` を直接呼び出しています。`checkCallingPermission`、`Binder.getCallingUid`、`getCallingPackage` などの呼び出し元検証 API を対象としたプロジェクト全体の正規表現スキャ

ンは、`owasp/sat/agoat` パッケージ全体で 0 件の一致を返しており、スーパークラスやユーティリティコードに隠された認可ロジックが存在しないことを確認しています。

脆弱なコード

AndroidManifest.xml :

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

DownloadInvoiceService.java (36-42 行目) :

```
@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 (64-80 行目) :

```
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>
```

悪用エビデンス

[静的解析] Manifest の監査により、Service が権限なしでエクスポートされていることが確認され、逆コンパイルされたソースが報告された脆弱コードと完全に一致することが判明しました。検証により以下の新たな調査結果が明らかになりました :

- **未使用の Intent パラメータ :** `onStartCommand()` は `Intent intent` を受け取りますが、これを一切参照しておらず、入力検証やアクションゲーティングが存在しないことを証明しています。
- **認可 API の欠如 :** プロジェクト全体のスキャンにより、`checkCallingPermission`、`checkCallingOrSelfPermission`、`Binder.getCallingUid`、`Binder.getCallingPid`、`getCallingPackage` の使用が 0 件であることが確認されました。
- **予測可能なファイル名 :** ファイル名は `String.valueOf(System.currentTimeMillis())` であり、±1 分のウィンドウ内で約 120,000 通りの値を取り得ます。これは極めて簡単に総当たり可能です。
- **Media Scanner への露出 :** `allowScanningByMediaScanner()` は `MediaStore.Downloads.EXTERNAL_CONTENT_URI` へのインデックス化を強制し、直接的なファイルシステムアクセスなしにファイルを発見可能にします。
- **タイミングサイドチャネル :** `setNotificationVisibility(1)` は `VISIBILITY_VISIBLE_NOTIFY_COMPLETED` に対応しており、システム通知を表示することでダウンロード完了時刻をロック画面および通知リスナーアクセス権を持つ任意のアプリに漏洩させます。
- **複合的なネットワーク信頼の問題 :** `network_security_config.xml` はユーザーインストール CA (`<certificates src="user" />`) を信頼し、平文通信を許可しています。これにより MITM による傍受やペイロードの置き換えが可能になります。

任意のサードパーティアプリケーションまたはADB シェルから Service を起動可能です：

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

ダウンロードされたファイルは `/sdcard/Download/<epoch_ms>`（公開外部ストレージ）に保存され、`READ_EXTERNAL_STORAGE` を保持する任意のアプリから読み取り可能です。

推奨事項

即時対応

`AndroidManifest.xml` 内の `DownloadInvoiceService` に対して `android:exported="false"` を設定してください。

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

恒久的な修正

外部からの呼び出しが必要な場合は、カスタムパーミッションを用いてサービスを保護してください。

```
<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" />
```

さらに、`onStartCommand()` 内でダウンロードをキューに入れる前に認証チェックを追加し、保存先を `Environment.DIRECTORY_DOWNLOADS` の代わりにアプリのプライベート内部ストレージまたは暗号化されたキャッシュディレクトリに変更してください。

検証

- 1. 異なるUIDを持つシェルから `adb shell am startservice -n owasp.sat.agoat/.DownloadInvoiceService` を実行し、呼び出しが拒否されることを確認してください（`SecurityException`）。
- 2. マニフェストを確認し、`exported="false"` または必要なパーミッションが設定されていることを確認してください。
- 3. ダウンロードしたファイルがパブリックな外部ストレージに書き込まれなくなったことを確認してください。

高 SharedPreferences、SQLite、External Storage にわたる機密認証情報の平文保存

アプリケーションは、ユーザー名、パスワード、およびPINハッシュを複数のローカルストレージメカニズムに永続化しており、暗号化、KeyStore保護、または制限的なファイル権限が設定されていません。

ソースコード解析の結果、`InsecureStorageSharedPrefs` は認証情報を `SharedPreferences("users")` に平文で書き込んでおり、`InsecureStorageSQLiteActivity` は暗号化されていないSQLiteデータベース `aGoat` に保存し、`InsecureStorageTempActivity` は内部ストレージおよび外部ストレージの両方に他アプリから読み取り可能な一時ファイルとして書き込んでいることが判明した。また、アプリは `android:allowBackup="true"` を宣言しているため、これらのファイルは `adb backup` により、あるいは `READ_EXTERNAL_STORAGE` 権限を持つすべてのアプリによって抽出可能である。これにより、ユーザーのアイデンティティに相当する状態情報がローカル攻撃者や悪意あるアプリケーションにさらされ、PINコードのバイパスを必要とせずに完全なアカウントのなりすましが可能となる。

技術的詳細

検出事項 #1

根本原因

複数のストレージ演習 Activity が、ユーザ名、パスワード、PIN ハッシュといった機密ユーザデータを平文形式および弱いファイル権限で永続ストレージに書き込んでいます。コードベース全体に `Cipher`、`KeyStore`、`EncryptedSharedPreferences` の使用箇所は存在しません。

脆弱なコード

`InsecureStorageSharedPrefs.java`（49～52行目）：

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

`InsecureStorageSQLiteActivity.java` (53~58行目) :

```
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` (47~50行目) :

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

悪用エビデンス

[StaticAnalysis]ソースレビューにより、SharedPreferences、SQLite、および一時ファイルにわたる平文の永続化が確認されました。マニフェストはまた

`android:allowBackup="true"` を宣言しており、`adb backup` による抽出が可能です。`InsecureStorageSDCardActivity` は認証情報を
`Environment.getExternalStorageDirectory()` に書き込んでおり、これは `READ_EXTERNAL_STORAGE` を保持するすべてのアプリからアクセス可能です。

検証エビデンス

根本原因

owasp.sat.agoatの複数のストレージ演習Activityが、機密性の高いユーザ認証情報（ユーザ名・パスワードのペア）を暗号化、パラメータ化クエリ、または安全なログ出力なしで永続ストレージに書き込んでいます。AndroidManifest.xmlはさらに、完全なバックアップとランタイムデバッグを有効化することでサンドボックスを弱体化させています。

脆弱なコード

`InsecureStorageSharedPrefs.java` (逆コンパイルした49~56行目) :

```
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` (逆コンパイルした33~38行目) :

```
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` (逆コンパイルした62~73行目) :

```
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` (逆コンパイルした 86 行目) — システムログの漏洩:

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

`AndroidManifest.xml` (5 行目):

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

悪用エビデンス

[Static Analysis] APK の逆コンパイルおよびソースレビューにより、以下の検証済みの調査結果が確認されました。

1. **平文 SharedPreferences の抽出**: `InsecureStorageSharedPrefs` はユーザ名とパスワードを `/data/data/owasp.sat.agoat/shared_prefs/users.xml` に、`"username"` および `"password"` というキーを使用して平文の XML で書き込んでいます。
2. **暗号化されていない SQLite データベース**: `InsecureStorageSQLiteActivity` は `/data/data/owasp.sat.agoat/databases/aGoat` を以下のスキーマで作成します。

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

認証情報はパラメータ化なしの生の `StringBuilder` 連結を介して挿入されており、平文の保存および SQL インジェクションの両方が確認されます。

3. **システムログによる認証情報の漏洩**: 86 行目では、平文のパスワードを含む完全な SQL クエリが `Log.v("Query", qry)` を介して Android システムログに書き込まれます。`READ_LOGS` を持つすべてのアプリ、または `logcat` セッションは、ファイルシステムに触れることなく認証情報を取得できます。
4. **adb backup による抽出経路**: 5 行目のマニフェストは `android:allowBackup="true"` を宣言しており、物理的な USB アクセスを持つ攻撃者が以下を実行することを許可します。

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

生成されたアーカイブは、`shared_prefs/users.xml` および `databases/aGoat` を含むアプリサンドボックス全体を含んでおり、root または PIN のバイパスを必要としません。

5. **debuggable によるランタイム抽出**: 同じマニフェストの行は `android:debuggable="true"` を宣言しており、JDWP デバッグを有効にします。攻撃者は `jdb` または Android Studio をアタッチしてメモリをダンプしたり、`SharedPreferences.Editor.putString()` または `SQLiteDatabase.execSQL()` にブレークポイントを設定したりして、ランタイム時に認証情報を傍受できます。
6. **エクスポートされたコンポーネントによる攻撃面の拡大**: マニフェストの監査により、パーミッション保護を欠く 2 つの明示的にエクスポートされたコンポーネントが特定されました。
 - Receiver `owasp.sat.agoat.ShowDataReceiver` (27 行目)
 - Service `owasp.sat.agoat.DownloadInvoiceService` (37 行目)
7. **INSERT における SQL インジェクション**: `InsecureStorageSQLiteActivity` 内の `StringBuilder` 連結により、不正な入力が入力された SQL 構造を変更することを許容します。たとえば、`admin', 'x'); DROP TABLE users; --` というユーザ名はデータベースを破壊するか、クエリの動作を変更します。

[エビデンスの制限] 検証の実行には、`InsecureStorageTempActivity` および `InsecureStorageSDCardActivity` に対する動的タスクは含まれていません。報告された平文の保存動作は、元の静的解析に基づくものです。

推奨事項

Immediate

アプリケーションのマニフェストで `android:allowBackup="false"` を設定し、バックアップを無効化してください。既存の機密性の高いローカルデータを暗号化するには、AndroidX の `EncryptedSharedPreferences` と `EncryptedFile`、またはデータベース暗号化のために SQLCipher を使用してください。

Permanent Fix

すべての認証情報を、Android の `KeyStore` で保護される `EncryptedSharedPreferences` に移行してください:

```
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
);
```

決して認証情報を外部ストレージや他のアプリから読み取り可能な一時ファイルに書き込まないでください。

Verification

逆コンパイルしたソースコードから `SharedPreferences`、`SQLiteDatabase`、`FileOutputStream` の使用箇所を検索してください。これらのAPIに平文のユーザー名やパスワード文字列が渡されていないことを確認してください。`adb backup` を実行し、認証情報ファイルが読み取れないことを検証してください。

高 ソルトなしMD5 ハッシュで保護された認証情報（AccessControllIssue1Activity）

本アプリケーションはPINをソルトなしのMD5ダイジェストとして内部SharedPreferencesに保存しており、オフラインでのブルートフォース攻撃により平文PINを数ミリ秒で容易に復元可能です。

ハッシュがソルトなしで単一イテレーションであるため、`adb backup`（`allowBackup="true"`）または `run-as`（`debuggable="true"`）を利用して `pinDetails.xml` を抽出した攻撃者は、制限されたPINの鍵空間に対してオフライン辞書攻撃を実行可能です。復元されたPINを用いると正当な認証フローを介した不正アクセスが可能となるほか、deep-link bypassと組み合わせることで権限昇格に繋がる可能性があります。

技術的詳細

検出事項 #1

根本原因

`AccessControllIssue1Activity.hashPIN()` はソルト、イテレーション、キーストレッチングのいずれも行わずに `MessageDigest.getInstance("MD5")` を使用しています。生成されたダイジェストは `SharedPreferences("pinDetails", 0)` のキー `"pin"` に書き込まれています。ソルトなしのMD5でハッシュ化された4～6桁の数字PINは、1秒未満で総当たり検索可能です。

脆弱性のあるコード

AccessControllIssue1Activity.java (111～131行目) :

```
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.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));
                Intrinsics.checkExpressionValueIsNotNull(format,
                    "java.lang.String.format(this, *args)");
                return format;
            }
        }, 30, (Object) null);
    return md;
}
```

SharedPreferencesの利用 (85～89行目) :

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

悪用に関する証拠

【静的解析】ソースコード監査により、ハッシュ処理にソルトがなく、暗号的に破綻したMD5 アルゴリズムが使用されていることが確認されました。概念実証のブルートフォーススクリプトを、同一アルゴリズムで生成されたハッシュに対して実行し、脆弱性の程度を定量化しました：

```
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
```

実際の出力：

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

再計算されたハッシュ `hashlib.md5("7391".encode('utf-8')).hexdigest()` はターゲットのダイジェストを完全に再現しており、保存されたハッシュを取得した攻撃者が平文PINを即座に復元できることを確認しています。

検証結果

根本原因

`AccessControlIssue1Activity.hashPIN()` はソルト、イテレーション回数、キーストレッチングのいずれも行わずに `MessageDigest.getInstance("MD5")` を呼び出しています。生成されたダイジェストは `SharedPreferences("pinDetails", 0)` のキー `"pin"` に書き込まれています。検証の結果、`activity_access_control_issue1.xml` および `activity_access_verify_pin.xml` のいずれも、PIN欄に対して `android:inputType`、`android:digits`、`android:maxLength`、またはプログラムによる検証を適用していないことが判明しました。これは、アプリケーションが入力に対して長さや文字種の制約を課していないことを意味します。`AndroidManifest.xml` は `android:allowBackup="true"` および `android:debuggable="true"` を宣言しており、攻撃者がroot権限なしで `adb backup` または `run-as` を介して `pinDetails.xml` を抽出できるようにしています。

脆弱性のあるコード

AccessControllIssue1Activity.java — `hashPIN()` (111~131行目)：

```
public final String hashPIN(String pinValue) {
    Intrinsics.checkNotNull(pinValue, "pinValue");
    MessageDigest messageDigest = MessageDigest.getInstance("MD5");
    byte[] bytes = pinValue.getBytes(Charsets.UTF_8);
    Intrinsics.checkNotNull(bytes,
        "(this as java.lang.String).getBytes(charset)");
    byte[] digest = messageDigest.digest(bytes);
    Intrinsics.checkNotNull(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));
                Intrinsics.checkNotNull(format,
                    "java.lang.String.format(this, *args)");
                return format;
            }
        }, 30, (Object) null);
    return md;
}
```

SharedPreferences書き込みシーク — `createPIN()` (76~82行目)：

```
public final boolean createPIN(String pinValue) {
    Intrinsics.checkNotNull(pinValue, "pinValue");
    SharedPreferences sharedPreferences = getSharedPreferences("pinDetails", 0);
    SharedPreferences.Editor editor1 = sharedPreferences.edit();
```

```
editor1.putBoolean("pinSet", true);
editor1.putString("pin", hashPIN(pinValue));
return editor1.commit();
}
```

SharedPreferencesの取得および比較 — `isPinCorrect()` (85~88行目) :

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

Null 安全性の欠陥: `"pin"` キーが存在しない状態で `"pinSet"` が `true` のままである場合、`isPinCorrect()` は null 参照に対して `.equals()` が呼び出されるため、`NullPointerException` でクラッシュします。

エクスポートされたバイパスコンポーネント — `AndroidManifest.xml` (20~24行目) :

```
<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>
```

悪用に関する証拠

【静的解析】Manifestの監査により、抽出の前提条件の両方が有効であることが確認されました:

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

Androidのデフォルト完全バックアップは `/data/data/owasp.sat.agoat/shared_prefs/` 以下のすべてのSharedPreferencesファイルを含むため、`pinDetails.xml` はあらゆる `adb backup` アーカイブに含まれます。

【静的解析】脆弱な同一アルゴリズムで生成されたハッシュに対して概念実証のブルートフォーススクリプトを実行した結果、平文PINが即座に復元されました:

```
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
```

実際の出力:

```
Cracked PIN: 7391
Time elapsed: 0.003887 s
```

再計算されたハッシュはターゲットのダイジェストを完全に再現しており、保存されたハッシュを取得した攻撃者がオフラインで数ミリ秒以内にPINを復元できることを確認しています。

【静的解析】 `AccessControl1ViewActivity` は `androgoat://vulnapp` ディープリンクスキームを介してエクスポートされており、直接的なバイパスを提供します:

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

これにより、攻撃者はPINを知らずに保護された画面に到達できます。ハッシュの抽出と組み合わせることで、復元されたPINを正当な `isPinCorrect()` 検証フローに入力することも可能です。

推奨事項

即時対応

`AccessControlIssue1Activity.verifyPINView()` にレート制限とアカウントロックアウトを実装し、ハッシュ実装が再構築されるまでの間、オンライン推測攻撃を遅らせる。

恒久対策

MD5をbcrypt、scrypt、Argon2idなどの現代的なパスワードハッシュ関数に置き換える。ユーザーごとに暗号学的に安全なランダムソルトを使用し、高いワークファクターを設定する。

例（jBCryptを使用したbcrypt）：

```
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);
}
```

検証

- 1. `AccessControlIssue1Activity.java` を精査し、`MessageDigest.getInstance("MD5")` が削除されていることを確認する。
- 2. `hashPIN()` がランダムソルトとコストファクターを含むbcrypt/Argon2idの文字列を返すことを確認する。
- 3. `isPinCorrect()` がライブラリの定数時間検証関数（例：`BCrypt.checkpw`）を使用していることを確認する。
- 4. `pinDetails.xml` に32文字の16進MD5文字列が保存されていないことを確認する。

高 Confirmed Clickjacking on Sensitive Activities in owasp.sat.agoat

SYSTEM_ALERT_WINDOWを持つ悪意あるアプリは、4つの機密性の高いアクティビティ上にインタラクティブなオーバーレイを描画し、ユーザーのタッチ操作を傍受して不正な請求書のダウンロード、プロモーションの検証、OTPの生成、PINの送信を誘発する可能性があります。

攻撃用コンパニオンアプリを用いた動的テストにより、`AccessControl1ViewActivity`、`HardCodeActivity`、`ClipboardActivity` および `AccessControlIssue1Activity` 上の機密性の高いボタンに対して、オーバーレイウィンドウからタッチイベントが伝播し、ユーザーの知情同意なしに特権アクションが実行されることが確認されました。すべての25個のアクティビティにおいて `android:filterTouchesWhenObscured` および実行時のタッチフィルタリングがまったく存在せず、すべてのインタラクティブ要素がタブジャッキングに対して脆弱です。さらに、`AccessControl1ViewActivity` は `androgoat://vulnapp` ディープリンク経由でエクスポートされており、攻撃者は認証後の画面に直接遷移して、事前のPIN検証なしにその画面をクリックジャックできます。

技術的詳細

検出事項 #1

根本原因

本アプリケーションの25個すべてのActivityにおいて、クリックジャッキング対策が完全に欠如している。インタラクティブなViewのいずれも `android:filterTouchesWhenObscured="true"` を宣言しておらず、また `MotionEvent.FLAG_WINDOW_IS_OBSCURED` の検証を伴う `onFilterTouchEventForSecurity()` を実装したActivityも存在しない。`android.permission.SYSTEM_ALERT_WINDOW` を付与された悪意あるアプリケーションは、`TYPE_APPLICATION_OVERLAY` ウィンドウを生成してタッチイベントを傍受し、下層の機密性の高いViewへ伝播させることが可能である。

脆弱なコード

【静的解析】逆コンパイルされたソースに対する網羅的なgrepにより、防御用ウィンドウフラグおよびタッチフィルタリングの該当がゼロ件であることが確認された：

```
# 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
```

`AndroidManifest.xml` はカスタムURLスキーム経由で `AccessControl1ViewActivity` を公開しており、PIN検証なしに認証後のアクションへ直接遷移可能である：

```
<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>
```

エクスプロイトーションの証拠

エミュレータ上に `SYSTEM_ALERT_WINDOW` パーミッションおよび `TYPE_APPLICATION_OVERLAY` ウィンドウを持つ攻撃者側アプリケーション (`com.attacker.overlay`) を展開した。動的テストの結果、4つの機密性の高い画面においてクリックジャッキングが成功することを確認した。攻撃者アプリは傍受したタップごとに `CLICKJACKED!` をログ出力し、対象アプリは下層のアクションを実行した：

対象Activity	機密アクション	オーバーレイ座標	結果
<code>AccessControl1ViewActivity</code>	請求書のダウンロード	x=540,y=393	CLICKJACKED confirmed — ボタンアクションがトリガーされた
<code>HardCodeActivity</code>	プロモーションコードの確認	x=540,y=935	CLICKJACKED confirmed — 検証がトリガーされた
<code>ClipboardActivity</code>	OTPの生成	x=540,y=779	CLICKJACKED confirmed — OTPが生成された
<code>AccessControlIssue1Activity</code>	PINの検証	x=540,y=859	CLICKJACKED confirmed — PINが受理された

確認に使用した攻撃者側オーバーレイの設定：

```
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
);
```

保存された証拠アーティファクト：

- `/workspace/overlay_attack_logcat.txt` — タイムスタンプ付き `CLICKJACKED!` エントリ
- `/workspace/final_attack_demo.mp4` — `HardCodeActivity` クリックジャッキングの画面録画
- `/workspace/clickjacking_pin.mp4` — PIN検証クリックジャッキングの画面録画
- `/workspace/screen_overlay_test2.png` — 請求書ダウンロードオーバーレイのスクリーンショット
- `/workspace/screen_hardcode_overlay.png` — プロモーションコードオーバーレイのスクリーンショット
- `/workspace/screen_clipboard_overlay.png` — OTPオーバーレイのスクリーンショット
- `/workspace/screen_pin_overlay.png` — PINオーバーレイのスクリーンショット

検証証拠

検証試行 1

攻撃者側オーバーレイアプリは、`AccessControl1ViewActivity` の請求書ダウンロードボタン (x=540,y=393) で `CLICKJACKED!` をログ出力し、対象アプリはダウンロードアクションを実行した

検証試行 2

攻撃者側オーバーレイアプリは、`HardCodeActivity` のプロモーションコード確認ボタン (x=540,y=935) で `CLICKJACKED!` をログ出力し、対象アプリは検証を実行した

検証試行 3

攻撃者側オーバーレイアプリは、`ClipboardActivity` のOTP生成ボタン (x=540,y=779) で `CLICKJACKED!` をログ出力し、対象アプリはOTPを生成した

検証試行 4

攻撃者側オーバーレイアプリは、`AccessControlIssue1Activity` のPIN検証 (x=540,y=859) で `CLICKJACKED!` をログ出力し、対象アプリはPINを受理した

検証試行 5

画面録画 `/workspace/final_attack_demo.mp4` および `/workspace/clickjacking_pin.mp4` は、完全なエクスプロイトチェーンを捕捉した

検証試行 6

スクリーンショット

ト `/workspace/screen_overlay_test2.png`、`/workspace/screen_hardcode_overlay.png`、`/workspace/screen_clipboard_overlay.png`、`/workspace/screen_pin_overlay.png` は、オーバーレイの配置を記録している

検証試行

根本原因

本アプリケーションの25個すべてのActivityにおいて、クリックジャッキング対策が完全に欠如している。静的解析の結果、逆コンパイルされたソース全体において `android:filterTouchesWhenObscured="true"`、`onFilterTouchEventForSecurity()`、`FLAG_SECURE` のいずれも検出されなかった。また、`AndroidManifest.xml` 内のintent-filterにより `AccessControl1ViewActivity` が外部に公開されており、ライフサイクルメソッド内で認証チェックを一切行っていないため、PIN検証なしに外部から直接起動可能である。

脆弱なコード

`AccessControl1ViewActivity.java` には `onCreate()` における認証ゲートが存在せず、セッション状態を検証する `onResume()` および `onNewIntent()` のオーバーライドもない：

```
@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);
}
```

`AndroidManifest.xml` は `android:exported="false"` を設定せずにこのActivityを公開している：

```
<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>
```

エクスプロイトーションの証拠

拡張動的オーバーレイテスト： 25個すべてのActivityに対する体系的なテストの結果、**25個中22個**がタップジャッキングに脆弱であることが確認された。攻撃を確認できなかったのは `SplashActivity` および `XSSActivity`（ネイティブのインタラクティブ要素なし）、および `BinaryPatchingActivity`（デフォルトでボタンが無効）のみである。部分的な緩和策は一切検出されなかった。攻撃が確認された高リスクなワークフローは以下の通りである：

- `InputValidationsOSCMDInjectionMain2Activity` — RUNボタンのクリックジャッキングによるOSコマンド実行
- `TrafficActivity` — HTTP/HTTPS/証明書ピンングボタンのクリックジャッキング
- `RootDetectionActivity` / `EmulatorDetectionActivity` — オーバーレイによる整合性チェックのバイパス
- `InsecureStorageSDCardActivity` / `InsecureStorageSQLiteActivity` / `InsecureStorageSharedPrefs` / `InsecureStorageTempActivity` — オーバーレイ経由でトリガー可能な認証情報保存アクション
- `SQLInjectionActivity` — オーバーレイによるSQLクエリ実行

ディープリンク認証バイパス： ADBを使用した動的テストにより、認証後の `AccessControl1ViewActivity` へPIN入力なしに直接アクセスできることを確認した：

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

`dumpsys activity activities` の出力は即座に以下を示した：

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

`am force-stop` 後のコールドスタート検証も成功した：

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

結果のタスクスタック：

```
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}
```

ウォームスタートおよびコールドスタートのいずれにおいてもPINプロンプトまたは認証ダイアログは表示されなかった。また、起動時のlogcatにはPIN検証またはセッションチェックに関するメッセージは一切含まれていなかった。

攻撃者側オーバーレイ設定：

```
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
);
```

検証により保存された証拠アーティファクト：

- `/workspace/dynamic_overlay_comprehensive_report.md`
- `/workspace/traffic_clickjack.mp4` 、 `/workspace/oscmd_clickjack.mp4`
- `/workspace/batch1_results.json` から `/workspace/batch4_results.json` 、および `/workspace/verification_results.json`
- `/workspace/overlay_attack_logcat.txt`
- `/workspace/deeplink_access_control1_view.png` 、 `/workspace/deeplink_cold_start_access_control1_view.png`

推奨事項

即時緩和策

機密性の高いレイアウトXMLファイル内のすべてのクリティカルなインタラクティブビューに `android:filterTouchesWhenObscured="true"` を追加します。これにより、オーバーレイウィンドウから発生するタッチイベントが即座にブロックされます。

```
<Button
    android:id="@+id/downloadInvoiceButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:filterTouchesWhenObscured="true" />
```

恒久的な修正

すべての機密性の高いアクティビティの `onCreate()` で `WindowManager.LayoutParams.FLAG_SECURE` を適用し、画面キャプチャおよびオーバーレイによるリコネクションをブロックします。

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

または、`AndroidManifest.xml` のテーマを介して適用します。

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

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

多層防御を実現するため、ランタイムでタッチイベントのフィルタリングを実装します。

```
@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);
}
```

`AccessControl1ViewActivity.onCreate()` に認可ゲートを追加し、エクスポートされたディープリンクを保護します。

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

検証

1. コンパニオンオーバーレイアプリを再実行し、各機密性の高いボタンをクリックします。対象アプリが応答せず、ログに `FLAG_WINDOW_IS_OBSCURED` が表示されることを確認してください。
2. 保護対象のアクティビティに対して `adb shell screencap` を実行し、ブラックスクリーンまたはセキュリティ例外が発生することを確認してください。
3. `grep -r "filterTouchesWhenObscured" /workspace/decompiled_source/` を実行し、クリティカルなビューでの一致を確認してください。

高 Android Logcatへの生の認証情報およびSQLクエリの漏洩

複数のアクティビティが `Log.e`、`Log.v`、`System.out.println` を使用し、生のユーザー名、パスワード、およびSQL文を無条件にAndroidシステムログに書き込んでいるため、ログへのアクセス権を持つ任意のアプリや `adb` を持つ攻撃者による認証情報の窃取が可能です。

`owasp.sat.agoat` の静的解析により、機密性の高いユーザー入力がマスキングされることなくログシンクに直接流れ込む3つのアクティビティが検出されました。
`InsecureLoggingActivity` はボタンクリックのたびに、生のユーザー名とパスワードをlogcatおよび標準出力に出力します。`InsecureStorageSQLiteActivity` は平文の認証情報を含む完全なSQLINSERT文をログ出力し、`SQLInjectionActivity` は生のSELECTクエリとそのクエリ結果のデータベースレコードの両方を漏洩します。多くの端末でAndroidログはワールドリーダブルであり、`adb logcat` で簡単にアクセスできるため、悪用の複雑性を必要としない高影響度の情報漏洩チャネルとなっています。

技術的詳細

検出事項 #1

根本原因

開発者はユーザー入力およびデータベース操作のデバッグに `android.util.Log` と `System.out.println` を使用しており、文字列がシステムログバッファに書き込まれる前にサニタイズ、条件付きゲーティング、マスキングのいずれも適用されていない。

脆弱なコード

`InsecureLoggingActivity.java` (59行目および70行目) :

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

`InsecureStorageSQLiteActivity.java` (87行目) :

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

`SQLInjectionActivity.java` (82~83行目および93~94行目) :

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

悪用の証拠

[静的解析] 静的監査により、`EditText.getText()` から `StringBuilder` の連結を経由してログシンクに直接至る無条件のデータフローが追跡された。上記のスニペットは `sources/owasp/sat/agoat/` にある逆コンパイル済みソースツリーから原文のまま抽出されている。動的実行は実施されておらず、実行時のlogcat取得は行われなかったが、当該コードパスは無条件であり、呼び出しのたびにデータが出力される。

検証の証拠

根本原因

静的source-to-sink解析により、`owasp.sat.agoat` の3つのアクティビティが生ユーザー認証情報およびSQLクエリ文字列を、サニタイズや条件付きゲーティング、マスキングを一切行わずに `android.util.Log` および `System.out.println` のシンクに直接渡していることが確認された。逆コンパイル済みソースに対する対象絞り込みgrepにより、3つの該当ファイルすべてにおいて `BuildConfig.DEBUG` の参照、`Log.isLoggable()` のチェック、およびマスキング関数 (`replace`、`mask`、`redact`、`substring`) が存在しないことも検証された。さらに `AndroidManifest.xml` では `android:debuggable="true"` が宣言されており、adb logcatへのアクセスが極めて容易な状態となっており、情報漏洩のチャネルを拡大させている。

脆弱なコード

`InsecureLoggingActivity.java` (45~70行目) – `InsecureLoggingActivity$onCreate$1` 内の `onClick` :

```
// 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` (64~87行目) – `try/catch` 外で無条件にログ出力されるSQLINSERT:

```
// 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` (56~94行目) – 平文のSELECTおよび結果セットのログ出力:

```
// 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
```

悪用の証拠

[静的解析]検証では3つのアクティビティすべてに対して正式な source-to-sink トレースを実施し、`EditText.getText()` からログシンクへの無条件のデータフローを確認した：

- InsecureLoggingActivity** — 54行目および58行目で生のユーザー名とパスワードを `StringBuilder sb` に取り込み、59行目で `Log.e` により出力している。65行目および69行目では2番目のビルダにデータが格納され、70行目で `System.out` に出力されている。いずれのシンクも `if (DEBUG)` による保護が存在しない。
- InsecureStorageSQLiteActivity** — 87行目の `Log.v("Query", qry)` は `try/catch` ブロック（75〜86行目）の後に配置されており、`execSQL` が成功しても例外をスローしても実行される。SQLINSERT文字列全体には平文のパスワードが含まれている。
- SQLInjectionActivity** — 82〜83行目は `getCount() > 0` の場合に結果セット全体（平文のデータベースパスワードを含む）をログ出力する。93〜94行目はボタンクリックのたびに無条件に実行され、生のSELECTクエリと、結果セットのレコード、またはユーザー入力に「notfound」サフィックスを付加した内容を含むバッファを出力する。

`AndroidManifest.xml` の監査により、さらに `android:debuggable="true"`（5行目）が確認された。これにより、USB/adb アクセスを持つ攻撃者が `adb logcat` を実行し、出力された認証情報およびSQL文を、デバイスのルート化やlogcatアクセス制限の迂回を必要とせず取得可能となるため、実機での悪用が著しく容易になっている。

証拠の制限事項： 今回の検証では動的実行および実行時のlogcat取得は実施されておらず、被攻撃性は静的な逆コンパイルソース解析のみで確認された。

推奨事項

即時対応

ユーザー認証情報やSQLクエリを処理している該当ファイルから、`Log.e`、`Log.d`、`Log.v`、および `System.out.println` のすべての呼び出しを削除またはコメントアウトしてください。

恒久的な修正

機密性の高いパターンを検査し、出力前に `[REDACTED]` に置き換える集中管理型のログユーティリティを実装してください。ProGuardまたはR8のルールを使用して、リリースビルドから `Log.*` の呼び出しを削除してください：

```
-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(...);
}
```

検証

アプリをリリースモードでビルドし、認証情報入力フローを実行した上で、`adb logcat | grep -i "Error\|Query"` を実行してください。ログにユーザー名、パスワード、またはSQLクエリのテキストが出力されていないことを確認してください。

高 InsecureStorageSQLiteActivity における未サニタイズ execSQL による SQL インジェクション

InsecureStorageSQLiteActivity は、サニタイズされていない2つのEditTextの値を連結してexecSQL()に渡すことでINSERT文を構築しており、これによりスタックド・クエリ・インジェクション、レコードの任意改変、およびローカルデータベースの完全な整合性侵害が可能となります。

`InsecureStorageSQLiteActivity` は、ユーザーの認証情報をローカルの `aGoat` SQLite データベースに保存するアクティビティである。攻撃者が制御可能な2つの `EditText` フィールドの文字列を直接連結して `INSERT` 文を構築し、生成された文字列を `SQLiteDatabase.execSQL()` に渡す。このメソッドはセミコロンで区切

られた複数のSQL文を受け入れるため、攻撃者は元の `INSERT` を早期に終了させ、`INSERT` ・ `UPDATE` ・ `DELETE` ・ `CREATE TABLE` を含む任意の追加SQL文を実行可能となる。ユーザー名およびパスワードの両フィールドが脆弱なインジェクションポイントであり、永続的な書き込みが行われるため、整合性の侵害は即座かつ継続的となる。本アクティビティは `SQLInjectionActivity` と同一のデータベースファイルを共有しており、インジェクションされたデータが読み取り指向のクエリに影響を与える。

技術的詳細

検出事項 #1

根本原因

`owasp/sat/agoat/InsecureStorageSQLiteActivity.java` の80行目で、アプリケーションは無害化されていない2つのUI入力をSQL文字列に直接埋め込み、`INSERT` 文を構築して `execSQL()` で実行している。`rawQuery()` とは異なり、`execSQL()` は単一ステートメントへの実行を制限しないため、スタック型クエリインジェクションが可能である。

脆弱なコード

静的解析で特定された脆弱なクエリ構築パターンは以下の通りである：

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

`userName` と `password` の両方は `EditText` ウィジェット（`R.id.userName` および `R.id.password`）から取得されており、入力検証やパラメータ化は行われていない。

悪用エビデンス

[シミュレーション] 正確な `aGoat` スキーマを持つローカルのシミュレーションSQLiteデータベースを使用し、再構成されたペイロードを実行した。以下の結果は任意のステートメント実行を確認するものである：

- **ユーザー名フィールドを用いた引用符エスケープ／早期終了：** 入力 `test','test');` `--` は、データベースに対して `INSERT INTO users (username, password) VALUES('test','test');` `--'` を実行させ、事実上 `(test, test)` を挿入し、残りをコメントアウトした。
- **ユーザー名フィールドを用いたマルチステートメントインジェクション：** 入力 `test','test');` `INSERT INTO users (username, password) VALUES('attacker','attacker');` `--` は、2つの独立したステートメントが実行される結果となった。シミュレーションにより2行が挿入されたことが確認され、任意の二次ステートメント実行が証明された。
- **パスワードフィールドを用いたマルチステートメントインジェクション：** 入力 `test');` `INSERT INTO users (username, password) VALUES('attacker2','attacker2');` `--` もまた2行の挿入が確認され、両方の入力フィールドが独立して脆弱であることが確認された。
- **UPDATEによる整合性侵害：** `INSERT` の後に `UPDATE users SET password='hacked' WHERE username='admin'; --` というスタック型ペイロードを付加した。シミュレーションにより `admin` のパスワードが恒久的に `hacked` に変更されたことが確認された。
- **CREATETABLEによるデータ窃取：** スタック型ペイロードにより新規テーブル `stolen_data` を作成し、既存の認証情報をすべて挿入した。これにより、攻撃者が制御するスキーマオブジェクトを介した完全なデータ窃取（exfiltration）が確認された。

検証エビデンス

根本原因

[静的解析] JADXでデコンパイルされたソース

`/workspace/source_code_analysis_b456ad45d2f249a09fc5ea8e695da9e6/base.apk/sources/owasp/sat/agoat/InsecureStorageSQLiteActivity.java` は、80行目に脆弱なシンクが存在することを確認している。`onCreate()` メソッド（44～90行目）は、`R.id.userName`（58行目）および `R.id.password`（59行目）の `EditText` ハンドルを取得し、保存ボタン（`R.id.SQLButton`）のクリックリスナー（60行目）を設定する。`onClick()`（62行目）内部では、直接の文字列連結によりクエリを構築した後、`this.mDB.execSQL(qry)` に渡している。`?` プレースホルダー、`bindArgs`、および中間的な無害化処理は一切存在しない。

脆弱なコード

JADXによりKotlinからトランスパイルされたデコンパイル済みJavaソースは、正確な連結パターンを以下に示す：

```
// 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);
```

検証により、`Log.v("Query", qry)`（81行目）および`Log.d("Error", ...)`（84行目）が完全なクエリ文字列（インジェクトされたペイロードを含む）をシステムログに書き込むことも確認された。これは `READ_LOGS` または物理的なADBアクセスを持つ攻撃者に対する追加のサイドチャネルを提供する。

悪用エビデンス

[静的解析] 検証により、両方の入力フィールドが検証、エスケープ、またはホワイトリストのロジックなしに独立して脆弱なシンクに到達することが確認された。レイアウトXML（`activity_insecure_storage_sqlite.xml`）は `R.id.userName` および `R.id.password` をソースの `EditText` ウィジェットに直接バインドしており、80行目からの後方トレースは防御的制御がゼロであることを示している。

クロスアクティビティの影響は、`SQLInjectionActivity.java`（45～97行目）の解析により検証された。このアクティビティは同一のデータベースファイルをオープンする：

```
// 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);
```

両方のアクティビティが同一の `"aGoat"` データベースを参照し、`execSQL()` は即座にディスクへコミットするため、`InsecureStorageSQLiteActivity` を介して実行される整合性を損なうスタック型ペイロード（例：`UPDATE`、`DELETE`、`CREATE TABLE`）は、`SQLInjectionActivity` の `SELECT` クエリに即座に反映される。

マニフェストの監査はさらに影響を拡大する：`android:allowBackup="true"` および `android:debuggable="true"` は `adb backup` または `run-as` による `aGoat` データベースの抽出とオフライン改竄を許容する。さらに、`cleartextTrafficPermitted="true"` および `network_security_config.xml` におけるユーザーCAの信頼設定により、ネットワーク傍受が容易になる。

推奨事項

緊急の緩和策

恒久的な修正をデプロイするまでの一時的な対応として、`Save` ボタンを無効化するか、クライアント側での入力文字数制限を追加してください。マニフェストから `android:debuggable="true"` を削除してください。

恒久的な修正

`execSQL()` の呼び出しを `ContentValues` を使用する `SQLiteDatabase.insert()` に置き換え、文字列連結を完全に排除してください：

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

アプリケーションがスキーマ管理に `execSQL()` を使用しなければならない場合、ユーザー入力を含む文字列で `execSQL()` が決して呼び出されないようにしてください。すべてのデータ操作は、パラメータ化API（`insert()`、`update()`、`delete()`）またはバインドパラメータを持つ `SQLiteStatement` を使用するようになしてください。

検証

- 静的解析を再実行し、ユーザー入力を連結した `execSQL()` が一件も存在しないことを確認してください。
- 修正済みビルドに対して `'); DELETE FROM users; --` などのマルチステートメントペイロードを試行し、意図した単一行の挿入のみが行われることを確認してください。
- 各テスト操作後に `users` テーブルを確認し、予期しないスキーマ変更や追加の行が出現しないことを確認してください。

高 インターネット公開されたFirebase Database

Firebaseデータベースを公開状態にしていると、不正アクセス、データ改ざん、情報漏洩、悪意ある攻撃、および信用失墜を招く可能性があります。

Firebaseは、モバイルおよびWebアプリケーション開発プラットフォームであり、リアルタイムデータベースを含む各種ツールとサービスを開発者に提供しています。Firebaseデータベースが公開されている場合、認証や認可なしに誰でもデータベースに格納されたデータにアクセスし、操作できることを意味します。

Firebaseデータベースを公開することのセキュリティ影響は深刻であり、以下のセキュリティリスクを招く可能性があります。

- 不正アクセス：認証および認可なしに誰でもデータベース内のデータにアクセス可能となるため、データベースに格納された機密情報が権限のないユーザーに閲覧される可能性があります。
- データ改ざん：攻撃者がデータベース内のデータを変更、追加、削除できるため、データの欠損や不正確なデータが生じ、アプリケーションやビジネスに重大な影響を及ぼす可能性があります。
- 情報漏洩：攻撃者がパスワード、個人情報、財務データ、その他の機密情報など、データベースに格納された機密情報にアクセスできます。
- 悪意ある攻撃：攻撃者はインジェクション攻撃、クロスサイトスクリプティング（XSS）、その他の攻撃を仕掛け、Firebaseデータベースに接続されたアプリケーションの脆弱性を悪用する可能性があります。
- 信頼の喪失：機密情報が漏洩すると、顧客からの信頼と信用を失い、ビジネスの評価に重大な影響を及ぼす可能性があります。

Firebaseデータベースが公開されているかどうかを確認するには、以下のcurlコマンドを使用します。

```
<> BASH

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

<project-id> を確認対象のFirebaseプロジェクトIDに置き換えてください。このコマンドは、FirebaseデータベースのルートノードをJSON形式で取得しようとします。

Firebaseデータベースが公開されている場合、認証や認可なしにデータを取得できるはずですが、

データベースが安全に適切に構成されている場合、データへのアクセスが認可されていないことを示すエラーメッセージが返されます。

技術の詳細

検出事項 #1

Firebase `https://androgoat-42597.firebaseio.com` は **公開状態で** アクセス可能であり、以下のコマンドで検証できます：`curl --request GET --url https://androgoat-42597.firebaseio.com/.json`。キーは `res/values/strings.xml` で検出されました。

推奨事項

Firebase データベースを非公開にするには、以下の技術的な手順に従ってください。

- Firebase Authentication を有効化：Firebase Authentication を使用すると、ユーザーを認証し、データベースへのアクセスを制御できます。メールアドレスとパスワード、Google、Facebook など、さまざまなプロバイダーを利用してユーザーの認証を行うことができます。
- Firebase Security Rules の設定：Firebase Security Rules を使用すると、データへのアクセス方法や、データベースの読み書きを許可するユーザーを定義できます。ユーザー認証やデータベース内の特定のデータパスに基づいてルールを定義できます。

参考資料

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

高 LOGJAM 共通素数の脆弱性

広く共有されたDiffie-Hellman素数の使用により、インターネットトラフィックの大部分に対する国家レベルの事前計算攻撃が可能となる。

この脆弱性は、サーバーが広く共有されているDH素数を使用しており、効率的な事前計算攻撃を可能にし、国家レベルの脅威アクターがインターネット上の大量のトラフィックを受動的に復号できることを示している。

LOGJAM-common_primesは、数百万もの環境で共有されている標準的なDHパラメーター（素数）をサーバーが使用する場合に発生する。共有素数の使用は安全と見なされていたが、数体ふるい法（numberfield sieve）により、攻撃者は各素数に対して高コストな暗号テーブルの事前計算を一度行うだけで、その素数を使用するあらゆる接続を迅速に解読できる。

仕組み：

- 攻撃者が広く使用されているDH素数を特定する（例：ApacheやOpenSSLのデフォルトパラメーター）
- 対象の素数に対して高コストな事前計算（数週間の計算時間）を実行する
- その素数を使用するあらゆるサーバーからDH鍵交換を受動的に傍受する
- 事前計算済みのテーブルを使用して、個別の接続を数分で解読する
- 同じ素数を共有する数百万台のサーバーに対して攻撃をスケールさせる

前提条件：

- サーバーが共通またはデフォルトのDH素数（1024ビット以下）を使用している
- 初期事前計算に相当な計算リソースが必要（1024ビットの場合、約1億ドル）
- 対象の接続からネットワークトラフィックを取得できること
- 中間者攻撃の能力は不要であり、完全に受動的な攻撃である

攻撃シナリオの例：大手クラウドプロバイダーが数千台のサーバーにわたってデフォルトの1024ビットDHパラメーターを使用しているとする。国家レベルの脅威アクターは、この特定の素数に対する暗号テーブルの事前計算に数か月を費やす。完了後、同アクターはインターネットトラフィックを受動的に監視し、これらのサーバーへのあらゆるTLS接続をリアルタイムで復号できる。これにより、能動的な干渉や検知を伴うことなく数百万のユーザーに影響が及ぶ。

最も一般的な1024ビット素数の1つを破壊すれば、上位HTTPSドメインの18%に対する受動的な盗聴が可能となり、最も一般的な素数の2つを破壊すれば、世界中のVPNサーバーの66%およびSSHサーバーの26%が侵害される。

技術的詳細

検出事項 #1

LOGJAM-common_primesが `demo.testfire.net` で検出されました

- 発見内容：RFC2409/Oakley Group 2

推奨事項

LOGJAM共通素数攻撃を軽減するには：

主要な対策—固有のDHパラメータを生成する：

```
# Generate unique 2048-bit DH parameters
openssl dhparam -out unique-dhparams.pem 2048

# Apache: use custom parameters
SSLOpenSSLConfCmd DHParameters /path/to/unique-dhparams.pem

# Nginx: specify unique parameters
ssl_dhparam /path/to/unique-dhparams.pem;
```

デフォルトおよび共通の素数の使用を避ける：

以下の広範に侵害されている素数は使用しないこと：

- デフォルトのApache 512ビット素数（数百万台のサーバーで使用されている）
- デフォルトのOpenSSL 512ビット素数（1995年からSSLeayに含まれている）
- RFC 5114標準素数（NSAの影響を受けている可能性がある）

ECDHEへの移行：

最善の解決策—DHの脆弱性を完全に排除する：

```
# Apache: prioritize ECDHE over DHE
SSLCipherSuite ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:!DHE

# Nginx: use only ECDHE ciphers
ssl_ciphers ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:!DHE;
```

最低2048ビットのパラメータを使用する：

DHEが必須の場合：

```
# Ensure minimum key size
ssl_dhparam_size 2048; # Nginx
```

共通素数のテスト：

```
# Extract DH parameters from server
openssl s_client -connect example.com:443 -cipher DHE 2>&1 | \
  openssl dhparam -inform PEM -text -noout

# Compare against known vulnerable primes
# Check if prime matches Apache default or other common values
```

追加の保護策：

- TLS1.3を使用し、有限体DHを完全に排除する
- MITM試行を検出するため、証明書ピンニングを実装する
- 大量トラフィック解析を示す異常な接続パターンを監視する
- 暗号スイートの選択時にPerfect Forward Secrecyの影響を考慮する

重要な知見：たとえ強固な1024ビットDHパラメータであっても、多数のサーバー間で共有されると事前計算コストが償却されるため脆弱となる。一意性は、鍵長と同等に重要である。

参考資料

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

高 モバイルSQLインジェクション脆弱性

モバイルSQL Injection脆弱性は、攻撃者がモバイルアプリケーションのデータベースに対して悪意のあるSQLクエリを実行可能にし、不正アクセス、データ漏洩、またはデータ改ざんにつながる可能性があります。

モバイルSQL Injectionとは、攻撃者がモバイルアプリケーションに悪意のあるSQLステートメントを注入することで、機密データへの不正アクセスやデータベースの改ざんを可能にする脆弱性です。

例

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()
}
```

技術的詳細

検出事項 #1

クラス `SQLiteDatabase` のメソッド `execSQL` の位置 `0` のパラメータはSQLインジェクションの脆弱性がある可能性があります。非到達可能コンポーネント `owasp.sat.agoat.InsecureStorageSQLiteActivity` からTaintがトレースされました。TaintElementの発生元：クラス `InsecureStorageSQLiteActivity`、メソッド `onCreate`、位置 `0`、シグネチャ `(Landroid/os/Bundle;)V`。コールトレース：

```
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
```

位置 `android.database.sqlite.SQLiteDatabase.execSQL`

ソースコードを抽出できません。

位置 `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.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(',');
    String v1_2 = this.$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.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;
}
```

位置 `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.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(',');
    String v1_2 = this.$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.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;
}

```

位置 `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;
}

```

位置 `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.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v1_8, username);
    v0_1.append(v1_8.getText().toString());
    v0_1.append(',');
    String v1_2 = this.$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.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;
}

```

位置 `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;
}

```

位置 `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;
}

```

位置 `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;
}

```

位置 `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;
}

```

```
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;
}
}
```

検出事項 #2

ユーザー制御パラメータがSQLインジェクションに対して脆弱なSQLパラメータの構築に使用されています。到達不可能なコンポーネント：

`owasp.sat.agoat.InsecureStorageSQLiteActivity` からTaintがトレースされ、`owasp.sat.agoat.InsecureStorageSQLiteActivity` で発見されました。パラメータ `0` からシンク `void android.database.sqlite.SQLiteDatabase.execSQL(java.lang.String)` まで。Taintトレース：

```
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()
```

メソッド `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;
}

```

メソッド `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;
}

```

メソッド `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 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;
}

```

メソッド `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;
}

```

メソッド `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;
}

```

メソッド `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;
}

```

メソッド `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;
}

```

検出事項 #3

`SQLiteDatabase` クラスの `execSQL` メソッドの位置 ⑥ におけるパラメータは、SQL インジェクションに脆弱な可能性があります。到達不能なコンポーネント `owasp.sat.agoat.InsecureStorageSQLiteActivity` からテイントが追跡されました。 `TaintElement` の発生源：クラス `InsecureStorageSQLiteActivity` 、メソッド `onCreate` 、位置 ⑥ 、シグネチャ `(Landroid/os/Bundle;)V` 。コールトレース：

```

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

```

`android.database.sqlite.SQLiteDatabase.execSQL` において

Unable to extract source code.

owasp.sat.agoat.InsecureStorageSQLiteActivity\$onCreate\$1.onClick において

```
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;
}
```

android.database.sqlite.SQLiteDatabase.execSQL において

Unable to extract source code.

owasp.sat.agoat.InsecureStorageSQLiteActivity\$onCreate\$1.onClick において

```
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;
    }
}

```

`owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick` において

```

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.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;
}

```

`owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.onClick` において

```

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.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;
}

```

`owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate` において

```

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;
}

```

`owasp.sat.agoat.InsecureStorageSQLiteActivity$onCreate$1.<init>` において

```

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;
}

```

`owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate` において

```

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;
}

```

`owasp.sat.agoat.InsecureStorageSQLiteActivity.onCreate` において

```

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;
}

```

検出事項 #4

ユーザー制御可能なパラメータが、SQLインジェクションに対して脆弱なSQLパラメータの構築に使用されています。到達不可能なコンポーネントからの汚染トレース：`owasp.sat.agoat.SQLInjectionActivity` が `owasp.sat.agoat.SQLInjectionActivity` で検出されました。パラメータ `0` からシンク `android.database.Cursor android.database.sqlite.SQLiteDatabase.rawQuery(java.lang.String, java.lang.String[])` まで。汚染トレース：

```

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()

```

メソッド `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;
}

```

メソッド `owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p0)
{
    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.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;
}

```

メソッド `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;
}

```

メソッド `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 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;
}

```

メソッド `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;
}

```

メソッド `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;
}

```

メソッド `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;
}

```

メソッド `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;
}

```

メソッド `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;
}

```

検出事項 #5

ユーザ制御パラメータがSQLインジェクションに脆弱なSQLパラメータの構築に使用されています。到達不能なコンポーネントからTaintがトレースされました:

`owasp.sat.agoat.SQLInjectionActivity` が `owasp.sat.agoat.SQLInjectionActivity` で検出されました。パラメータ `0` からシンク `android.database.Cursor` `android.database.sqlite.SQLiteDatabase.rawQuery(java.lang.String, java.lang.String[])` まで。Taintトレース:

```

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()

```

メソッド `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.$username;
    kotlin.jvm.internal.Intrinsics.checkNotNullValue(v1_6, username);
    v0_1.append(v1_6.getText().toString());
    v0_1.append(' ');
    String v0_2 = v0_1.toString();
    try {
        owasp.sat.agoat.SQLInjectionActivity.access$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;
}

```

メソッド `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;
}

```

メソッド `owasp.sat.agoat.SQLInjectionActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p0)
{
    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;  
}
```

メソッド `owasp.sat.ag oat.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$OnClickListener) new  
owasp.sat.ag oat.SQLInjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));  
    return;  
}
```

メソッド `owasp.sat.ag oat.SQLInjectionActivity$onCreate$1.<init>()` :

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

メソッド `owasp.sat.ag oat.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$OnClickListener) new  
owasp.sat.ag oat.SQLInjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));  
    return;  
}
```

メソッド `owasp.sat.ag oat.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$OnClickListener) new  
owasp.sat.ag oat.SQLInjectionActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)))));  
    return;  
}
```

検出事項 #6

ユーザー制御可能なパラメータが、SQLインジェクションに脆弱なSQLパラメータの構築に使用されています。到達不可能なコンポーネント

`owasp.sat.ag oat.SQLInjectionActivity` から Taint がトレースされ、`owasp.sat.ag oat.SQLInjectionActivity` で検出されました。パラメータ `@` からシンク `android.database.Cursor android.database.sqlite.SQLiteDatabase.rawQuery(java.lang.String, java.lang.String[])` まで。Taint トレース :

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

```
at owasp.sat.agoat.SQLInjectionActivity.onCreate()
at owasp.sat.agoat.SQLInjectionActivity.onCreate()
```

メソッド `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;
}
```

メソッド `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.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;
}

```

メソッド `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;
}

```

メソッド `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;
}

```

メソッド `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;
}

```

メソッド `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;
}

```

メソッド `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;
}

```

メソッド `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;
}

```

```
}
```

推奨事項

Mobile SQL Injection脆弱性を軽減するため、以下を検討してください。

- SQLコードとユーザー入力を分離するため、パラメータ化クエリまたはプリペアドステートメントを使用する。
- セカンドオーダーSQLインジェクションを軽減するため、データベースに挿入する前にユーザー入力をサニタイズおよび検証する。
- 既知の脆弱性に対処するため、SQLドライバーを定期的に更新する。

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],
    );
  });
}

```

参考資料

- [OWASP Mobile Security Testing Guide \(MSTG\)](#)

中 クライアントサイドのブール値フラグで管理者権限が決定され、サーバー側での検証が行われていない

BinaryPatchingActivityはローカルのブール値フィールドを使用して管理UI機能へのアクセスを制御していますが、このフィールドはバックエンドAPI、暗号化トークン、再認証チャレンジのいずれでも検証されていません。

Activityは `private final boolean isAdmin` フィールドを `false` に初期化し、`onCreate()` 内で管理者ボタンの有効化および管理者テキストの表示を判定するために使用しています。サーバーサイドでのロールチェック、署名検証、ステップアップ認証のいずれも存在しないため、攻撃者はランタイム時にFridaを使用してこのフラグを切り替えるか、APKのDalvikバイトコードを改竄することで権限を昇格させることができます。これはクライアントサイドの信頼のみに依存し、二次的な検証が一切不要な権限昇格の経路を示しています。

技術的詳細

検出事項 #1

根本原因

`BinaryPatchingActivity` は `private final boolean isAdmin` フィールドを使用してadminUI要素へのアクセスを制御している。このフィールドは `false` に初期化され、`onCreate()` 内で判定に使用される。値が `true` の場合、adminボタンが有効化されTextViewが更新されるが、admin状態を検証するバックエンドAPI呼び出し、暗号学的証明、およびステップアップ認証は存在しない。

脆弱なコード

`BinaryPatchingActivity.java` (17行目、39〜54行目) :

```

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);
}

```

悪用の証拠

[静的解析] このフィールドはコンパイル時定数であり、リモート検証は存在しない。攻撃者はFridaを使用して `BinaryPatchingActivity.isAdmin()` をフックし `true` を返すようにできるほか、APKのバイナリパッチによりDalvikの初期化処理を `const/4 v0, 0x0` から `0x1` に変更できる。バックエンドは一切問い合わせられず、admin UI が有効化されても再認証はトリガーされない。

検証証拠

根本原因

`BinaryPatchingActivity` (パッケージ `owasp.sat.agoat`) は `private final boolean isAdmin` フィールドを使用して admin UI 要素へのアクセスを制御している。このフィールドは逆コンパイルされたJavaソース `base.apk/sources/owasp/sat/agoat/BinaryPatchingActivity.java` の17行目で宣言され、JVMのデフォルト値である `false` が設定され、`final` 修飾子によりコンストラクタ実行後は不変となる。`onCreate()` (48行目) で Activity は `this.isAdmin` を判定し、`true` の場合に `adminButton` を有効化し `isAdminText` を更新する。検証の結果、この Activity および到達可能なコールグラフ内に、外部入力ベクトル、ネットワークリクエスト、暗号学的証明、署名検証、およびステップアップ認証は一切存在しないことが確認された。

脆弱なコード

`BinaryPatchingActivity.java` (17行目、39～54行目) :

```
// 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);
}
```

`isAdmin` が `false` (デフォルト) の場合、レイアウト `activity_binary_patching.xml` は `isAdminText` に "Sorry, You don't have access to this feature." を表示し、`adminButton` は `android:enabled="false"` のまま無効となる。admin 状態をリモート側の権限機関で検証する代替コードパスは存在しない。

悪用の証拠

[静的解析] ソースコード監査により、`isAdmin` は合成コンストラクタ内で暗黙的に `false` に一度だけ代入され、その後は一切変更されていないことが確認された。`isAdmin\s*` というパターンをプロジェクト全体で検索した結果、一致はゼロであり、setter、リフレクション、またはネイティブコードによるフラグ変更は存在しないことが確認された。

マニフェスト監査の結果、`base.apk/AndroidManifest.xml` の5行目で以下が判明した：

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

`android:debuggable="true"` により、Fridaやその他のインスツルメンテーションツールがrootアクセス不要でアタッチ可能となり、攻撃者が実行時に `BinaryPatchingActivity.isAdmin()` をフックして `true` を強制的に返す障壁が大幅に低下する。この Activity は `android:exported` が `false` (デフォルト) のためエクスポートされていないが、アプリがローカルで実行されていれば、攻撃者はDalvikのイニシャライザを `const/4 vX, 0x0` から `const/4 vX, 0x1` にパッチすることで、起動するたびに admin UI を恒久的に有効化できる。admin インターフェースが有効化されても、バックエンドでの検証、再認証、および整合性チェックは一切トリガーされない。

推奨事項

即時対応

`BinaryPatchingActivity` からクライアントサイドの `isAdmin` フラグを削除する。その代わりに、認証済みTLS経由でバックエンドの `/user/role` エンドポイントに問い合わせ、取得したロールを短いTTL付きで `EncryptedSharedPreferences` にキャッシュする。

恒久的な修正

すべての特権アクションに対してサーバーサイドでのルール適用を実装する。クライアントは署名付きJWTまたはセッショントークンをバックエンドに送信し、adminデータの返却やadmin機能の有効化に先立ってロールクレームを検証するようにする：

```
// クライアントサイド：UIゲートのみ。バックエンドで認可を適用
// サーバーサイド：処理前に token.role == "admin" を検証
```

検証

Fridaを使用して `BinaryPatchingActivity.isAdmin()` をフックし、`true` を返すようにする。adminアクションを試行しても、バックエンドがリクエストを拒否するため失敗し続けることを確認する。また、ソースコードをレビューし、ローカルのブール値だけでは特権UI要素を有効化できないことを確認する。

中 Certificate Pinning in TrafficActivity Silently Fails Due to Broad Exception Handling

`TrafficActivity.doPinning()` 内の `OkHttpCertificatePinner` は、広範な `catch` ブロックによって `SSLPeerUnverifiedException` が握り潰されているため無効化されており、検知されない中間者攻撃を許容しています。

`TrafficActivity.doPinning()` は、`owasp.org` に対して3つのSHA-256ピンを設定した `OkHttp` の `CertificatePinner` を実装しています。しかしながら、ネットワークリクエスト全体が `try { ... } catch (Exception e) { e.printStackTrace(); }` で囲まれており、証明書ピンニングの失敗時に発生する `SSLPeerUnverifiedException` を握り潰してしまいます。さらにグローバルなユーザーインストールCAの信頼設定と組み合わせることで、攻撃者はプロキシ証明書をういてこのピン留めされた接続を傍受でき、ユーザーには一切通知されません。

技術的詳細

検出事項 #1

根本原因

開発者は `OkHttp` の `CertificatePinner.Builder` を用いて証明書ピンニングを追加しましたが、ピン留め対象のリクエストを広範な例外ハンドラで包み込み、安全に失敗させることなくスタックトレースを出力しています。これによりピンの不一致が発生してもアプリケーションが暗黙的に実行を継続するため、ピンニングのセキュリティ保証が損なわれます。

脆弱なコード

`TrafficActivity.java` 140～141行目：

```
CertificatePinner pinner1 = new CertificatePinner.Builder()
    .add("owasp.org", "sha256/gdU/UHClHJBFBIdeKuyHm/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` 147～148行目：

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

悪用エビデンス

[Static Analysis] 逆コンパイル済みソースによれば、ピンニングロジックは `doPinning()` 内に閉じており、ピン留め対象は `owasp.org` のみです。このアプリのNetwork Security Configは既にユーザーインストールCAを信頼しているため、TLSハンドシェイクはプロキシ証明書を受け入れます。ピンナーはその後 `SSLPeerUnverifiedException` をスローしますが、147～148行目の広範な `catch (Exception e)` ブロックがこれを見逃して実行を継続します。また、`okhttp3.CertificatePinner.check()` へのFridaフックによってもピンチェックを無力化できます：

```
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);
    };
});
```

検証エビデンス

根本原因

検証により、`owasp/sat/agoat/TrafficActivity.java` 139～149行目の `TrafficActivity.doPinning()` が、アプリケーション全体で唯一の証明書ピンニング実装箇所であることが確認されました。同期 `execute()` 呼び出しを単一の広範な `try` ブロックで包み、`catch (Exception e) { e.printStackTrace(); }` で閉じています。`SSLPeerUnverifiedException` は `Exception` のサブクラスであるため、ピンの不一致は決定論的に吸収され、安全に失敗しません。このアプリの `AndroidManifest.xml` は `android:networkSecurityConfig="@xml/network_security_config"` を参照しており、対応する `res/xml/network_security_config.xml` には `<certificates src="user"/>` が含まれ、OSにユーザーインストールCAの信頼を指示します。これにより、OkHttpのピンナーが拒否する前にプロキシ証明書がTLSハンドシェイクを通過できるようになります。

脆弱なコード

逆コンパイル済みソースツリー `/workspace/source_code_analysis_dfc434d7ae9421693bf4c0774f91126/base.apk/sources/owasp/sat/agoat/TrafficActivity.java` で検証済み：

139～146行目：

```
try {
    CertificatePinner pinner1 = new CertificatePinner.Builder()
        .add("owasp.org", "sha256/gdU/UHC1HJBfIdeKuyHm/Lq/aQVMLyuTtcvTEE/1JQ=")
        .add("owasp.org", "sha256/YLh1dUR9y6Kja30RrAn7JKnbgQ6/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();
}
```

同じパターンが別の逆コンパイル結果 `TrafficActivity$doPinning$1.java` 28行目にも反映されています。

悪用エビデンス

[Static Analysis] すべての逆コンパイル済みアプリソース

(`/workspace/jadx_out/sources`、`/workspace/jadx_out2/sources`、`/workspace/decompiled_source/sources`、および関連する解析ディレクトリ)を網羅的に検索した結果、`CertificatePinner.Builder` のインスタンス化は `TrafficActivity.doPinning()` およびその匿名内部クラス `TrafficActivity$doPinning$1.invoke2()` 内でのみ行われ、ピン留め対象のホスト名は常に `owasp.org` のみであることが確認されました。コードベース全体に代替ピンニングライブラリ (TrustKit、AndroidPinning)、カスタム `TrustManager`、`HostnameVerifier`、または `SSLContext` の操作は存在しません。唯一の `OkHttpClient.Builder.certificatePinner()` 呼び出し箇所は `TrafficActivity.java` の141行目です。

このメソッドは、UIボタンクリック `R.id.PinningButton` によって排他的にトリガーされ、Ankoの `doAsync` バックグラウンドスレッド内で `doPinning()` を起動します。Source-to-sink 解析により、ネットワークレスポンスは146行目の `android.util.Log.v("Response", body.string())` にのみ流れ、`Toast` や `TextView` などのUIシンク、`SharedPreferences`、ファイル、およびIPCシンクは一切存在しないことが確認されています。そのため、ピンの不一致により `client.newCall(request).execute()` が `SSLPeerUnverifiedException` をスローした場合、147～148行目で例外が捕捉され、`System.err` に出力された後、ユーザーに対するフィードバックなしに暗黙的に実行が継続されます。

エビデンスに関する制限事項： 検証フェーズ中、アクティブなインターセプティングプロキシまたはライブFridaインストールメンテナーによる動的悪用検証は実行されませんでした。本脆弱性は、例外処理階層およびコードベースのスコープ分離に対する決定論的な静的解析により確認されています。

推奨事項

緊急対応

`doPinning()` 内の包括的なcatchブロックを、セキュアに失敗する具体的な例外処理に置き換えてください：

```
} catch (SSLPeerUnverifiedException e) {
    Log.e("PINNING_FAILURE", "Certificate pinning failed for owasp.org", e);
    // Terminate connection and notify user
    return;
} catch (IOException e) {
    Log.e("NETWORK_ERROR", "Network error occurred", e);
}
```

恒久的な修正

- `SSLPeerUnverifiedException` を握りつぶさないでください。ピンニングの失敗はすべて、その接続に対して致命的なセキュリティエラーとして扱ってください。
- OkHttp `CertificatePinner` または `network_security_config.xml` 内の `<pin-set>` を使用し、`owasp.org` に限らずすべての本番エンドポイントにピンニングを拡張してください。

検証

- 1. プロキシCAを使用して、doPinning() 内の https://owasp.org リクエストを傍受してください。
- 2. 修正がない場合、アプリは例外をログに記録するだけで、データを表示しません。
- 3. 修正後は、アプリはピンニング失敗をログに記録し、接続を中断し、ネットワークリクエストが完了しない状態である必要があります。

中 不安全な外部ストレージ権限による認証情報漏洩（InsecureStorageSDCardActivity）

InsecureStorageSDCardActivity は、ユーザーの認証情報を外部ストレージ上の一時ファイルに書き込み、そのファイルを明示的に world-readable および world-writable に設定します。

本脆弱性により、ユーザーの認証情報が同デバイスにインストールされた他のアプリケーションに対して露出する。想定されるファイルパストラバーサルベクトルは本APKビルドには存在しないことが確認された一方、既存の安全でないストレージ実装は、共有ストレージ上に平文の認証情報を保持しているため、依然として重大な機密性リスクを構成している。

技術的詳細

検出事項 #1

根本原因

このActivityは Environment.getExternalStorageDirectory() に一時ファイルを作成し、続いて setReadable(true) および setWritable(true) を呼び出すことで、デバイス上のすべてのアプリケーションに対して読み取りおよび書き込みアクセス権を付与しています。また、絶対ファイルパスは System.out.println() を介してlogcatに漏洩しています。

脆弱なコード

InsecureStorageSDCardActivity.java（68〜76行目）：

```
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();
```

悪用証拠

[Static Analysis] ソースコードにより、Saveボタンの押下ごとに、すべてのアプリケーションから読み取り可能なファイルが作成され、その内部にユーザー名、パスワード、タイムスタンプが連結された状態で格納されることが確認されました。READ_EXTERNAL_STORAGE を保持するサードパーティアプリケーションは、これらのファイルを列挙して読み取ることが可能です。logcatへの漏洩（System.out.println）により絶対パスがさらに開示されるため、ファイルシステムの列挙は不要となります。

攻撃シナリオ： READ_EXTERNAL_STORAGE を保持する悪意あるアプリがlogcatを監視するか、外部ストレージのルートディレクトリを列挙して users*.tmp ファイルを検出し、平文の認証情報を抽出します。

検証証拠

根本原因

このアプリケーションは targetSdkVersion=26（Android 8.0）を対象としており、レガシーな外部ストレージ動作が強制されます。そのため Environment.getExternalStorageDirectory() は共有され、グローバルに参照可能な名前空間（例： /sdcard/ または /storage/emulated/0/ ）を返します。このディレクトリに作成されたファイルは、基盤となるsdcardfs/FUSE/エミュレートされたFAT32マウントがUnixパーミッションビットを反映しないため、READ_EXTERNAL_STORAGE を保持する他のインストール済みアプリケーションから本質的にアクセス可能です。検証により、android:debuggable="true" および android:allowBackup="true" の両方がマニフェストで有効化されていることが確認されました。これにより、ランタイムメモリ検査およびadbバックアップ抽出を通じて機密性リスクが増大しています。

脆弱なコード

逆コンパイルされたソース（base.apk/sources/owasp/sat/agoat/InsecureStorageSDCardActivity.java）の検証により、各不安全な操作の正確な行番号が特定され、完全な平文データの組み立てが追跡されました：

ディスクに書き込まれる認証情報の組み立て（45～51行目）：

```
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();
```

共有ストレージへのファイル作成（68～70行目）：

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

logcatへのパス開示（71行目）：

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

パーミッション呼び出し（72～73行目）：

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

[Static Analysis] `setReadable(boolean)` および `setWritable(boolean)` の呼び出しが、単一引数のオーナー専用オーバーロードであることが確認されました。API 26のレガシー外部ストレージでは、この区別は機能的に無意味です。エミュレートされたファイルシステムは `chmod` ビットを反映せず、共有外部ストレージのルートに配置されたファイルは實際上、`READ_EXTERNAL_STORAGE` を持つすべてのアプリから読み取り可能となるためです。

悪用証拠

[Static Analysis] マニフェストおよびSDKコンテキスト：

- `AndroidManifest.xml` は `WRITE_EXTERNAL_STORAGE` および `READ_EXTERNAL_STORAGE` を宣言しています。
- `targetSdkVersion="26"` および `minSdkVersion="18"` は `base.apk/apktool.yml` から抽出されました。
- `android:debuggable="true"` および `android:allowBackup="true"` の両方が存在します。

[Static Analysis] 緩和策の欠如：

- アプリケーションパッケージ内で `setReadable(false)`、`setWritable(false)`、`chmod`、`setExecutable`、および `setPermissions` を対象としたプロジェクト全体の検索は0件を返しました。パーミッションの縮小、クリーンアップ、またはサブディレクトリの分離に関するロジックは、コードベース内のどこにも存在しません。

[Static Analysis] 入力検証の欠如：

- `username.getText().toString()` および `password.getText().toString()` は、nullチェック、長さ制限、またはサニタイゼーションなしに、直接 `data` 文字列に連結されています。

[Static Analysis] 予測可能な検出：

- `File.createTempFile("users", "tmp", ...)` は外部ストレージのルートに `users*.tmp` を生成します。無条件の `System.out.println` により絶対パス（例：`userinfo /sdcard/users12345.tmp`）がlogcatに漏洩するため、ファイルシステムの列挙は不要となります。`READ_EXTERNAL_STORAGE` を保持する悪意ある他のインストール済みアプリは、正確なパスを直接対象として平文の認証情報文字列を抽出できます。

推奨事項

即時対応

`InsecureStorageSDCardActivity.java` から、明示的な `setReadable(true)` および `setWritable(true)` の呼び出しを削除してください。

恒久対策

機密性の高い認証データを外部ストレージに保存しないでください。ファイルの書き込み先をアプリのプライベートな内部ディレクトリに変更し、Androidのデフォルトの権限モデルを使用してください：

```
File internalDir = getFilesDir();
File userinfo = File.createTempFile("users", "tmp", internalDir);
// setReadable/setWritable calls removed - internal files are app-private by default
FileWriter fw = new FileWriter(userinfo);
```

```
fw.write(data);
fw.close();
```

また、絶対パスが漏洩する `System.out.println` のログ出力を削除してください。

検証

- ランタイム環境でファイル作成をインターセプトし、新しいファイルが `/data/data/owasp.sat.agoat/files/` 配下に作成されることを確認します。
- ファイルモードが他のUIDに対して読み書きアクセスを許可していないことを確認します (`ls -l` で `rw-----` と表示されること)。
- Save ボタンをタップした際に、logcatで絶対パスが出力されなくなったことを確認します。

中 FLAG_SECURE未設定により、機密画面で不正な画面キャプチャが許容される

owasp.sat.agoat の 25 個すべてのアクティビティに FLAG_SECURE が設定されておらず、PIN の入力や認証情報の入力、OTP の生成など、機密画面のスクリーンショットおよび画面録画が制限なく許可されています。

`WindowManager.LayoutParams.FLAG_SECURE` がすべての Activity ウィンドウに設定されていないため、`adb` およびサードパーティツールを用いて機密ワークフローの画面キャプチャおよび動画録画が無制限に許可されています。これは PIN 認証、クレデンシャル入力、OTP 生成、決済画面において動的に確認されました。攻撃者または侵害されたプロセスは、ユーザー操作なしにスクリーンショットから直接クレデンシャルや機密トークンを収集し、この情報収集結果を活用してより正確なオーバーレイ攻撃を構築する可能性があります。

技術的詳細

検出事項 #1

根本原因

`owasp.sat.agoat` のすべての Activity は、`WindowManager.LayoutParams.FLAG_SECURE` なしでウィンドウを初期化しています。このフラグは、システムのスクリーンショットユーティリティ、`adb`、およびサードパーティの画面録画アプリケーションによる画面コンテンツのキャプチャを防止するための Android の標準的なメカニズムです。

脆弱なコード

[静的解析] `onCreate()` 内で `getWindow().addFlags(WindowManager.LayoutParams.FLAG_SECURE)` または `getWindow().setFlags(...)` を呼び出す Activity は存在しません。25 個のすべての Activity にわたる逆コンパイルされた smali および Java ソースには、`FLAG_SECURE` への参照がゼロです。

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

悪用の証拠

エミュレータ上での動的テストにより、すべての機密性の高い画面で無制限の画面キャプチャおよび録画が可能であることが確認されました：

```
# Succeeded on PIN verification, credential input, OTP, and payment screens
adb shell screencap /sdcard/screen_pin_overlay.png
adb shell screenrecord /sdcard/record.mp4
```

`dumpsys window` の分析により、対象アプリケーションのウィンドウにいずれも `FLAG_SECURE` フラグが設定されていないことが確認されました。

記録された証拠には以下が含まれます：

- `/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`

これにより、悪意のあるアプリケーションまたは侵害されたデバイスが、ユーザーに知られることなく画面キャプチャから資格情報、PIN、OTP、およびクレジットカード番号を直接収集できるようになります。

検証証拠

検証試行 1

adb shell screencap が、PIN 認証および資格情報入力を含むすべての機密性の高い画面で成功しました

検証試行 2

adb shell screenrecord が、無制限にリアルタイムでオーバーレイ攻撃をキャプチャしました

検証試行 3

dumpsys window の分析により、対象アプリのウィンドウに FLAG_SECURE が設定されていないことが確認されました

検証試行 4

スクリーンショット

/workspace/screen_pin_overlay.png、/workspace/screen_hardcode_overlay.png、/workspace/screen_clipboard_overlay.png、/workspace/screen_overlay が、無制限の画面コンテンツをキャプチャしました

検証試行 5

根本原因

`owasp.sat.agoat` の `AccessControl1ViewActivity` は、マニフェスト宣言に `action.VIEW` とカスタムスキーム `androgoat://vulnapp` を含む `<intent-filter>` が存在するにもかかわらず `android:exported="false"` が省略されているため、暗黙的に exported になっています。さらに、この Activity のエントリポイントには認証ロジックが含まれておらず、アプリケーションは関連する `AccessControlIssue1Activity` で PIN 認証を強制しています。

脆弱なコード

`/workspace/agoat_decode/AndroidManifest.xml` のマニフェスト宣言：

```
<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>
```

`/workspace/agoat_analysis/sources/owasp_sat_agoat_AccessControl1ViewActivity.java` および対応する smali の静的解析により、以下が確認されています：

- `onCreate(Bundle)` はコンテンツビューを設定し、`startService()` を呼び出す `OnClickListener` をアタッチします。
- `onNewIntent(Intent)` はオーバーライドされていません。
- `onResume()` はオーバーライドされていません。
- `verifyIfPinSet()`、`isPinCorrect()`、`verifyPINView()`、または shared-preferences による PIN 検証への参照はありません。

悪用の証拠

デバイスには PIN が設定されており（`pinSet=true`、MD5 ハッシュ値 `1234`）：

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

ADB による動的悪用が、PIN 入力なしで成功しました：

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

出力：

```
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` により、この Activity がフォーカス状態にあることが確認されました：

```
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に直接起動の記録が残っています：

```
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
```

`/workspace/deeplink_accesscontrol1.png` にキャプチャされたスクリーンショットは、PIN 認証なしにディープリンクから直接起動された Activity の視覚的証拠を提供しています。

推奨事項

即時の軽減策

`FLAG_SECURE` をまず最もリスクの高いアクティビティ（PIN 認証、クレデンシャル入力、OTP 生成）に適用する：

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

恒久的な修正

すべての機密画面で使用するベースアクティビティクラスを作成し、ウィンドウ保護を一元化する：

```
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
        );
    }
}
```

すべての機密アクティビティに対して、ベースクラスを `AndroidManifest.xml` に登録する：

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

検証

- 各機密アクティビティをフォアグラウンドに表示した状態で `adb shell screencap /sdcard/test.png` を実行する。キャプチャされた画像がブラックアウトされていることを確認する。
- `adb shell screenrecord /sdcard/test.mp4` を実行し、保護対象のアクティビティの内容が録画から除外されていることを確認する。
- `dumpsys window` の出力を確認し、ウィンドウ属性に `FLAG_SECURE`（0x00002000）が含まれていることを検証する。

中 不安全的なバックアップおよびデバッグフラグによるアプリケーションサンドボックスの露出

アプリケーションマニフェストで `android:debuggable="true"` および `android:allowBackup="true"` が有効になっており、USB デバッグアクセス権を持つ任意のユーザーが、アプリのプライベートデータをすべて抽出・変更・復元したり、実行中のプロセスにデバッガをアタッチしたりできる状態です。

これらの manifest フラグは重要なサンドボックス保護を無効化します。`android:allowBackup="true"` は `adb backup` および `adb restore` によるデータ全体の抽出と改ざんを許可し、`android:debuggable="true"` は shell ユーザーが `run-as` 経由でアプリケーションの UID を取得し JDWP デバッガをアタッチすることを可能にします。両者が組み合わさることで、認証データの上書きや実行時メモリの検査を可能とする複数の容易な攻撃経路が生じます。

技術的詳細

検出事項 #1

根本原因

アプリケーションマニフェストに `android:debuggable="true"` および `android:allowBackup="true"` が宣言されています。これらのフラグにより、USB デバッグアクセスを持つ任意のユーザーは `adb backup` を介してアプリのプライベートデータをすべて抽出したり、`adb restore` を介して改ざん済みデータを復元したり、`run-as` を介してアプリケーションの UID としてコマンドを実行したりできます。これにより、SharedPreferences、SQLite データベース、その他のサンドボックスファイルへの制限なしの読み取り・書き込みアクセスが許可されます。

悪用エビデンス

[Static Analysis] `AndroidManifest.xml` に両方のフラグが含まれています：

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

`run-as` を使用することで、攻撃者はアプリのサンドボックスに直接書き込みできます：

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

`adb backup` を使用することで、攻撃者はアプリのデータ全体を抽出し、ファイルをオフラインで改ざんした上で、改ざん済みアーカイブを復元できます：

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

これらのベクトルにより、ローカルデータの完全性および機密性が完全に侵害されます。

検証エビデンス

根本原因

検証では根本原因をマニフェストフラグ自体にとどまらず、ストレージ層の完全性制御の欠如や高価値な平文データストアの存在まで拡大しました。静的解析により、`AndroidManifest.xml` に `android:fullBackupContent` が存在しないことが確認されました。これは、デフォルトのバックアップ範囲が `/data/data/owasp.sat.agoat/` ディレクトリ全体を対象としていることを意味します。さらに、すべての機密データ永続化ロジックが暗号化、HMAC、またはデジタル署名を欠いていることが判明し、アプリケーションは復元時に改ざん済みのサンドボックスファイルを暗黙のうちに受け入れてしまいます。

脆弱なコード

検証により、特定のアプリケーションコンポーネントが作成した5つの具体的なサンドボックス内ターゲットが特定されました：

1. `pinDetails.xml` — ソルトなし MD5 PIN 保存 `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();
}
```

`hashPIN()` メソッドは `MessageDigest.getInstance("MD5")` を使用しており (~113 行目相当)、暗号的に破綻したソルトなしダイジェストを生成しています。

2. `users.xml` — 平文クレデンシャル保存 `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. `aGoat` SQLite データベース — 平文クレデンシャルリポジトリ `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` — World-Readable な一時ファイルによるクレデンシャル漏洩 `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` — 完全性のみのゲーム状態 `InsecureStorageSharedPrefs1Activity.java:91-95` :

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

悪用エビデンス

検証ではマニフェストフラグを確認し、ソースコードを参照した悪用チェーンを構築しました：

静的マニフェスト確認：

```
<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:supportRtl="true"
    android:theme="@style/AppTheme">
```

`android:fullBackupContent` 属性は存在せず、`res/xml/` に対する正規表現検索によりバックアップ除外ルールは検出されませんでした。

`adb backup/restore` チェイン検証済み：

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

復元後の静的トレースにより、`AccessControlIssue1Activity.java:72-88` が完全性チェックなしに `pinDetails.xml` を読み取っていることが確認されました。

`isPinCorrect()` メソッドはユーザー入力のPINを保存されたMD5ダイジェストと比較します。`5f4dcc3b5aa765d61d8327deb882cf99` (passwordのMD5)を注入することで、passwordを使用した認証が可能になります。

`run-as` 読み取り・書き込みチェーン検証済み：

```
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'
```

攻撃者データを注入する書き込みコマンドも検証されました：

```
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'"
```

クレデンシャルレジストリ：検証では攻撃者制御下のPIN・ハッシュペアが初期化されました：

- `attacker_pin` → `5f4dcc3b5aa765d61d8327deb882cf99` (対象：`pinDetails.xml`)。

推奨事項

即時対応

`AndroidManifest.xml` から `android:debuggable="true"` を削除し、 `android:allowBackup="false"` を設定する：

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

恒久的な修正

リリースビルドに `debuggable="true"` が含まれないよう徹底する。リリースバリエーションでいずれかのフラグが有効になっている場合に失敗するよう、ビルド時のlintチェックまたはCI/CDゲートを実装する。ユーザーの利便性のためにバックアップが必要な場合は、従来のフルバックアップモデルの代わりに、Androidの暗号化バックアップメカニズム（クライアント側キーを使用したAutoBackup）を実装する。

検証

- 1. リリースAPKをビルドし、以下を実行する：

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

- いずれの属性も存在しないこと、または両方とも明示的に `false` に設定されていることを確認する。
- 2. `adb backup -f test.ab owasp.sat.agoat` を実行し、失敗するか空のアーカイブが生成されることを確認する。
 - 3. `adb shell run-as owasp.sat.agoat` を実行し、コマンドが拒否されることを確認する。

中 アプリケーションソースコード内にハードコードされた認証情報

複数のハードコードされたシークレット（ユーザ名、パスワード、および認証キーを含む）が、アプリケーションのソースコードに直接埋め込まれており、exported BroadcastReceiverを介して実行時に露出しています。

ハードコードされた認証情報は、認証や認可において秘密性を前提とするすべてのシステムのセキュリティを損ないます。 `ShowDataReceiver` がエクスポートされているため、悪意のあるサードパーティアプリケーションは実行時にユーザーの操作なしにこれらの秘密情報を漏洩させることができます。これらの認証情報がバックエンド環境間で再利用されている場合、不正アクセスへの直接的な経路を提供します。

技術的詳細

検出事項 #1

根本原因

`ShowDataReceiver.java` および `HardCodeActivity.java` には、認証クレデンシャルおよびシークレットを表すリテラル文字列が含まれており、これらはAPKにコンパイルされています。 `ShowDataReceiver` はエクスポートされており、デバイス上の任意のアプリケーションがToastを表示させて、これらの値を実行時に漏洩させることが可能です。

脆弱なコード

`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>
```

悪用の証跡

APKの静的逆コンパイルにより、動的な実行を必要とせずに正確な値が確認されました：

シークレット	値	ソース位置
ユーザー名	CrazyUser	ShowDataReceiver.java
パスワード	CrazyPassword	ShowDataReceiver.java
キー	123myKey456	ShowDataReceiver.java
プロモーションコード	NEW2019	HardCodeActivity.java
Firebase URL	https://androgoat-42597.firebaseio.com	res/values/strings.xml

これらの値は、APKの逆解析（JADX/apktool）によって容易に復元可能です。また、エクスポートされた ShowDataReceiver に対して broadcastIntent を送信することで、Toast内にクレデンシャルが表示され、値が漏洩します。

検証の証跡

根本原因

APKは、平文のクレデンシャルリテラルをDEXバイトコードに直接コンパイルしています。 ShowDataReceiver は AndroidManifest.xml 内で android:exported="true" として宣言されており、パーミッション要件がないため、同じデバイスにインストールされた任意のアプリケーションが到達可能な、保護されていないエントリポイントとなっています。

脆弱なコード

/workspace/my_source/sources/owasp/sat/agoat/ShowDataReceiver.java の逆コンパイルソース：

```
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 はプロモーションコードをコンパイル済みリテラルとして格納し、ローカルな比較に使用しています：

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

レシーバーがエクスポートされ保護されていないことを示すマニフェスト宣言：

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

悪用の証跡

ADBを介して broadcastIntent を送信することで、動的な悪用が確認されました：

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

ツールの出力：

```
Broadcasting: Intent { flg=0x400000 cmp=owasp.sat.agoat/.ShowDataReceiver }
Broadcast completed: result=0
```

ブロードキャストは正常に配信されました（ result=0 ）。これにより、ユーザー操作やパーミッションなしに、任意のサードパーティアプリケーションがクレデンシャルを漏洩させる Toast を表示させることが可能であることが証明されました。

コードベース全体を対象とした ripgrep 検索により、これらのシークレットが報告された箇所にのみ存在し、バックエンドのシンクがないことが確認されました：

シークレット	検出ファイル	Toast/UI 以外での使用
<code>CrazyUser</code>	<code>ShowDataReceiver.java:17</code>	なし
<code>CrazyPassword</code>	<code>ShowDataReceiver.java:17</code>	なし
<code>123myKey456</code>	<code>ShowDataReceiver.java:17</code>	なし
<code>NEW2019</code>	<code>HardCodeActivity.java:49</code>	なし（ローカルの <code>String.equals</code> のみ）
<code>androgoat-42597</code>	<code>res/values/strings.xml:130</code>	なし（コード上で読み込まれていない）

[外部プロープ] アプリケーションコードでは使用されていません（Firebase SDKのインポートおよび `R.string.firebase_url` への参照はありません）が、ハードコードされたFirebase URL `https://androgoat-42597.firebaseio.com` は未認証のHTTP リクエストに対して `200 OK` およびボディ `{"password":"androgoat","user":"androgoat"}` で応答し、当該エンドポイントが公開されていることを確認しました。

推奨事項

即時対応

- すべてのハードコードされたクレデンシャルを直ちにローテーションしてください。APKに埋め込まれているあらゆる値は公開状態にあるものとして扱ってください。
- 正当なアプリケーション機能に不要な場合は、exportedとして宣言された `ShowDataReceiver` を削除または無効化してください。

恒久対応

- デバイスアステーションを用いて認証を行う安全なリモート構成サーバーからシークレットを読み込むか、実行時にAndroid Keystore/Tink ライブラリから取得してください。
- パスワード、API キー、認証トークンをアプリケーションのソースコードやリソースに埋め込んでコンパイルしないでください。

検証

- 修正後、APKを再デコンパイルしてください。
- ソースツリー内で既知の平文文字列（ `CrazyUser` 、 `CrazyPassword` 、 `123myKey456` 、 `NEW2019` ）を検索してください。
- Java ソース、Kotlin ソース、 `strings.xml` 、 `assets/` 、 `res/raw/` に一致が一件もないことを確認してください。
- `AndroidManifest.xml` 内で `ShowDataReceiver` がexportedではなくなっていることを確認してください。

中 暗号化の脆弱性: 脆弱なハッシュアルゴリズム

クライアントは既知の暗号学的脆弱性を持つ脆弱なハッシュ関数を使用しています。

ハッシュ関数は、データをエンコードして-固定長-のハッシュ値を出力する関数です。衝突（2つの異なる入力から同じハッシュ値が生成されること）の発生確率を極限まで抑えることを目的としており、メッセージやファイルの完全性検証、ファイルやデータの識別、シークレットやパスワードの確認などに利用できます。

一般的なハッシュアルゴリズム:

- `MD5` : Message Digest Algorithm (MD5)は、1991年にMD4の代替として設計されました。2005年3月、同じMD5ハッシュを持つ2つの異なるファイルが公開され、その翌年には1分以内に衝突を発見できるアルゴリズムが発表されました。ハッシュアルゴリズムでは、2つの異なる入力から同じハッシュ値が得られることを計算量的に不可能にすることが求められますが、MD5はこの要件を致命的に欠いています。2008年、Software Engineering InstituteはMD5を「暗号学的に破綻しており、今後の使用に適さない」と結論付けました。
- `SHA1` : Secure Hash Algorithm 1(SHA1)はNSAによって設計されました。MD5同様、SHA1も衝突攻撃に脆弱です。2005年にはすでに安全でないと見なされ、2011年にNational Institute of Standards and Technology (NIST)によって正式に非推奨とされました。
- `SHA2` : NSAが設計したハッシュアルゴリズム群です。SHA-2はSHA1とは大きく異なる構造を持ち、高い安全性が認められています。
- `SHA-256` : SHA-2ファミリの一つです。接尾辞の256は、出力されるハッシュ値の長さが256ビットであることを示しています。SHA-256は安全性と信頼性が高い業界標準であり、衝突の可能性は事実上不可能（2²⁵⁶）です。
- `SHA3` : SHA3はSecure Hash Algorithmファミリの最新規格であり、2015年にNISTによって公開されました。同じ規格シリーズの一つですが、内部構造はSHA1やSHA2のMD5的な構造とは大きく異なり、高い安全性が認められています。
- `Argon2` : 2015年のPassword Hashing Competitionで優勝したアルゴリズムです。現在最も安全なパスワードハッシュアルゴリズムの一つとして認められており、Internet Engineering Task Force (IETF)、Password Hashing Competition (PHC)、The Open Worldwide Application Security Project (OWASP)など、多くのセキュリティおよび暗号規格に採用されています。
- `Bcrypt` : 1999年に登場したBcryptは、ブルートフォース攻撃に対して高い耐性を持つ低速かつ適応的なアルゴリズムです。ユーザーが計算コストを制御できるため、ハードウェアの進化に合わせて調整可能です。Bcryptは、Open Web Application Security Project (OWASP) および Payment Card Industry Data Security Standard (PCI DSS) を含む多くのセキュリティ規格にも採用されています。
- `HMAC` : Hash-based message authentication codeはハッシュアルゴリズムではなく、ハッシュアルゴリズムと秘密鍵を組み合わせる暗号認証技術です。HMACの強度は基盤となるハッシュアルゴリズムに依存するため、非推奨または脆弱なハッシュアルゴリズムと共に使用するべきではありません。

MD5 および **SHA-1** は安全ではなく、HMACと併用した場合であっても使用してはなりません。

技術の詳細

検出事項 #1

アプリケーションはMD5を使用しています。MD2、MD4、およびMD5は衝突攻撃により破られているハッシュ関数です。コールトレース：

```
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
```

検出事項 #2

このアプリケーションはMD5を使用しています。MD2、MD4、MD5は衝突攻撃により安全性が損なわれたハッシュ関数です。コールトレース：

```
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...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));
}
```

検出事項 #3

このアプリケーションはMD5を使用しています。MD2、MD4、およびMD5は、衝突攻撃により暗号的に破綻したハッシュ関数です。コールトレース：

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

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);
}
```

検出事項 #4

このアプリケーションはMD5を使用しています。MD2、MD4、およびMD5は衝突攻撃により破られているハッシュ関数です。コールトレース：

```
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.checkExpressionValueIsNotNull(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;
    }
}

```

検出事項 #5

アプリケーションはMD5を使用しています。MD2、MD4、MD5は衝突攻撃により暗号的に破られたハッシュ関数です。

コールトレース：

```

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.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.checkExpressionValueIsNotNull(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.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;  
}
```

検出事項 #6

アプリケーションはMD5を使用しています。MD2、MD4、MD5は衝突攻撃により破られたハッシュ関数です。コールトレース：

```
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 `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;
    }
}
```

検出事項 #7

アプリケーションはMD5を使用しています。MD2、MD4、MD5は衝突攻撃により破綻したハッシュ関数です。コールトレース：

```
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 `java.security.MessageDigest.getInstance`

ソースコードを抽出できませんでした。

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;
}
```

推奨事項

弱いハッシュアルゴリズムの使用リスクを軽減するため、以下を考慮してください。

- MD5 および SHA-1 は安全性が低く、衝突が発生しやすいため、使用を避けてください。HMAC を含め、いかなる目的にも使用すべきではありません。
- ハッシュ化や署名検証には、SHA-256 または SHA-3 を採用してください。高い安全性を提供し、業界で広く受け入れられている標準規格です。
- 機密データ (PII、PHI など) のハッシュ化には、Argon2 または Bcrypt を検討してください。ブルートフォース攻撃に耐性があり、セキュリティ標準で推奨されています。

- メッセージ認証にHMACを使用する場合は、SHA-256やSHA-3などの安全なハッシュアルゴリズムと組み合わせ、堅牢なセキュリティを確保してください。

参考資料

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

中 平文 HTTP リクエスト

CVSS v3: 9.1

アプリケーションが暗号化されていないチャンネルでデータにアクセスしており、通信中のデータの機密性と完全性を損なっています。

平文の HTTP 通信とは、モバイルアプリとバックエンドサーバー間でやり取りされるデータが暗号化されていない状態を指します。これにより、データは容易に読み取られたり、悪意のある攻撃者に傍受されたりするリスクがあります。このような暗号化の欠如は、モバイルアプリに重大なセキュリティリスクをもたらします。ログイン認証情報や個人情報、金融情報などの機密データが平文の HTTP 接続を介して送信されると、盗聴や中間者攻撃を受ける脆弱性が生じます。

モバイルアプリにおける平文の HTTP 通信に伴う主なリスクの一つは、ユーザーデータが外部に露出する可能性です。攻撃者は通信を傍受して重要な情報を抽出し、なりすまし、アカウントへの不正アクセス、個人情報の悪用につなげることがあります。また、攻撃者はこれらの脆弱性を悪用してアプリのコンテンツを改ざんしたり、通信チャンネルに悪意のあるコードを注入したりするリスクもあります。

さらに、平文の HTTP 通信はユーザーのプライバシーを損なう可能性があります。モバイルアプリは、アナリティクスやターゲティング広告の目的でユーザーデータを収集・送信することが少なくありません。暗号化がない状態では、これらのデータが第三者に容易にアクセスされ、ユーザーの機密性が損なわれるだけでなく、プライバシー侵害に発展する可能性があります。

安全な通信プロトコルを実装していないモバイルアプリは、なりすまし攻撃のリスクも抱えています。攻撃者は偽の Wi-Fi ネットワークを構築したり、他の手法を用いたりして通信を傍受し、正当なバックエンドサーバーに扮することができます。ユーザーは気づかないうちにデータを攻撃者に送信してしまい、それが悪意のある目的に悪用される可能性があります。

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."
        }
    }
}
}
}

```

技術的詳細

検出事項 #1

暗号化されていない平文のHTTP URL : `http://demo.testfire.net/` リクエスト :

```

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

```

レスポンス :

```

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

```

推奨事項

モバイルアプリにおける平文HTTP通信のリスクを軽減するには、安全な通信プロトコルとベストプラクティスの導入が不可欠です。モバイルアプリの通信セキュリティを強化するための詳細な推奨事項を以下に示します。

- **HTTPS 暗号化の導入:** モバイルアプリとバックエンドサーバー間のすべての通信にHTTPS (HTTP Secure) を使用することは、最も基本的な対策です。HTTPSは送信時にデータを暗号化し、機密情報の機密性と安全性を確保します。信頼できる認証局 (CA) からSSL/TLS証明書を取得し、すべてのAPIエンドポイントとデータ転送にHTTPS接続を強制してください。
- **証明書ピンニング:** HTTPS接続のセキュリティを強化するため、証明書ピンニングを実装してください。これは、サーバーのSSL証明書またはその公開鍵をモバイルアプリ内にハードコードすることを指します。これにより、アプリは意図したサーバーとのみ通信し、攻撃者が独自の証明書を使用しようとする中間者攻撃を防ぐことができます。
- **HSTS (HTTP Strict Transport Security) の有効化:** HSTSヘッダーを使用し、ユーザーが誤ってURLに「http://」と入力した場合でも、アプリのWebブラウザが常にHTTPS接続を使用するように指示してください。これにより、潜在的なダウングレード攻撃を防ぎ、すべての通信が暗号化されることを保証します。
- **最新のTLSバージョンの使用:** モバイルアプリが最も堅牢なセキュリティ機能と暗号化アルゴリズムを利用できるよう、TLS (Transport Layer Security) プロトコルの最新バージョンを使用してください。
- **証明書チェックの実装:** 失効または期限切れのSSL証明書をチェックするようアプリを設定し、侵害された可能性のあるサーバーとのやり取りを回避してください。
- **機密データの暗号化:** デバイスに保存される、またはサーバーへ送信される機密データを暗号化してください。強力な暗号化アルゴリズムを利用し、暗号化キーを安全に管理することで、データ転送のセキュリティをさらに強化できます。
- **すべてのAPIにおける安全なデータ転送:** アプリが使用するすべてのAPIを評価し、それらのデータ転送にもHTTPSが使用されていることを確認してください。これには、アプリに統合されたサードパーティ製APIやサービスも含まれます。

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();

    // ログイン用の安全な HTTPS リクエストをシミュレートします (実際の安全な API エンドポイントに置き換えてください)。
    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),
    ],
  ),
),
);
}
}

```

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
        }

        // ログイン用の安全な HTTPS リクエストをシミュレートします（実際の安全な API エンドポイントに置き換えてください）。
        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()

            // ログイン用の安全な HTTPS リクエストをシミュレートします (実際の安全な API エンドポイントに置き換えてください)。
            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."
            }
        }
    }
}

```

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

攻撃ベクトル:

品 ネットワーク (N)

隣接 (A)

ローカル (L)

物理 (P)

攻撃の複雑さ:

低 (L)

高 (H)

必要な権限:

None (N)

低 (L)

高 (H)

ユーザー関与:

None (N)

要 (R)

スコープ:

変更あり (C)

変更なし (U)

機密性:

高 (H)

低 (L)

None (N)

完全性:

高 (H)

低 (L)

None (N)

可用性:

高 (H)

低 (L)

None (N)

参考資料

- 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)

中

CERTIFICATE_EXPIRED

期限切れのSSL/TLS証明書はブラウザやクライアントから信頼されなくなり、セキュリティ警告やサービス中断、潜在的なセキュリティ脆弱性（中間者攻撃など）を招きます。

期限切れのSSL/TLS証明書は最新のブラウザやクライアントで信頼されなくなり、信頼チェーンが破断し、セキュリティ警告が表示され、さらに man-in-the-middle（MitM）攻撃などの複数の潜在的なセキュリティ脆弱性にさらされます。

証明書が期限切れになると、ブラウザは「接続はプライベートではありません」や「証明書の有効期限が切れています」といった顕著なセキュリティ警告を表示します。これらの警告はクライアントとサーバー間の安全な通信を妨害し、ユーザーの信頼とビジネスの継続性に直接影響を与えます。eコマースやその他の機微な環境では、これは多くの場合ユーザーの離脱、金銭的損失、および信用失墜を招きます。

期限切れ証明書のセキュリティリスクは、ユーザーへの影響を超えて広がります。攻撃者は有効な証明書がない状況を悪用して MitM 攻撃を実行し、送信途中の機微なデータを傍受または改ざんできます。サーバーを認証する有効な証明書がない場合、暗号化通信は保護機能を失い、データが傍受、改ざん、または偽造のリスクにさらされたままになります。

たとえば、期限切れの証明書を持つ Web サイトは、ユーザーデータの機密性と完全性をもはや保証できません。攻撃者は接続をスプーフィングして Web サイトを偽装し、ログイン認証情報や支払い情報を傍受する可能性があり、不正アクセスや金銭詐欺につながります。

期限切れの証明書は、PCI DSS、HIPAA、GDPR などの複数のセキュリティ基準への不適合も意味します。これらの規制では、機微なデータを保護するために有効な SSL/TLS 証明書が求められます。不適合は、法的ペナルティやさらなる信用失墜につながる可能性があります。

技術的詳細

検出事項 #1

demo.testfire.net で CERTIFICATE_EXPIRED が検出されました

- 検出内容：有効期限まで 60 日未満（33 日）

推奨事項

有効期限切れのSSL/TLS証明書に伴うリスクに対処するため、組織は以下の予防的な対策を講じるべきです。

- 自動監視ツール：SSL/TLS 証明書の状態を継続的に監視し、有効期限が近づいた際にアラートを送信するツールを導入します。これにより、期限切れに気づかず発生するサービス中断を防ぐことができます。
- 緊急更新手順：想定外の期限切れに備え、迅速に証明書を更新できる明確な緊急手順を策定します。これにより、ダウンタイムを最小限に抑え、潜在的なセキュリティリスクを防ぎます。
- 定期監査：SSL/TLS 証明書のインベントリを定期的に監査し、期限切れの証明書を迅速に特定して更新します。常に最新のインベントリを維持することで、セキュリティ上の抜け漏れを防ぎます。
- 自動証明書プロビジョニング：SSL/TLS 証明書の発行と更新を自動化し、継続的な有効性を維持するシステムを導入します。Kubernetes の cert-manager などのツールにより、このプロセスを効率化できます。

Kubernetesにおける自動証明書プロビジョニング：

Kubernetesではcert-managerを使用してSSL/TLS証明書の管理を自動化できます。このツールはLet's Encryptなどの認証局（CA）と連携し、証明書の自動発行と更新を行います。

- cert-managerのインストール：

```
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/latest/download/cert-manager.yaml
```

以下はcert-managerで証明書の自動プロビジョニングを行うためのYAML設定例です。

```
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
```

Certbotを使用した監視：

Kubernetesを使用しない環境では、Certbotを利用してSSL証明書の発行と更新を自動化できます。

```
# 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
```

Cron Jobを使用した自動証明書更新：

cronjobを利用して更新プロセスを自動化できます。

```
# Edit your crontab with: crontab -e
0 0 * * * /usr/bin/certbot renew --quiet
```

このcronjobは毎日深夜0時に実行され、更新が必要な証明書を確認することで、手動介入なしに継続的な保護を確保します。

参考資料

- [NIST Guidelines for Key Management](#)
- [SSL Labs Best Practices](#)

中 安全でないHTTPヘッダー設定

アプリケーションのヘッダー設定が未設定であるか、安全でないヘッダー値が設定されています。

安全でないヘッダー設定:

- **Content Security Policy:** コンテンツの信頼できるソースを指定することで、クロスサイトスクリプティング（XSS）のリスクを軽減します。
- **Cookie:** HttpOnlyやSecureなどの属性を設定し、cookieデータへの不正アクセスから保護することで、セキュリティを強化します。
- **Cross-Origin Resource Sharing:** 異なるドメイン間でのリソース共有方法を制御し、悪意あるアクセスを防止します。
- **HTTP Public Key Pinning:** 特定のサイトで有効な公開鍵を指定することで、中間者攻撃から保護します。
- **Redirection:** リダイレクトが安全であり、信頼できる宛先のみに誘導されるようにすることで、オープンリダイレクト脆弱性を防止します。
- **Referrer Policy:** サイト間を移動する際に送信されるリファラー情報の量を定義し、プライバシーを強化します。
- **Subresource Integrity:** サードパーティドメインから読み込まれたリソースが改ざんされていないか、暗号化ハッシュを検証することで確認します。

- **X-Content-Type-Options:** ブラウザによるコンテンツタイプの MIME スニーフィングを防止し、コンテンツタイプベースの攻撃リスクを低減します。
- **X-Frame-Options:** ページのフレームへの埋め込みを制御することで、クリックジャッキング攻撃を防止します。
- **X-XSS-Protection:** ブラウザに組み込まれた XSS フィルタを有効化し、検出されたクロスサイトスクリプティング攻撃をブロックします。
- **Permissions-Policy:** ブラウザで使用可能な機能や API を制御し、信頼できないコンテンツの機能を制限することで、セキュリティを強化します。
- **Clear-Site-Data:** サイトがブラウザに対して、指定されたオリジンに保存されたデータ（cookie、ローカルストレージ、キャッシュなど）の削除を要求でき、データ漏洩やプライバシー上の懸念の影響を軽減します。

技術的詳細

検出事項 #1

demo.testfire.net で MISSING_SECURITY_HEADERS を検出

- 検出内容：--

推奨事項

安全でないヘッダー設定を防ぐため、以下を検討してください。

1. Content Security Policy (CSP):

コンテンツソースへの制限を適用し、クロスサイトスクリプティング（XSS）攻撃や不正なリソース読み込みに伴うリスクを軽減します。

```
Content-Security-Policy: default-src 'self'; script-src 'self' https://cdnjs.cloudflare.com;
```

2. Cookie Security Headers:

Secure および HttpOnly フラグを実装し、Cookie の窃取や改ざんを防止して、ユーザーセッションのセキュリティを強化します。

```
Set-Cookie: sessionId=abc123; Secure; HttpOnly;
```

3. Cross-Origin Resource Sharing (CORS):

CORS ポリシーを適切に構成し、異なるオリジンからのリソースアクセスを制限することで、クロスサイトリクエストフォージェリ（CSRF）やクロスオリジンにおけるデータ漏洩リスクを軽減します。

```
Access-Control-Allow-Origin: https://example.com
```

4. HTTP Public Key Pinning (HPKP):

HPKP を利用して公開鍵を特定の Web サーバーにバインドし、不正な証明書を使用した Man-in-the-Middle（MitM）攻撃から保護します。

```
Public-Key-Pins: pin-sha256="base64==" ; max-age=5184000; includeSubDomains;
```

5. Redirection Headers:

厳格な制御を実装してオープンリダイレクションの脆弱性を防止し、安全なリダイレクションを確保することで、フィッシング攻撃からユーザーを保護します。

```
Location: https://example.com/secure-page
```

6. Referrer Policy:

適切なリファラーポリシーを設定し、Referer ヘッダーに含まれる情報量を制御することで、機密データの露出リスクを低減します。

```
Referrer-Policy: strict-origin-when-cross-origin
```

7. Subresource Integrity (SRI):

SRI を実装してスクリプトやスタイルシートなどの外部リソースの整合性を検証し、不正な改変やサプライチェーン攻撃を防御します。

```
<script src="https://example.com/example.js" integrity="sha256-base64==" crossorigin="anonymous"></script>
```

8. X-Content-Type-Options:

'nosniff' ディレクティブを有効化し、ブラウザによるレスポンスの MIME スニーフィングを防止することで、コンテンツタイプ混同攻撃に伴うリスクを軽減します。

```
X-Content-Type-Options: nosniff
```

9. X-Frame-Options:

X-Frame-Optionsを設定してWebコンテンツのフレームへの埋め込みを制限し、クリックジャッキング攻撃を防御するとともに、Webページの整合性を確保します。

```
X-Frame-Options: DENY
```

10. X-XSS-Protection:

XSS保護メカニズムを有効化し、ブラウザに対して潜在的な悪意のあるスクリプトを無害化またはブロックするよう指示することで、XSS攻撃を軽減します。

```
X-XSS-Protection: 1; mode=block
```

参考資料

- [OWASP HTTP Header Security](#)
- [Content Security Policy \(MDN\)](#)
- [X-Frame-Options \(MDN\)](#)

中 安全でないHTTPヘッダー設定: HTTP Strict Transport Security (HSTS)

安全でないHTTPヘッダー設定: HTTP Strict Transport Security (HSTS).

HTTP Strict Transport Security (HSTS) は、Webサーバーが準拠するユーザーエージェント (Webブラウザなど) に対し、安全な (HTTPS) 接続のみを使用して通信することを宣言するWebセキュリティポリシーの仕組みです。サーバーは、「Strict-Transport-Security」というHTTPレスポンスヘッダーフィールドを介して、ユーザーエージェントにHSTSポリシーを伝達します。HSTSポリシーは、ユーザーエージェントがサーバーに安全な方法でのみアクセスする期間を指定します。

WebアプリケーションがユーザーエージェントにHSTSポリシーを発行すると、準拠するユーザーエージェントは以下のように動作します：

- Webアプリケーションを参照する安全でない (HTTP) リンクを、自動的に安全な (HTTPS) リンクに変換します。(例えば、<http://example.com/some/page/> はサーバーにアクセスする前に <https://example.com/some/page/> に変更されます。)
- 接続のセキュリティが確保できない場合 (例：サーバーのTLS証明書が自己署名の場合)、ユーザーエージェントはエラーメッセージを表示し、ユーザーがWebアプリケーションにアクセスすることを許可しません。

技術的詳細

検出事項 #1

`demo.testfire.net` で HSTS が検出されました。

- 所見：未対応

推奨事項

ウェブサーバーを設定し、HTTP リクエストを HTTPS にリダイレクトするよう構成してください。

Apache の場合は httpd.conf を変更してください。その他の設定については、外部参照セクションを参照してください。

```
# 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>
```

- [Strict-Transport-Security](#)

中 不安全なTLS暗号スイートのサポート

クライアントは、既知の暗号上の弱点を持つ暗号スイートの組み合わせをサポートしています。

Transport Layer Security (TLS) は、クライアントがセキュア接続を要求し、サポートする暗号スイートの一覧を提示することから始まります。

暗号スイートは、認証、暗号化、メッセージ認証コード (MAC)、および鍵交換のアルゴリズムの組み合わせです。各アルゴリズムはプロトコルネゴシエーション時に特定の役割を担います。例えば、脆弱なアルゴリズムを使用すると、チャンネル全体のセキュリティが深刻に損なわれる可能性があります。

エンドポイントが、既知の暗号学的脆弱性を抱える暗号スイートと Secure Sockets Layer / Transport Layer Security (SSL/TLS) プロトコルの組み合わせをサポートしていることが判明しました。そのため、これらを効果的なトランスポート層のセキュリティとして依存すべきではありません。接続を傍受可能な攻撃者は、通過するトラフィックに影響を与えたり、復号したりする可能性があります。

以下のすべての設定において、暗号学的脆弱性が実証されています。

- 非推奨の SSL プロトコル (v2 または v3) のサポート
- 128ビット未満の鍵長を提供する暗号スイートのサポート
- TLS プロトコルバージョン 1.1 未満との組み合わせで BEAST 攻撃に脆弱な CBC モードを提供する暗号スイートのサポート
- CRIME 攻撃に脆弱な TLS 内の圧縮の使用
- RC4 を暗号として提供する暗号スイートのサポート
- DES を暗号として提供する暗号スイートのサポート
- MD5 を署名アルゴリズムとして提供する暗号スイートのサポート

技術的詳細

検出事項 #1

cipherlist_OBSOLETE が demo.testfire.net で検出されました。

- 所見: 旧式の暗号スイートが提供されています。

検出事項 #2

cipherlist_OBSOLETE が owasp.org で検出されました

- 検出内容: 廃止された暗号スイートが提供されています

推奨事項

推奨される TLS 設定では、以下の推奨事項を適用してください。

- TLSv1 以上に制限し、 TLS1.2 を優先する
- SSLv3 が必要な場合は、プロトコルダウングレード攻撃を防ぐため、 TLS Fallback SCSV 機能の実装を推奨する
- Anonymous Diffie-Hellman (ADH) を無効化する
- aNULL および eNULL 暗号スイートを無効化する
- Export キー交換スイートを無効化する
- RC4 サポートを削除する
- DES サポートを削除する
- MD5 サポートを削除する
- SHA1 より SHA256 を優先する
- AES256 より AES128 を優先する。256bit 版はセキュリティ上の利点が限定的であり、タイミング攻撃に対する堅牢性が低い
- Client-Initiated Renegotiation を無効化する
- TLS 圧縮を無効化する
- 鍵長が 128bit を超える暗号のみを有効にする
- Perfect-Forward Secrecy プロパティを持つ暗号スイート (DHE 、 ECDHE) のみを有効にする
- Logjam 攻撃への対策として、カスタムの Diffie-Hellman グループを使用する
- HTTP Strict Transport Security ヘッダーフィールドを実装する
- OSCP stapling を実装する

以下は、 [Mozilla SSL Configuration Generator](#) を使用して生成した、Nginx 向けのセキュアな TLS 設定のサンプルです。

```

server {
    listen 443 ssl;

    # SERVER HELLO でクライアントに送信される証明書は、ssl_certificate に連結して指定する
    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;

    # DHE 暗号スイート用の Diffie-Hellman パラメータ。2048bit を推奨
    ssl_dhparam /path/to/dhparam.pem;

    # モダン構成。必要に応じて調整してください。
    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-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 が必要) (15768000秒 = 6カ月)
    add_header Strict-Transport-Security max-age=15768000;

    # OCSP Stapling ---
    # ssl_certificate 内の URL から OCSP レコードを取得し、キャッシュする
    ssl_stapling on;
    ssl_stapling_verify on;

    ## ルートCAおよび中間証明書を使用して、OCSP レスポンスの信頼チェーンを検証する
    ssl_trusted_certificate /path/to/root_CA_cert_plus_intermediates;

    resolver <IP DNS resolver>;

    ....
}

```

参考資料

- [Mozilla Secure TLS/SSL Configuration](#)
- [Transport Layer Protection Cheat Sheet \(OWASP\)](#)
- [SSL/TLS Deployment Best Practices \(SSLabs\)](#)

中 TLS_FALLBACK_SCSV 非対応

サーバーはTLS_FALLBACK_SCSVをサポートしておらず、プロトコルダウングレード攻撃に対して脆弱です。

この脆弱性は、サーバーが **TLS_FALLBACK_SCSV** (Signaling Cipher Suite Value) をサポートしていないことを示しています。これはSSL/TLS接続におけるプロトコルダウングレード攻撃を防ぐために設計された仕組みです。

TLS_FALLBACK_SCSV は以下の理由から極めて重要です。

1. クライアントが接続失敗により下位プロトコルバージョンにフォールバックする際に、その旨をサーバーに通知できる。
2. サーバーが悪意のあるダウングレード試行を検知して防止できる。
3. SSL/TLS 接続における最高レベルのセキュリティを維持するのに役立つ。

TLS_FALLBACK_SCSV がサポートされていない場合、サーバーは以下の脅威にさらされます。

1. プロトコルダウングレード攻撃：攻撃者が古く安全性の低いプロトコルバージョンの使用を強制できる。
2. 中間者 (MitM) 攻撃：脆弱なプロトコルにダウングレードすることで、攻撃者が通信を傍受および復号しやすくなる。
3. 旧プロトコルの既知の脆弱性の悪用：ダウングレードが成功すると、古いSSL/TLSバージョンに存在する既知の脆弱性が悪用される可能性がある。

攻撃シナリオ：

攻撃者がクライアントとサーバー間の初期ハンドシェイクを傍受します。攻撃者は接続を操作して失敗させ、クライアントにより低いプロトコルバージョンでの再試行を促します。**TLS_FALLBACK_SCSV** がない場合、サーバーはこのダウングレードを受け入れてしまい、古いプロトコルバージョンに存在する脆弱性によって接続がさらされる可能性があります。

TLS_FALLBACK_SCSV のサポートがないことは、SSL/TLSセキュリティのベストプラクティスに違反し、各種セキュリティ標準および規制へのコンプライアンスに影響を与える可能性があります。

技術的詳細

検出事項 #1

demo.testfire.net で **fallback_SCSV** を検出

- 診断結果：未対応

推奨事項

TLS_FALLBACK_SCSV（Signaling Cipher Suite Value）がサポートされていない場合、ダウングレード攻撃のリスクが生じる可能性があります。このリスクを軽減するための対策は以下のとおりです。

予防的な対策

- SSL/TLSバージョンのフォールバックを無効化**：サーバーを古いSSL/TLSバージョンへのフォールバックが行われないように設定します。これにより、攻撃者が接続を脆弱なプロトコルに強制的にダウングレードすることを防ぎます。
- 最小TLSバージョンの強制**：クライアントおよびサーバー両方で許容する最小TLSバージョン（例：TLS1.2）を設定し、脆弱なバージョンへのダウングレードを防止します。
- 定期的なセキュリティスキャン**：インフラストラクチャを定期的にスキャンし、TLS設定の脆弱性を特定して対処します。
- TLSネゴシエーションの異常パターン監視**：プロトコルの強制ダウングレードを試みる挙動を検出するための監視体制を構築します。

実装例

1. Nginx設定

NginxでSSL/TLSバージョンのフォールバックを無効化し、最小TLSバージョンを強制するには：

```
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. Apache設定

Apacheでも同様の結果を得るには、以下の設定を使用します：

```
<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. テスト用OpenSSLコマンド

OpenSSLを使用してサーバーのTLS設定をテストするには：

```
openssl s_client -connect example.com:443 -tls1_2
```

このコマンドはTLS1.2接続の確立を試行します。成功した場合、サーバーが最新のTLSバージョンを使用するよう正しく設定されていることを示します。

ダウングレード試行の監視

- fail2banの使用

fail2ban を使用してログを監視し、潜在的なダウングレード試行を検出できます：

1. `/etc/fail2ban/filter.d/tls-downgrade.conf` にカスタムフィルタを作成します：

```
[Definition]
```

```
failregex = SSL routines:SSL23_GET_CLIENT_HELLO:unsupported protocol.*client=<HOST>
```

2. `/etc/fail2ban/jail.local` にjailを追加します：

```
[tls-downgrade]
enabled = true
filter = tls-downgrade
logpath = /var/log/nginx/error.log
maxretry = 3
bantime = 3600
```

3. fail2banを再起動します：

```
sudo systemctl restart fail2ban
```

この設定により、サポートされていない（おそらく古い）SSL/TLSバージョンを使用した接続を繰り返し試行するIPアドレスが遮断されます。

これらの対策を実施することで、TLS_FALLBACK_SCSVがサポートされていない場合でも、TLSダウングレード攻撃のリスクを大幅に低減できます。

参考資料

- [NIST Guidelines for TLS Implementations](#)
- [RFC 7507 - TLS Fallback Signaling Cipher Suite Value \(SCSV\) for Preventing Protocol Downgrade Attacks](#)
- [OWASP TLS Cheat Sheet](#)

中 SSL/TLS証明書の暗号鍵および署名アルゴリズムの脆弱性

SSL/TLS証明書は暗号鍵と署名アルゴリズムの両方が脆弱であり、暗号化接続の安全性を損ない、証明書の偽造やその他の暗号攻撃を許容する可能性があります。

暗号学的に脆弱な鍵や署名アルゴリズムを使用したSSL/TLS証明書は、重大なセキュリティリスクをもたらし、システムを暗号攻撃に対して脆弱な状態にします。

暗号鍵および署名アルゴリズムの脆弱性に関する主なポイント：

1. 脆弱な暗号鍵：

- RSA鍵：2048ビット未満のRSA鍵は脆弱と見なされます。現在の推奨は最低2048ビットを使用し、長期的なセキュリティには3072ビット以上を使用することです。
- DSA鍵：RSAと同様に、2048ビット未満のDSA鍵は脆弱と見なされ、使用を避けるべきです。
- 計算能力の向上：計算能力の向上に伴い、短い鍵長は攻撃に対してますます脆弱になっています。
- 業界標準：多くの業界標準およびコンプライアンス要件では、SSL/TLS証明書の最小鍵長が義務付けられています。

2. 脆弱な署名アルゴリズム：

- 廃止されたアルゴリズム：MD5およびSHA1は暗号学的に破綻していると見なされ、使用すべきではありません。
- 衝突攻撃：脆弱なアルゴリズムは衝突攻撃に対して脆弱であり、攻撃者が正当な証明書と同じ署名を持つ偽造証明書を作成できる可能性があります。
- 業界標準：多くの業界標準およびコンプライアンス要件では、脆弱な署名アルゴリズムの使用が禁止されています。
- 後方互換性：一部のシステムでは後方互換性のために脆弱なアルゴリズムを使用している場合がありますが、この慣行は強く推奨されません。

実世界での影響：1024ビットのRSA鍵を使用している、あるいはSHA1で署名された証明書を持つWebサイトは、膨大な計算リソースを持つ攻撃者の標的となる可能性があります。攻撃者は傍受した通信を復号したり、偽造証明書を作成したりして、中間者攻撃を可能にし、通信のセキュリティを侵害することができます。そのため、強固な鍵とアルゴリズムを要求する業界標準への準拠は、SSL/TLSセキュリティを維持するために不可欠です。

技術的詳細

検出事項 #1

WEAK_SIGNATURE_ALGORITHMSが `demo.testfire.net` で検出されました

- **検出内容：**TLS1.2において脆弱な署名アルゴリズム（RSA+SHA1）が検出されました。

推奨事項

弱いキーによる脆弱性への対策：

1. 十分な鍵長を使用する：

- RSAおよびDSAでは、最低でも2048ビットの鍵長を使用する。
- 長期的なセキュリティを確保する場合は、3072ビット以上の使用を検討する。

2. 楕円曲線暗号（ECC）の検討：

- ECCは短い鍵長で同等のセキュリティを実現し、高いパフォーマンスを発揮する。
- 高いセキュリティを確保するため、P-256やP-384といった曲線を使用する。

3. 鍵の定期ローテーション：

- 定期的に鍵をローテーションするポリシーを策定し、特に鍵長を強化する際は実施する。

4. 既存証明書の監査：

- SSL/TLS証明書を定期的に監査し、弱いキーを使用しているものを特定して差し替える。

5. 最新の認証局（CA）の使用：

- 強固な鍵要件を適用し、業界のベストプラクティスに準拠したCAを選択する。

6. 堅牢な鍵生成手法の導入：

- 鍵生成には暗号的に安全な乱数生成器を使用する。
- 鍵の生成および保存には、ハードウェアセキュリティモジュール（HSM）の使用を検討する。

参考資料

- [NIST Recommendation for Key Management](#)
- [Baseline Requirements for TLS Server Certificates](#)
- [SSL Labs Best Practices](#)

中 モバイルネットワーク接続の悪用

アプリケーションはモバイルネットワーク接続の使用を強制しています。

ダウンロードマネージャーは、長時間実行されるHTTPダウンロードを処理するシステムサービスです。アプリケーションは、モバイルネットワーク接続のみの使用を強制すべきではありません。

技術的詳細

検出事項 #1

モバイルネットワーク接続のみが許可されています

到達可能なコンポーネント `owasp.sat.agoat.DownloadInvoiceService$downloadFile$1`（`owasp.sat.agoat.DownloadInvoiceService` で検出）からTaintを追跡しました。const `1` から sink `android.app.DownloadManager$Request android.app.DownloadManager$Request.setAllowedNetworkTypes(int)` まで。Taintトレース：

```
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()
```

メソッド `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;
    }
}
```

```
}
```

メソッド `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;
    }
}
```

推奨事項

アプリケーションは、ユーザーがWi-Fiまたはモバイルネットワークのいずれかを使用できるようにすべきです。一般的な戦略として、Wi-Fi接続時のみ大容量データを取得する方法があります。

中 XMLインジェクション

XMLインジェクションは、信頼できないユーザー入力XMLデータに組み込まれる前に適切に検証またはサニタイズされない場合に発生する脆弱性です。攻撃者は悪意のあるXMLコンテンツを注入することでこの脆弱性を悪用でき、不正アクセス、データ改ざん、またはサービス拒否攻撃を引き起こす可能性があります。

XMLインジェクションは、ユーザー入力をXML設定ファイルやストリームに取り込む前に適切に検証およびサニタイズしていないことから生じる脆弱性です。この見落としにより、攻撃者は悪意のあるXMLコンテンツを注入してアプリケーションを悪用することが可能になります。XML構造を操作したり、悪意のあるエンティティを追加したりすることで、攻撃者はアプリケーションのロジックを妨害したり、データを改ざんしたり、機密情報を抽出したりする可能性があります。

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インジェクションデモ',
      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');

    // XMLデータを作成
    final xmlData = '''
<root>
  <username>${_usernameController.text}</username>
  <password>${_passwordController.text}</password>
</root>
''';

    // XMLを設定ファイルに保存
    await configFile.writeAsString(xmlData);
  }

  void login() {
    final username = _usernameController.text;
    final password = _passwordController.text;

    // ログイン処理を実行
    print('ユーザー名: $username、パスワード: $password でログイン処理を実行');
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('XMLインジェクションデモ'),
      ),
      body: Padding(
        padding: EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text('ユーザー名を入力: '),
            TextField(
              controller: _usernameController,
            ),
            SizedBox(height: 16.0),
            Text('パスワードを入力: '),
            TextField(
              controller: _passwordController,
              obscureText: true,
            ),
            SizedBox(height: 16.0),
            ElevatedButton(
              onPressed: () async {
                await saveConfigFile();
                login();
              },
              child: Text('ログイン'),
            ),
          ],
        ),
      ),
    );
  }
}

```

➤ SWIFT

```
import SwiftUI
```

```

struct ContentView: View {
    @State private var username = ""
    @State private var password = ""

    var body: some View {
        VStack {
            Text("ユーザー名を入力:")
            TextField("ユーザー名", text: $username)
                .textFieldStyle(RoundedBorderTextFieldStyle())

            Text("パスワードを入力:")
            SecureField("パスワード", text: $password)
                .textFieldStyle(RoundedBorderTextFieldStyle())

            Button(action: {
                saveConfigFile()
                login()
            }) {
                Text("ログイン")
            }
        }
        .padding()
    }

    func saveConfigFile() {
        // XMLデータを作成
        let xmlData = '''
<root>
  <username>\(username)</username>
  <password>\(password)</password>
</root>
'''

        // XMLを設定ファイルに保存
        let configFilePath = "config.xml"
        do {
            try xmlData.write(toFile: configFilePath, atomically: true, encoding: .utf8)
        } catch {
            print("XMLファイルの書き込みに失敗しました:\(error)")
        }
    }

    func login() {
        // ログイン処理を実行
        print("ユーザー名: \(username)、パスワード: \(password) でログイン処理を実行")
    }
}

@main
struct XMLInjectionApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}

```

KOTLIN

```

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, "設定ファイルの保存に失敗しました", Toast.LENGTH_SHORT).show()
    }
}

private fun login(username: String, password: String) {
    // ログイン処理を実行
    Toast.makeText(this, "ユーザー名:$username、パスワード:$password でログイン処理を実行", Toast.LENGTH_SHORT).show()
}
}

```

技術的詳細

検出事項 #1

クラス `WebView` のメソッド `loadData` の位置 0 のパラメータがXML injectionに脆弱である可能性があります。到達不可能なコンポーネント `owasp.sat.agoat.XSSActivity` からTaintがトレースされています。TaintElementの発生源：クラス `XSSActivity`、メソッド `onCreate`、位置 0、シグネチャ `(Landroid/os/Bundle;)V`。コールトレース：

```

at android.webkit.WebView.loadData
at owasp.sat.agoat.XSSActivity.onCreate
at owasp.sat.agoat.XSSActivity.onCreate

```

at `android.webkit.WebView.loadData`

Unable to extract source code.

at `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.checkNotNullExpressionValue(v0_3, webView1);
    v0_3.setWebChromeClient(new android.webkit.WebChromeClient());
    android.webkit.WebSettings v1_1 = v0_3.getSettings();
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(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;
    }

```

at `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;
}

```

推奨事項

XMLインジェクション脆弱性を軽減するには、以下を検討してください：

- 適切な入力検証およびサニタイゼーション手法を実装する。
- ユーザー入力を想定される形式に対して検証し、XMLタグやエンティティとして解釈される可能性のある文字を除去またはエスケープしてサニタイズする。
- データバインディングを安全に処理するセキュアなXMLパースライブラリまたはAPIの使用を検討する。

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');

    // XMLデータを作成
    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);

    // XMLを設定ファイルに保存
    await configFile.writeAsString(xmlData);
  }

  void login() {
    final username = _usernameController.text;
    final password = _passwordController.text;

    // ログイン処理を実行
    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"

    // 設定ファイルを読み込む
    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
    }

    // XMLをパースする
    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
    }

    // ユーザー入力から設定値を読み込む
    guard let username = getUserInput(prompt: "Enter username:"), let password = getUserInput(prompt: "Enter password:") else {
        print("Invalid input")
        return
    }

    // XML内の設定値を更新する
    if let usernameElement = root.elements(forName: "username").first {
        usernameElement.stringValue = username
    }

    if let passwordElement = root.elements(forName: "password").first {
        passwordElement.stringValue = password
    }

    // 変更したXMLを設定ファイルに保存する
    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
    }

    // 設定値を使用する
    login(username: username, password: password)
}

func getUserInput(prompt: String) -> String? {
    print(prompt, terminator: " ")
    return readLine()
}

func login(username: String, password: String) {
    // ログイン処理を実行する

```

```

        print("Performing login with username: \"(username)\" and password: \"(password)\"")
    }

    // プログラムを開始する
    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) {
        // ログイン処理を実行する
        Toast.makeText(this, "Performing login with username: $username and password: $password", Toast.LENGTH_SHORT).show()
    }
}

```

- [Testing for XML Injection \(OWASP\)](#)

低 usesCleartextTraffic 属性が設定されています

CVSS v3: 5.4

アプリケーションがusesCleartextTraffic属性を設定しています。

`android:usesCleartextTraffic` 属性は、アプリが平文のネットワーク通信（平文HTTPなど）を使用するかどうかを示します。

API レベル 27 以下をターゲットとするアプリのデフォルト値は "true" です。API レベル 28 以上をターゲットとするアプリでは、デフォルト値が "false" となります。

技術的詳細

検出事項 #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.

推奨事項

`android:usesCleartextTraffic` 属性の値を明示的に `false` に設定し、Android Network Security Config を定義してください。

targetAPI レベルが 27 以下のアプリでは、デフォルト値は `true` です。targetAPI レベルが 28 以上のアプリでは、デフォルトで `false` となります。

```
<? XML

<application android:icon="@drawable/icon" android:usesCleartextTraffic="false">
```

CVSSベクター詳細

CVSS V3 ベクター: CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N

攻撃ベクトル:	☒ ネットワーク (N) ☐ 隣接 (A) ☐ ローカル (L) ☐ 物理 (P)
攻撃の複雑さ:	☒ 低 (L) ☐ 高 (H)
必要な権限:	☐ None (N) ☒ 低 (L) ☐ 高 (H)
ユーザー関与:	☐ None (N) ☒ 要 (R)
スコープ:	☒ 変更あり (C) ☐ 変更なし (U)
機密性:	☐ 高 (H) ☒ 低 (L) ☐ None (N)
完全性:	☐ 高 (H) ☒ 低 (L) ☐ None (N)
可用性:	☐ 高 (H) ☐ 低 (L) ☒ None (N)

参考資料

- [Storage updates in Android 11](#)

低 非推奨のTLS/SSLプロトコルバージョンの使用

アプリケーションが、既知のセキュリティおよびプライバシー上の問題を抱える非推奨のSSL/TLSプロトコルを設定しています。

SSL/TLS ファミリーには6つのプロトコルがあります: SSLv2、SSLv3、TLSv1.0、TLSv1.1、TLSv1.2、およびTLSv1.3:

- SSLv2は安全ではなく、使用してはいけません。このプロトコルバージョンでは、DROWN攻撃により同一名のサイトやRSA鍵への攻撃が可能です。
- SSLv3は、SSLv3 POODLE攻撃のためHTTPで使用する場合に安全ではありません。他のプロトコルで使用する場合も脆弱と見なされます。このプロトコルは非推奨であり、使用してはいけません。
- TLSv1.0およびTLSv1.1はBEAST攻撃の影響を受けるレガシープロトコルです。最新の環境では一部の緩和策が実装されていますが、このプロトコルには依然として既知の弱点が残っており、PCI DSSおよびブラウザにより2020年1月より非推奨とされています。
- TLSv1.2およびv1.3に既知のセキュリティ問題はありません。

技術的詳細

TLS_1_0_DEPRECATED detected on demo.testfire.net

- **Finding:** TLS 1.0 offered: offered (deprecated)

推奨事項

TLS v1.2 または TLS v1.3 を主要なサポート対象プロトコルとすべきです。これらのバージョンは最新の認証付き暗号を提供するためです。

参考資料

- [TLSv1.0 and TLSv1.1 deprecation](#)
- [SSL and TLS deployment best practices](#)

低

Google Calendar - 公開

対象に埋め込まれた Google Calendar が公開されている状態で検出され、会議の詳細、参加者名、イベント予定、社内組織データなどの機密情報が漏洩する可能性があります。

対象に埋め込まれた公開状態の Google Calendar が検出されました。会議の詳細、参加者名、イベント予定、社内組織データなどの機密情報が漏洩する可能性があります。

技術的詳細

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 (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/"
}
```

検出事項 #2

Google Calendar - Exposure:

- <https://calendar.google.com/calendar/embed?src=hl6cjs6ep1h7oniqqueu2bhbo%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"
}
```

低 非推奨のTarget APIバージョン

CVSS v3: 5.4

アプリケーションはtargetSdkVersion属性を古いAndroid APIレベルの使用を許可するように設定しており、ユーザーをセキュリティ脆弱性にさらす可能性があります。

`android:targetSdkVersion` 属性は、アプリケーションに必要なAndroid Target APIレベルを指定します。`targetSdkVersion` を低く設定すると、アプリケーションは古いAndroidバージョンでの実行が可能になりますが、ユーザーをセキュリティ脆弱性に晒す可能性があります。

非推奨に関する通知は以下のリンクを参照してください: <https://support.google.com/googleplay/android-developer/answer/11926878?hl=en>

技術的詳細

検出事項 #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"/>
```

推奨事項

`android:targetSdkVersion` 属性に、より新しいAndroid APIレベルに対応する値を設定することを検討してください。これにより、アプリケーションはセキュリティ改善の恩恵を受けられ、古いAndroidリリースに関連する脆弱性のリスクを低減できます。

CVSSベクター詳細

CVSS V3ベクター： CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N

攻撃ベクトル:

品 ネットワーク (N) 隣接 (A) ローカル (L) 物理 (P)

攻撃の複雑さ:

低 (L) 高 (H)

必要な権限:

None (N) 低 (L) 高 (H)

ユーザー関与:

None (N) 要 (R)

スコープ:

変更あり (C) 変更なし (U)

機密性:

高 (H) 低 (L) None (N)

完全性:

高 (H) 低 (L) None (N)

可用性:

高 (H) 低 (L) None (N)

参考資料

- [Android API Levels](#)
- [Target API level deprecation notice](#)

低

属性 hasFragileUserData が未設定です

CVSS v3: 5.4

アプリケーションが hasFragileUserData 属性を明示的に設定していません。

`android:hasFragileUserData` は、Androidにおいてアプリが保護を必要とする脆弱なユーザーデータを含むかどうかを開発者が指定する属性です。

脆弱なユーザーデータとは、紛失・盗難・不正利用された場合にユーザーに危害または損害を与える可能性があるあらゆるデータを指します。

これには、社会保障番号、クレジットカード番号、医療記録、その他の機密データなど、機密性の高い個人情報が含まれます。

この値は、アプリケーションが重要なユーザーデータを扱うかどうかを明示するため、アプリケーション内で明示的に設定されるべきです。

値が true の場合、ユーザーがアプリをアンインストールする際、アプリのデータを保持するかどうかを確認するプロンプトが表示されます。

技術的詳細

検出事項 #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">
```

推奨事項

`hasFragileUserData` フラグは、アプリケーションの `AndroidManifest.xml` ファイルに追加できます。

</ XML

```
<application android:icon="@drawable/icon" android:hasFragileUserData="true">
```

CVSS V3ベクター：	CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N
攻撃ベクトル:	品 ネットワーク (N) 隣接 (A) ローカル (L) 物理 (P)
攻撃の複雑さ:	低 (L) 高 (H)
必要な権限:	None (N) 低 (L) 高 (H)
ユーザー関与:	None (N) 要 (R)
スコープ:	変更あり (C) 変更なし (U)
機密性:	高 (H) 低 (L) None (N)
完全性:	高 (H) 低 (L) None (N)
可用性:	高 (H) 低 (L) None (N)

参考資料

- Android Developer - Fragile User Data

低 タスクハイジャック

アプリケーションはタスクハイジャックに対する保護がなく、デバイス上の悪意あるアプリケーションがそのフォアグラウンドタスクを乗っ取る可能性があります。

Androidタスクは、ユーザーが特定の作業を行う際にインタラクトするアクティビティの集合です。異なるアプリのアクティビティが同じタスク内に存在できるため、以下のパラメータを操作することで、悪意あるアクティビティを対象アプリケーションのタスクに移動させる可能性があります。

- Task Affinity**：taskAffinity 属性で制御
- Task Reparenting**：allowTaskReparenting 属性で制御

Task Affinity は、AndroidManifest.xml ファイル内の <activity> タグで定義されるアクティビティ属性です。Task Affinity は、アクティビティが参加を希望するタスクを指定します。デフォルトでは、アプリ内のすべてのアクティビティは同じ affinity を持ち、これはアプリのパッケージ名となります。

</ 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>
```

allowTaskReparenting がアクティビティ A に対して true に設定されており、かつ同じ affinity を持つ新しいタスクが前面に表示されると、システムは再配置可能なアクティビティ A を元のホスティングタスクから新しいフォアグラウンドタスクスタックへ移動します。

タスクハイジャック攻撃には、いくつかの手法があります。

- Task Affinity Control**：アプリケーションのパッケージ名が com.mySecureApp.app で、アクティビティ A1 を持つとします。悪意あるアプリケーションは M1 と M2 の 2 つのアクティビティを持ち、M2.taskAffinity = com.mySecureApp.app かつ M2.allowTaskReparenting = true と設定されています。悪意あるアプリが M2 で開かれている状態で対象アプリケーションを起動すると、M2 が前面に移動し、ユーザーは悪意あるアプリケーションとインタラクトすることになります。
- Single Task Mode**：アプリケーションの起動モードが singleTask に設定されている場合、M2.taskAffinity = com.mySecureApp.app とした悪意あるアプリケーションは、対象アプリケーションのタスクスタックをハイジャックできます。
- Task Reparenting**：アプリケーションが taskReparenting を true に設定している場合、悪意あるアプリケーションは対象アプリケーションのタスクを悪意あるアプリケーションのスタックに移動させることができます。

タスクハイジャックはフィッシングや使用拒否攻撃に悪用可能であり、過去には銀行を狙ったマルウェアトロイの木馬によって悪用されてきました。新しい攻撃バリエーション (StandHogg 2.0) はコードベースの攻撃であるため、検出は極めて困難です。

タスクハイジャックは Android 11 で対策されており、これは WindowManager の confused deputy に関する CVE-2020-0267 の修正に含まれています。

技術の詳細

検出事項 #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"/></intent-filter></activity>
```

検出事項 #2

The activity `owasp.sat.agoat.AccessControlIssue1Activity` 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.AccessControlIssue1Activity"/>
```

検出事項 #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"/>
```

検出事項 #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"/>
```

検出事項 #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"/>
```

検出事項 #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"/>
```

検出事項 #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"/>
```

検出事項 #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"/>
```

検出事項 #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"/>
```

検出事項 #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"/>
```

検出事項 #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"/>
```

検出事項 #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"/>
```

検出事項 #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"/>
```

検出事項 #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"/>
```

検出事項 #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"/>
```

推奨事項

Task Hijacking 脆弱性には複数の形態があり、それぞれに対応した修正が必要です：

- `AndroidManifest.xml` の `<activity>` タグでアプリケーションアクティビティの `task affinity` を `""`（空文字列）に設定し、アクティビティがランダムに生成された `task affinity` を使用するよう強制するか、`<application>` タグで設定してアプリケーション内のすべてのアクティビティに適用します。

または

- `android:launchMode` を `singleInstance` に設定します。 `singleInstance` は、同じタスクに他のアクティビティが作成されないことを保証します。
- `singleTask` を launch mode に指定しないか、悪意のあるフォアグラウンドタスクの存在を検出する監視サービスを導入してください。
- アクティビティ起動Intentで `FLAG_ACTIVITY_NEW_TASK` フラグを設定しないか、設定する場合は `FLAG_ACTIVITY_CLEAR_TASK` と併用してください：

🔗 JAVA

```
Intent i = new Intent(this, AnActivity.class);
i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
startActivity(i);
```

- `taskAffinity` とともに `allowReparenting` を指定しないか、悪意のあるフォアグラウンドタスクの存在を検出する監視サービスを導入してください。
- Explicit Intent の使用を優先してください。Explicit Intent では、対象アプリケーションのパッケージ名または完全修飾コンポーネントクラス名を指定することで、どのアプリケーションがIntentを処理するかを明確にします。Implicit Intent は一般的なアクションのみを指定します。

参考資料

- [Understand Tasks and Back Stack - Android Documentation](#)
- [Towards Discovering and Understanding Task Hijacking in Android](#)
- [StrandHogg Attack](#)
- [StandHogg Attack 2.0](#)
- [Task Hijacking exploited by Mobile Banking Malware](#)
- [CVE-2020-0267: WindowManager Confused Deputy](#)

潜在的

ANALYTICS_LIBRARY におけるネットワークセキュリティ設定の不備およびトランスポート保護の欠如

ANALYTICS_LIBRARY サロゲート (TrafficActivity) は、デフォルトの未ピン留めOkHttpClientを介してアナリティクストラフィックを送信している。一方、Network Security Configurationはグローバルに明文HTTPを許可し、ユーザーインストールのCAを信頼している。このパイプラインにはペイロード暗号化、リクエスト署名、および整合性制御が欠如しており、MitMによる傍受、リプレイ、および改ざんが容易に行える。

スコープされたANALYTICS_LIBRARY コンポーネント (`owasp.sat.agoat.TrafficActivity`) は、証明書ピンング、カスタムトラスト管理、ペイロード暗号化を行わずに初期化されたデフォルトの `OkHttpClient` を使用して、外部へのネットワークリクエストを発行しています。また、アプリケーションの `network_security_config.xml` はすべての宛先に対して明文トラフィックを明示的に有効化し、ユーザーインストールの認証局を信頼する設定となっており、これによりデバイスのネイティブなTLS保護が無効化されています。すべての29個のファーストパーティクラスに対する徹底的な静的解析により、analytics送信パイプラインにおいてペイロード暗号化 (AES/RSA/DDES)、HMAC 署名、リクエストチェックサム、OkHttpインターセプタ、nonce生成のいずれも確認されませんでした。現状のビルドでは空のパラメータなしGETリクエストのみを送信し、機密性の高いユーザーデータを流出させていないものの、トランスポートスタックはアーキテクチャレベルで破綻しています。暗号学的な検知なしに、あらゆるネットワーク攻撃者がanalyticsイベントを傍受、リプレイ、改変、または合成可能です。本番用のanalytics SDKがこのコードベースに導入された場合、同じ明文通信のパーミッションとピンングされていないTLSチャネルを継承し、機密性と完全性が即座に侵害されることになります。

技術的詳細

検出事項 #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) {
  Intrinsic.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/UHC1HJBfIdeKuyHm/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.

推奨事項

即時対策

- **平文通信の制限:** グローバルな `<base-config cleartextTrafficPermitted="true">` を、すべての analytics エンドポイントで `cleartextTrafficPermitted="false"` とする明示的な `<domain-config>` エントリに置き換えてください。特定のレガシーエンドポイントで平文が絶対に必要な場合は、その単一ドメインにのみスコープを厳密に限定してください。
- **ユーザーCA信頼の削除:** プロダクションビルドのベース設定から `<certificates src="user" />` を削除してください。内部テストでプロキシCAが必要な場合は、`<debug-overrides>` を使用してデバッグビルドフレーバーに限定し、テストデバイスのみにスコープしてください。

恒久対策

- **デフォルト証明書ピンニングの実装:** `TrafficActivity` で使用されるデフォルトの `OkHttpClient` に、すべての analytics エンドポイントについて少なくとも2つの SHA-256 バックアップピンを含む `CertificatePinner` を設定してください:

```
CertificatePinner pinner = new CertificatePinner.Builder()
    .add("analytics.example.com", "sha256/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA")
    .add("analytics.example.com", "sha256BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB")
    .build();

OkHttpClient client = new OkHttpClient.Builder()
    .certificatePinner(pinner)
    .build();
```

- **ペイロードの暗号化と署名の有効化**： analytics ライブラリが機密データを送信する場合は、リクエストボディを AES-GCM で暗号化し、暗号文に対して HMAC-SHA256 署名を付与してください。リプレイ攻撃を防ぐため、モノトニックな nonce またはタイムスタンプを含めてください。
- **デバッグフラグの無効化**： プロダクション用の `AndroidManifest.xml` で `android:debuggable="false"` および `android:allowBackup="false"` を設定してください。

検証

- APKを再デコンパイルし、`res/xml/network_security_config.xml` のベース設定に `cleartextTrafficPermitted="true"` や `<certificates src="user" />` が含まれていないことを確認してください。
- `TrafficActivity.java` がデフォルトの `OkHttpClient` を `new OkHttpClient.Builder().certificatePinner(...).build()` で初期化していることを確認してください。
- デコンパイルしたソース内で `Cipher.getInstance()` および `Mac.getInstance()` を検索し、暗号化および署名ルーティンがanalyticsコールチェーン内に存在することを確認してください。
- ユーザーインストール済みCAプロキシでトラフィックを傍受し、ピンングの不一致によりアプリが `SSLPeerUnverifiedException` で接続を拒否することを確認してください。

潜在的 アプリケーションに保存された機密情報

アプリケーション内にパスワード、トークン、その他の機密情報が保存されています。

シークレット、パスワード、API キーなどの漏洩は、重大な影響を及ぼす可能性があります。

- **機密性の喪失:** シークレットやパスワードが漏洩すると、権限のない者が機密情報にアクセスする可能性があります。
- **完全性の喪失:** 漏洩したシークレットやパスワードを悪用してシステムやデータに不正アクセスし、情報の改ざんや破損を引き起こす可能性があります。
- **可用性の喪失:** 漏洩したシークレットやパスワードの不正利用により、正当なユーザーのアクセスが拒否され、対象のシステムやデータの可用性が損なわれる場合があります。
- **評判毀損:** 機密情報を適切に保護できなかったことが明らかになれば、企業や個人の評判が損なわれる可能性があります。
- **法的責任:** 漏洩した情報の性質や該当する法域の法律に応じて、シークレットやパスワードの漏洩が法的責任を招く可能性があります。
- **過剰課金:** APIキーの漏洩により、権限のない第三者がAPIリソースにアクセスし、課金対象の操作を実行する可能性があります。過剰課金に繋がるリスクがあります。

技術的詳細

検出事項 #1

Secret Detected: Google OAuth Access Token

File: <https://android.googleapis.com:443//auth>

Detected Secret: ya29.m

検出事項 #2

Secret Detected: Generic Username/Password

File: <https://github.com:443//satishpatnayak/MyTest/blob/master/AndroGoatInvoice.txt>

Detected Secret: Login": "satishpatnayak

検出事項 #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×tamp=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_AuopgbxleQNXUJKLW5CluuGX7ujD1o
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=ChIJNlt_tDeuEmsRUsoyG83frY4&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
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&near par&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"]}'

検出事項 #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_tDeuEmsRUsoyG83frY4&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
Text Search- Places	✗	\$2.83/1.000 reqs	curl https://maps.googleapis.com/maps/api/place/queryautocomplete/json?input=pizza+near&par&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:bggehv7mz6y&q=test
Firebase Cloud Messaging	✗	NA	curl -i -s -X POST --header "Authorization: key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w" --header "Content-Type: applicat" https://fcm.googleapis.com/fcm/send -d '{"registration_ids": ["1"]}'

推奨事項

機密データはアプリケーション本体に含めないでください。必要に応じてこのデータにアクセスできるよう、サービスの認証情報を安全に暗号化・保存・取得する方法を採用してください。

過剰請求のリスクを防ぐため、使用量に応じてコストが高額になりうるサービスでは、API key pinning の実装や認証を必要とする API の利用を検討してください。API key pinning は暗号署名を要求することで、特定のアプリケーションにキーの使用を制限する仕組みで、サービスプロバイダー（例: Google Maps）側で有効化できます。

不正アクセスのリスクがあるキーについては、権限とロールを非クリティカルな機能に制限するか、認証を必要とするAPIを介してサービスを公開することが重要です。

さらにセキュリティを強化するため、APIには適切なアクセス制御とレート制限を設けるとともに、キーを定期的にローテーションして不正利用を防止してください。

参考資料

- [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](#)

潜在的 パストラバーサル

アプリケーション内のパストラバーサルの存在は、攻撃者がファイルパスの一部を操作できる脆弱性を示しています。この改変されたパスは、適切な検証や精査なしにファイルシステムAPIによって利用されます。

アプリケーションが、意図された境界を超えたファイルシステムへのアクセスを攻撃者に許可する可能性があります。これにより、機密ファイルやディレクトリへの不正アクセスが発生するおそれがあります。攻撃者はファイルパスを操作することでアクセス制御を回避し、構成ファイル、ユーザークレデンシャル、機密文書などの重要データを取得または改変できます。この脆弱性は重大な脅威であり、攻撃者による権限昇格、任意コード実行、またはシステムへの追加攻撃を可能にします。

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}")
    }
}
```

技術の詳細

検出事項 #1

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.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$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$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;
}

```

検出事項 #2

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$0.getApplicationInfo().dataDir));
        v2_0.setReadable(1);
        v2_0.setWritable(1);
        CharSequence v3_1 = new java.io.PrintWriter(v2_0);
        CharSequence v4_1 = new StringBuilder();
        v4_1.append(username is );
        String v5_1 = this.$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$0.getApplicationInfo().dataDir));
        v2_0.setReadable(1);
        v2_0.setWritable(1);
        CharSequence v3_1 = new java.io.PrintWriter(v2_0);
        CharSequence v4_1 = new StringBuilder();
        v4_1.append(username is );
        String v5_1 = this.$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()` :

```

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;
}
```

検出事項 #3

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;
}
```

検出事項 #4

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.$username;
kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v1_6, username);
v0_1.append(v1_6.getText().toString());
v0_1.append('');
String v0_2 = v0_1.toString();
try {
    owasp.sat.agoat.SQLInjectionActivity.access$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$0.getApplicationContext();
    StringBuilder v4_10 = new StringBuilder();
    v4_10.append(Data not saved: Error occurred - );
    v4_10.append(v2_13.getMessage());
    android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
    android.util.Log.v(Query, v0_2);
    return;
}
if (v3_2 == null) {
    kotlin.jvm.internal.Intrinsics.throwNpe();
}
android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
StringBuilder v4_5 = new StringBuilder();
if (v3_3 == null) {
    String v5_13 = new StringBuilder();
    v5_13.append(User: ());
    android.widget.EditText v6_2 = this.$username;
    kotlin.jvm.internal.Intrinsics.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;
}

```

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;
}

```

推奨事項

パス包含制御: パスを正規化し、対象が宛先ディレクトリ内に含まれているかどうかを確認します。

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("Invalid file path");
        return;
    }
}

```

```

    }

    inputFile.copy(outputFile.path)
        .then(()) => print("File copied successfully"))
        .catchError((error) => print("Error: $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

    // Check if the fileURL is within the allowed directory
    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("File contents: \(contents)")
    } else {
        print("Failed to read file.")
    }
}

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("Enter the file name:")
    val fileName = readLine()

    try {
        val content = fileName?.let { readFile(it) }
        println("File content: $content")
    } catch (e: Exception) {
        println("Error: ${e.message}")
    }
}

```

絶対パスの使用: 可能な限り、相対パスではなく絶対パスの使用を推奨します。絶対パスを使用することで、アプリケーションはファイルやディレクトリの正確な場所を明示的に指定でき、解釈の余地を残しません。

DART

```
import 'dart:io';
import 'package:path/path.dart' as path;

void main() {
  final absolutePath = '/path/to/file.txt';
  var file = File(absolutePath);

  // print('File name: ${path.basename(dir.file.path)}');
  print('File name: ${path.basename(file.path)}');
}
```

🔗 SWIFT

```
import Foundation

func main() {
  let absolutePath = "/path/to/file.txt"
  let fileManager = FileManager.default

  if fileManager.fileExists(atPath: absolutePath) {
    // Perform operations on the file
    print("File exists at \(absolutePath)")
  } else {
    print("File not found at \(absolutePath)")
  }
}

main()
```

🔗 KOTLIN

```
import java.io.File

fun main() {
  val absolutePath = "/path/to/file.txt"
  val file = File(absolutePath)

  if (file.exists()) {
    // Perform operations on the file
    println("File exists at $absolutePath")
  } else {
    println("File not found at $absolutePath")
  }
}
```

参考資料

- [Improper Limitation of a Pathname to a Restricted Directory](#)

潜在的

アプリケーションデータの不安全な保存

アプリケーションデータの安全でない保存により、他のアプリによるデータの読み取りおよび改ざんが可能な状態。

モバイルアプリケーションにおける不安全なストレージの脆弱性とは、ユーザーの認証情報、個人情報、その他の機密データといった機密性の高いデータが、デバイス上の world-readable な領域に保存されるセキュリティ上の欠陥を指します。これにより、デバイス上の他のアプリケーションや、デバイスに物理的にアクセスできる攻撃者によって、データが不正に取得されるリスクが生じます。

不安全なストレージは、アプリケーションが world-readable な領域に書き込む場合に限らず、アプリケーションが設定ファイルなどのデータを world-writable な領域から読み込む場合にも発生します。world-writable な領域では、攻撃者が設定ファイルを書き換えることで、アプリケーションの動作を改竄する可能性があります。

🔗 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 {
            // Dummy authentication process
            val token = authenticate("username", "password") // authentication logic

            // Store token in public storage
            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()
        // 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 in public storage
        storeTokenInPublicStorage(token: token)
    }

    private fun 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: () {
            // Dummy authentication process
            String token = authenticate("username", "password");

            // Store token in public storage
            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.');
```

技術の詳細

検出事項 #1

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.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;
}

```

検出事項 #2

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.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$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;
}

```

検出事項 #3

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;
}

```

検出事項 #4

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.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$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;
}

```

検出事項 #5

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.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$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;
}

```

検出事項 #6

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.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password -);
        java.io.File v1_4 = this.$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()` :

```

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;
}

```

検出事項 #7

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.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$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;
}

```

検出事項 #8

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.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$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$.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;
}

```

推奨事項

- セキュアストレージの利用: 認証トークンや認証情報などの機密データは、安全に保存してください。iOSのKeychainやAndroidの暗号化対応SharedPreferencesなど、プラットフォームが提供するセキュアストレージの仕組みを利用してください。
- データの暗号化: ユーザーデバイスに保存される機密データを、強力な暗号化アルゴリズムを用いて保存時に暗号化し、保護してください。
- 機密情報のハードコーディングを避ける: ユーザー名、パスワード、トークンなどの機密情報をコードに直接記述するのではなく、前述のセキュアストレージの仕組みを利用することを検討してください。
- 過度に緩いファイルパーミッションを避ける: 機密ファイルやディレクトリへのアクセスを制限するよう、ファイルパーミッションを適切に設定してください。例えばAndroidでは、ファイル作成時に `MODE_WORLD_READABLE` や `MODE_WORLD_WRITEABLE` を使用しないでください。これらはすべてのアプリケーションに対して広範な読み取りまたは書き込みアクセスを許可してしまいます。ファイルパーミッションの設定では常に最小権限の原則に従い、アプリケーションの機能に必要な範囲のみへアクセスを限定してください。
- 入力検証: パブリックストレージ領域からデータを読み込む際は、必ず検証とサニタイズを行ってください。

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.');  }  
}
```

参考資料

- [CWE-359: Exposure of Private Information \("Privacy Violation"\)](#)
- [OWASP Mobile Top 10](#)
- [CWE-200: Information Exposure](#)

潜在的

バックアップモードが有効です

CVSS v3: 3.9

アプリケーションがバックアップモードを有効化しています。

バックアップ機能は、Androidにおいてユーザーがデータと設定をデバイス間でバックアップおよび復元できる機能です。デフォルトでは、Androidは **/data** パーティションに保存されたプライベートファイルを含むアプリケーションの完全バックアップを実行します。Androidデバイスでバックアップ機能が有効になっている場合、**Backup Manager Service** はアプリデータ、Wi-Fiパスワード、その他の設定などを、ユーザーの Google Drive アカウントに定期的にアップロードします。このデータは同じデバイスに復元できるほか、ユーザーが同じ Google アカウントにサインインしていれば別のデバイスにも復元できます。

Androidの Backup Manager Service の主な機能は以下の通りです。

1. **自動バックアップ**: Backup Manager Service は定期的にアプリデータを自動バックアップします。通常は1日に1回ですが、デバイスが電源に接続された場合やユーザーが手動で開始した場合など、特定のイベント発生時にもバックアップをトリガーできます。

2. **インクリメンタルバックアップ:** Backup Manager Service はインクリメンタルバックアップを実行します。これは前回のバックアップ以降に変更があったデータのみを対象とするため、バックアップサイズの削減と処理の高速化を実現します。
3. **暗号化バックアップ:** Backup Manager Service が作成するバックアップは、各ユーザーのデバイス固有のキーで暗号化されます。これにより、バックアップデータにアクセスできるのはユーザー本人、またはその Google アカウントの認証情報を持つ者に限定されます。
4. **アプリ固有のバックアップ:** Backup Manager Service では、アプリごとにバックアップ対象のデータと除外するデータを指定できます。これによりバックアップサイズを抑えつつ、機密データが含まれないようにすることが可能です。
5. **復元機能:** Backup Manager Service には、新規またはファクトリーリセット済みのデバイスにアプリデータを復元する機能のほか、既存デバイス上でアプリをアンインストールして再インストールした後にデータを復元する機能も備わっています。

技術的詳細

検出事項 #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">
```

推奨事項

アプリのマニフェストファイルで `android:allowBackup` 属性を設定し、バックアップの有効/無効を切り替えます。デフォルト値は true ですが、意図を明確にするため、以下に示すようにアプリケーションのマニフェストに明示的に設定することを推奨します。

<> XML

```
<manifest>
  <application android:allowBackup="true">
  </application>
</manifest>
```

アプリケーションにバックアップや復元の対象にしたいくない機密データが含まれる場合は、 `android:allowBackup` 属性を false に設定することで、以下に示すようにバックアップを無効にできます。

<> XML

```
<manifest>
  <application android:allowBackup="false">
  </application>
</manifest>
```

CVSSベクター詳細

CVSS V3ベクター： CVSS:3.0/AV:P/AC:L/PR:H/UI:N/S:U/C:H/I:N/A:N

攻撃ベクトル:

🌐 ネットワーク (N) 🏠 隣接 (A) 🏠 ローカル (L) 🚒 物理 (P)

攻撃の複雑さ:

🔑 低 (L) 🔑 高 (H)

必要な権限:

🔑 None (N) 🔑 低 (L) 🔑 高 (H)

ユーザー関与:

🔑 None (N) 🔑 要 (R)

スコープ:

🏠 変更あり (C) 🏠 変更なし (U)

機密性:

🔑 高 (H) 🔑 低 (L) 🔑 None (N)

完全性:

🔑 高 (H) 🔑 低 (L) 🔑 None (N)

可用性:

🔑 高 (H) 🔑 低 (L) 🔑 None (N)

参考資料

- [Random Musings on the M Developer Preview: the Ugly \(Part Two\)](#)
- [DRD22: Do not cache sensitive information](#)

危険なSinkメソッドが、ユーザが制御しうるパラメータまたは信頼できないSourceメソッドを用いて呼び出されています。

ソースメソッドまたはユーザー制御可能なパラメータが、シンクメソッドを呼び出すために使用されています。

ソースとは、非信頼ユーザーから入力される可能性のある、信頼できないデータ入力を指します。シンクとは、攻撃者からアクセス可能な場合に悪用されて攻撃が実行される可能性がある、危険なメソッドを指します。

ソースおよびシンクは、インジェクション、間接オブジェクト参照、未承認データアクセスなどの脆弱性についてレビューする必要があります。

技術的詳細

検出事項 #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.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;  
}
```

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.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.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;  
}
```

検出事項 #2

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 `void` `owasp.sat.agoat.AccessControlIssue1Activity.startActivity(android.content.Intent)`. Taint trace:

```
at owasp.sat.agoat.AccessControlIssue1Activity.startActivity()
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick()
at owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView()
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.<init>()
at owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView()
at owasp.sat.agoat.AccessControlIssue1Activity$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$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;
}
```

Method `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;
}
```

Method `owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.<init>()` :

```
AccessControlIssue1Activity$verifyPINView$1(owasp.sat.agoat.AccessControlIssue1Activity p1)
{
    this.this$0 = p1;
    return;
}
```

Method `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;
}
```

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.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.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;
}

```

検出事項 #3

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 `void` `owasp.sat.agoat.AccessControlIssue1Activity.startActivity(android.content.Intent)`. Taint trace:

```

at owasp.sat.agoat.AccessControlIssue1Activity.startActivity()
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick()
at android.content.Intent.<init>()
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.onClick()
at owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView()
at owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.<init>()
at owasp.sat.agoat.AccessControlIssue1Activity.verifyPINView()
at owasp.sat.agoat.AccessControlIssue1Activity$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$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;
}

```

Method `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;
}

```

Method `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;
}

```

Method `owasp.sat.agoat.AccessControlIssue1Activity$verifyPINView$1.<init>()` :

```

AccessControlIssue1Activity$verifyPINView$1(owasp.sat.agoat.AccessControlIssue1Activity p1)
{
    this.this$0 = p1;
    return;
}

```

Method `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;
}

```

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.checkNotNull(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$Delegate v0 = this.getDelegate();
    v0.installViewFactory();
    v0.onCreate(p5);
    if ((v0.applyDayNight()) && (this.mThemeId != 0)) {
        if (android.os.Build$VERSION.SDK_INT < 23) {
            this.setTheme(this.mThemeId);
        } else {
            this.onApplyThemeResource(this.getTheme(), this.mThemeId, 0);
        }
    }
    super.onCreate(p5);
    return;
}

```

Method `owasp.sat.agoat.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;
}

```

検出事項 #4

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.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;
    }
}

```

検出事項 #5

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.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.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;
}

```

検出事項 #6

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.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.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$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;
}

```

検出事項 #7

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.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.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$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;
}
```

検出事項 #8

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$1.onClick()
at java.io.FileWriter.<init>()
at owasp.sat.agoat.InsecureStorageSDCardActivity$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$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.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;
}

```

検出事項 #9

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.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.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;
}

```

検出事項 #10

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.$username;  
        kotlin.jvm.internal.Intrinsics.checkNotNull(v1_0, username);  
        v0_1.append(v1_0.getText().toString());  
        v0_1.append( Password - );  
        java.io.File v1_4 = this.$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$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;
}
```

検出事項 #11

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;
}

```

検出事項 #12

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$1.onClick()
at owasp.sat.agoat.InsecureStorageSDCardActivity$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;
}

```

検出事項 #13

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.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$0.getApplicationInfo().dataDir));
        v2_0.setReadable(1);
        v2_0.setWritable(1);
        CharSequence v3_1 = new java.io.PrintWriter(v2_0);
        CharSequence v4_1 = new StringBuilder();
        v4_1.append(username is );
        String v5_1 = this.$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$0.getApplicationInfo().dataDir));
        v2_0.setReadable(1);
        v2_0.setWritable(1);
        CharSequence v3_1 = new java.io.PrintWriter(v2_0);
        CharSequence v4_1 = new StringBuilder();
        v4_1.append(username is );
        String v5_1 = this.$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.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()` :

```

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;
}

```

```
}
```

検出事項 #14

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.$ip;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, ip);
    v0_1.append(v1_4.getText().toString());
    String v0_2 = v0_1.toString();
    StringBuffer v1_2 = new StringBuffer();
    try {
        android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_1, Runtime.getRuntime().exec(ip1, null, null));
        v2_1.waitFor();
        String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_3, reader.readLine());
    } catch (android.widget.TextView v2_2) {
        v2_2.printStackTrace();
        android.widget.TextView v2_3 = this.$result;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(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.checkExpressionValueIsNotNull(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.checkExpressionValueIsNotNull(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.checkExpressionValueIsNotNull(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;
}

```

```
}
```

検出事項 #15

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.$ip;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, ip);
    v0_1.append(v1_4.getText().toString());
    String v0_2 = v0_1.toString();
    StringBuffer v1_2 = new StringBuffer();
    try {
        android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);
        kotlin.jvm.internal.Intrinsics.checkNotNull(v2_1, Runtime.getRuntime().exec(ip1, null, null));
        v2_1.waitFor();
        String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();
        kotlin.jvm.internal.Intrinsics.checkNotNull(v4_3, reader.readLine());
    } catch (android.widget.TextView v2_2) {
        v2_2.printStackTrace();
        android.widget.TextView v2_3 = this.$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$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;
}
```

推奨事項

推奨事項は、特定された脆弱性の種類によって異なります。

潜在的 パーミッションなしで宣言されたサービス

CVSS v3: 3.4

宣言されたサービスがグローバルパーミッションで保護されていません。

`service` は、ユーザーの操作なしにバックグラウンド処理を実行できるアプリケーションコンポーネントです。また、他のアプリケーションに機能を公開するためにも使用されます。これは、サービスへの接続を確立してインタラクトするための `Context.bindService()` の呼び出しに相当します。

保護されていないサービスは他のアプリケーションから呼び出される可能性があり、機密情報へのアクセスや特権アクションの実行を許容する危険性があります。

技術的詳細

検出事項 #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"/>
```

推奨事項

`service` は外部コンポーネントに対して複数のメソッドを公開できます。 `checkPermission` メソッドを使用して、各メソッドに対して任意のパーミッションを定義することも可能です。

また、マニフェストの `<service>` タグでパーミッションを適用することで、サービスを分離しアクセスを制限することも可能です。

```
<? 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>
```

サービスは、個別のIPC呼び出しの実装を実行する前に `checkCallingPermission` メソッドを呼び出すことで、その呼び出しに対してパーミッションを適用できます。

CVSSベクター詳細

CVSS V3ベクター：

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

攻撃ベクトル:

品 ネットワーク (N)

🔌 間接 (A)

🏠 ローカル (L)

🏢 物理 (P)

攻撃の複雑さ:

🔑 低 (L)

🔒 高 (H)

必要な権限:

👤 None (N)

👤 低 (L)

👑 高 (H)

ユーザー関与:

👤 None (N)

👤 要 (R)

スコープ:

🏠 変更あり (C)

🏠 変更なし (U)

機密性:

🔍 高 (H)

🔍 低 (L)

🔍 None (N)

完全性:

🔍 高 (H)

🔍 低 (L)

🔍 None (N)

可用性:

🔍 高 (H)

🔍 低 (L)

🔍 None (N)

参考資料

- [Service \(Android Developer Documentation\)](#)
- [CWE-280: Improper Handling of Insufficient Permissions or Privileges](#)
- [Security Decisions Via Untrusted Inputs \(OWASP Mobile Top 10\)](#)

潜在的 URL操作

アプリケーションがコンテンツを取得するために使用するURLが攻撃者に制御されると、コンテンツの改竄が可能となり、悪性コードの実行や機密情報の窃取に至るセキュリティリスクが生じる。

アプリケーションがコンテンツを取得するために使用するURLを攻撃者が制御できる状態になると、重大なセキュリティリスクが生じます。URLを操作することで、攻撃者はアプリケーションを悪意のあるサーバーにリダイレクトしたり、レスポンスへ任意のコンテンツを注入したりできます。これにより、さまざまなセキュリティ脆弱性や深刻な影響が生じる可能性があります。具体的には以下の通りです。

- コード実行：攻撃者はURLを改変して自身が管理するサーバーを指すようにし、アプリケーションに悪意のあるコードを配信できる可能性があります。このコードはアプリケーションの脆弱性を悪用し、任意のコマンドを実行したり、ユーザーのシステムにマルウェアをインストールしたりする恐れがあります。
- データ窃取：アプリケーションを不正なサーバーにリダイレクトすることで、攻撃者はユーザーにログイン認証情報、クレジットカード情報、個人情報などの機密データの inputs を誘導し、取得した情報を悪用できる可能性があります。
- コンテンツ改竄：攻撃者はアプリケーションが取得したコンテンツを改変し、表示情報を変更したり、悪意のあるスクリプトを注入したりできます。これにより、誤解を招く情報の表示、Web ページの改ざん、フィッシング攻撃などの影響が生じる可能性があります。
- サプライチェーン攻撃：アプリケーションがライブラリやプラグインなどの外部ソースからコンテンツを取得する場合、URLを制御することで、攻撃者は正当なリソースを侵害されたバージョンや悪意のあるバージョンに置き換えることが可能になり、システム全体のセキュリティを損なう恐れがあります。

以下は、攻撃者が制御する inputs を取得するアプリケーションのサンプルコードです。

```
DART

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

class MainActivity extends StatefulWidget {
```

制限付き • Ostorlab

ページ: 178 / 189

```

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 fun 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
}
}

```

技術の詳細

検出事項 #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.$webview11.getSettings();
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v0_1, webSettings1);
    v0_1.setJavaScriptEnabled(1);
    String v1_4 = this.$url;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(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.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\$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$OnClickListener) new
owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1(((android.webkit.WebView) v1_2), ((android.widget.TextView)
this.findViewById(2131165367))))));
    return;
}
```

検出事項 #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$OnClickListener) new
owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1(((android.webkit.WebView) v1_2), ((android.widget.TextView)
this.findViewById(2131165367))))));
    return;
}

```

推奨事項

アプリケーションをURL改竄攻撃から保護するため、事前対策を講じることが極めて重要です。以下の推奨事項を検討してください。

- **入力検証:** コンテンツ取得に使用するURLが適切にフォーマットされ、想定されるパターンに準拠していることを保証するため、厳格な入力検証メカニズムを実装してください。これには、有効なURLスキームの確認、想定ドメイン名の強制、およびクエリパラメータの検証などが含まれます。
- **ホワइटリスト方式:** アプリケーションがコンテンツを取得する信頼できるドメインまたはソースのホワइटリストを維持してください。これらの信頼できるソースへのリクエストのみを許可し、悪意のあるコンテンツや不正なコンテンツへのアクセスリスクを最小化します。
- **コンテンツ整合性チェック:** 取得したコンテンツの整合性を検証するメカニズムを実装してください。受信したコンテンツの暗号学的ハッシュまたは電子署名を計算し、想定値と比較することで、改変や改竄を検出します。

これらの推奨事項を実装することで、アプリケーションのセキュリティを強化し、ユーザーデータを保護するとともに、URL改竄攻撃に伴うリスクを軽減できます。

以下は、ホワइटリスト検証を実装するサンプルコードです。

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(),
      ),
    );
  }
}
```

```
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()
    }
}
```

```
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
    }
}

```

ハードニング 不安全なネットワーク構成設定

CVSSv3: 9.1

アプリケーションにネットワークセキュリティ構成が指定されておらず、または安全でない設定が適用されています。

Android Network Security Configurationは、アプリケーションのネットワークセキュリティを宣言的に設定する機能です。

Android Network Security Configurationが未設定、または安全でない方法で設定されている場合、アプリケーションはMiTM攻撃に対して脆弱となる可能性があります。

技術的詳細

検出事項 #1

`res/xml/network_security_config.xml` : The `base-config` authorizes cleartext traffic.

推奨事項

Android Network Security Configurationは、アプリケーションのネットワークセキュリティを宣言的に設定するためのXMLファイルです。

アプリケーションへの Network Security Configuration の追加

1- アプリの `res/xml` ディレクトリに新規XMLファイルを作成します。 `network_security_config.xml` など、適切なファイル名を付けます。2- セキュリティ構成を定義します。具体例は以下を参照してください。3- 構成をマニフェストに適用します：

<> XML

```
<application
    android:networkSecurityConfig="@xml/network_security_config"
    <!-- Other attributes -->
    >
    <!-- Other configurations -->
</application>
```

Network Security Configuration の例

- デバッグ専用設定をサポートするカスタム認証局（CA）

<> 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>
```

- 平文通信の宣言的な無効化

<> 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>
```

- 証明書ピンニングキーの宣言的な設定

<> 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>
```

CVSS V3ベクター :

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

攻撃ベクトル:

品 ネットワーク (N) 隣接 (A) ローカル (L) 物理 (P)

攻撃の複雑さ:

低 (L) 高 (H)

必要な権限:

None (N) 低 (L) 高 (H)

ユーザー関与:

None (N) 要 (R)

スコープ:

変更あり (C) 変更なし (U)

機密性:

高 (H) 低 (L) None (N)

完全性:

高 (H) 低 (L) None (N)

可用性:

高 (H) 低 (L) None (N)

参考資料

- [Network Security Configuration \(Android developer\)](#)

ハードニング

デバッグモードが有効です

CVSSv3: 5.4

アプリケーションはデバッグモードが有効な状態でコンパイルされています。

AndroidManifestに定義される `Application` の `android:debuggable` 属性は、アプリケーションがデバッグ可能かどうかを決定します。

デバッグモードにより、攻撃者はアプリケーションのファイルシステムにアクセスし、デバッガーをアタッチして機密データへのアクセスや悪意のあるアクションの実行が可能になります。

以下の手順で、`jdb` を使用したデバッグセッションを開始できます。

- `adb jdwp` を使用して、対象アプリケーションの `PID` を特定します:

```
$adb jdwp
3466
15446
```

- `adb` を使用して通信チャンネルを作成し、`jdb` でアタッチします:

```
$adb forward tcp:7777 jdwp:$(adb shell ps | grep "package-id")
$jdb -attach localhost:7777
```

- アプリケーションのファイルシステムにアクセスします:

```
$adb shell
$run-as package-id
$...悪意のあるアクションを挿入...
```

攻撃者はソースコードへのアクセスなしにアプリケーションをデバッグでき、これを悪用してユーザーの権限で悪意のあるアクションを実行したり、アプリケーションの動作を変更したり、認証情報やセッションCookieなどの機密データにアクセスしたりできます。

技術的詳細

検出事項 #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">
```

推奨事項

AndroidManifestの **Application** タグ内で、属性 **android:debuggable** を **false** に設定し、デバッグモードを無効化してください。

<> XML

<application android:icon="@drawable/icon" android:debuggable="false">

CVSSベクター詳細

CVSS V3ベクター：CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N

攻撃ベクトル:

🌐 ネットワーク (N)

🔌 隣接 (A)

🏠 ローカル (L)

👤 物理 (P)

攻撃の複雑さ:

🔒 低 (L)

🔓 高 (H)

必要な権限:

🔑 None (N)

🔒 低 (L)

🔓 高 (H)

ユーザー関与:

👤 None (N)

👤 要 (R)

スコープ:

🔄 変更あり (C)

📄 変更なし (U)

機密性:

🔒 高 (H)

🔒 低 (L)

🔑 None (N)

完全性:

🔒 高 (H)

🔒 低 (L)

🔑 None (N)

可用性:

🔒 高 (H)

🔒 低 (L)

🟢 None (N)

参考資料

- [DRD10-J Do not release apps that are debuggable \(CERT Secure Coding\)](#)

攻撃対象領域