



RESTRINGIDO

REPORTE DE ANÁLISIS

PREPARADO PARA

Ostorlab Oficial

RIESGO GLOBAL - ALTO

FECHA

2026-10-15 18:09:53

Resumen ejecutivo

La aplicación  **PlayStore** (`owasp.sat.agoat:1.0 (1)`) tiene asignado un **ALTO | 9,1** nivel de riesgo y actualmente presenta **44** vulnerabilidades, **0** de las cuales son de riesgo crítico, **12** de riesgo alto, **16** de riesgo medio y **6** de riesgo bajo. Hay **8** vulnerabilidades potenciales que requieren mayor investigación. Para reforzar la seguridad de la aplicación, se recomienda implementar **2** medidas de endurecimiento.

La superficie de ataque de la aplicación es **Notable**, ya que utiliza **0** permisos, incluyendo **0** permisos peligrosos. La aplicación también expone **0** actividades, **0** servicio, y **0** receptores de difusión. Además, utiliza **0** funcionalidades. Asegúrese de que su superficie de ataque esté justificada y elimine cualquier funcionalidad o exposición innecesaria.

En general, la cobertura de la aplicación se considera **Limitada**. Para mejorar la cobertura, se recomienda realizar un análisis dinámico completo.

Resumen de la aplicación

Plataforma	 Android
Paquete	owasp.sat.agoat
Versión	1.0 (1)
Hash SHA1	<code>ea96f4adc27083696a449b1c6e5cd612a2cf0f5c</code>
Tamaño	2 MB
Fecha	May 19th 2026, 16:40:34

Resumen de hallazgos

Vulnerabilidades agrupadas por nivel de riesgo



Resumen de tickets

Resumen de estados de vulnerabilidades

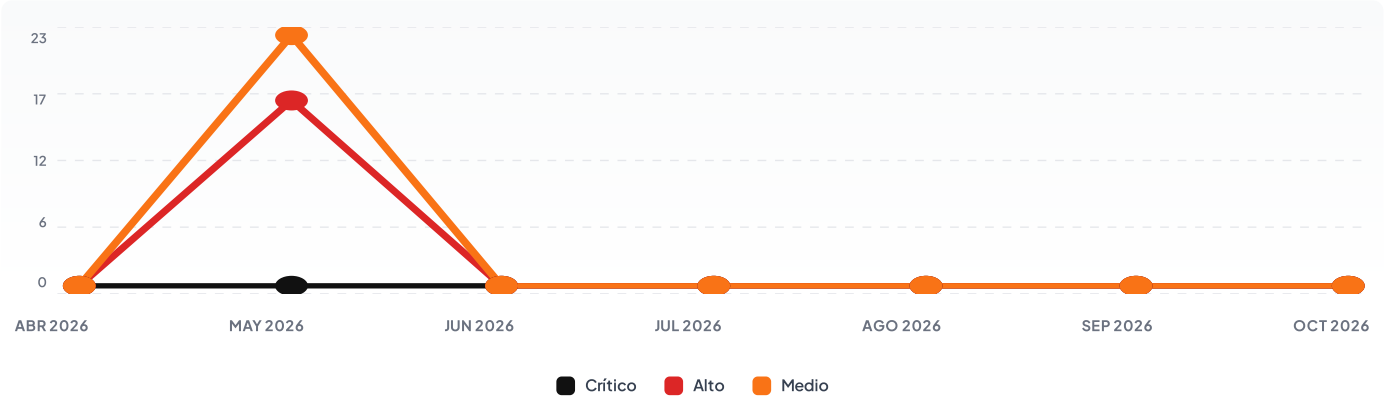


Top 10 de vulnerabilidades por severidad

OTP de baja entropía expuest...		1
Inyección SQL mediante raw...		1
Implicitly Exported Protected ...		1
Servicio exportado permite la ...		1
Credenciales sensibles almac...		1
Credenciales de autenticació...		1
Clickjacking confirmado en a...		1
Credenciales de autenticació...		1
Inyección SQL mediante exec...		1
Firebase Database expuesta ...		1

Tendencias de vulnerabilidades de la aplicación

Tendencias históricas de los últimos 6 meses



Resumen del análisis

Objetivo	Android: owasp.sat.agoat (1.0 (1))
Riesgo global	HIGH
Vulnerabilidades	34 entotal 0 CRÍTICA 12 ALTA 16 MEDIA 6 BAJA 8 POTENCIAL 2 ENDURECIMIENTO

Tabla de hallazgos

Riesgo CVSS	Estado(s)	Etiquetas	Título	Descripción corta
ALTO	Abierto	No hay etiquetas disponibles.	OTP de baja entropía expuesto mediante el portapapeles del sistema y un PRNG débil	<code>ClipboardActivity</code> genera un OTP de cuatro dígitos mediante un PRNG no criptográfico, lo copia al portapapeles global y lo muestra en un Toast, haciendo que el token sea predecible y accesible para cualquier otra aplicación instalada o IME.
ALTO	Abierto	No hay etiquetas disponibles.	Inyección SQL mediante rawQuery sin sanitizar en SQLInjectionActivity	SQLInjectionActivity construye una consulta SELECT concatenando directamente la entrada del usuario de un EditText en rawQuery() con selectionArgs como null, permitiendo la ejecución arbitraria de SQL, la enumeración de esquemas y el bypass de autenticación local.
ALTO	Abierto	No hay etiquetas disponibles.	Implicitly Exported Protected Activity Enables Authentication Bypass (AccessControlViewActivity)	AccessControlViewActivity se exporta implícitamente mediante un intent-filter de deep link y no contiene comprobaciones de autenticación internas, lo que permite que cualquier aplicación de terceros lance la vista protegida sin PIN.
ALTO	Abierto	No hay etiquetas disponibles.	Servicio exportado permite la descarga no autorizada de archivos (DownloadInvoiceService)	<code>DownloadInvoiceService</code> se exporta explícitamente en el manifiesto y descarga archivos al almacenamiento externo público sin realizar comprobaciones de autorización del llamador.
ALTO	Abierto	No hay etiquetas disponibles.	Credenciales sensibles almacenadas en texto plano en SharedPreferences, SQLite y almacenamiento externo	La aplicación persiste nombres de usuario, contraseñas y hashes de PIN en múltiples mecanismos de almacenamiento local sin cifrado, protección KeyStore ni permisos de archivo restrictivos.
ALTO	Abierto	No hay etiquetas disponibles.	Credenciales de autenticación protegidas con hash MD5 sin sal (AccessControlIssueActivity)	La aplicación almacena los PIN como hashes MD5 sin sal en las SharedPreferences internas, lo que permite una recuperación trivial del PIN en texto plano mediante fuerza bruta offline en milisegundos.
ALTO	Abierto	No hay etiquetas disponibles.	Clickjacking confirmado en actividades sensibles en owasp.sat.agoat	Una aplicación maliciosa con SYSTEM_ALERT_WINDOW puede dibujar superposiciones interactivas sobre cuatro actividades sensibles, interceptando los toques del usuario para desencadenar de forma no autorizada descargas de facturas, verificaciones de promociones, generaciones de OTP y envíos de PIN.
ALTO	Abierto	No hay etiquetas disponibles.	Credenciales de autenticación en bruto y consultas SQL filtradas al Logcat de Android	Múltiples actividades escriben incondicionalmente nombres de usuario, contraseñas y sentencias SQL en texto plano en los logs del sistema de Android mediante <code>Log.e</code> , <code>Log.v</code> y <code>System.out.println</code> , lo que permite la exfiltración de credenciales por cualquier aplicación con acceso a los logs o por un atacante con <code>adb</code> .
ALTO	Abierto	No hay etiquetas disponibles.	Inyección SQL mediante execSQL sin sanitizar en InsecureStorageSQLiteActivity	InsecureStorageSQLiteActivity construye una sentencia INSERT concatenando dos valores EditText sin sanitizar en execSQL(), lo que permite la inyección de consultas apiladas, la manipulación arbitraria de registros y el compromiso total de la integridad de la base de datos local.
ALTO	Abierto	Seguridad	Firebase Database expuesta públicamente	exponer públicamente una base de datos Firebase, lo que puede provocar acceso no autorizado, manipulación de datos, divulgación de información, ataques maliciosos y pérdida de reputación.
ALTO	Abierto	Privacidad Seguridad	Vulnerabilidad de primos comunes LOGJAM	Uso de primos Diffie-Hellman ampliamente compartidos que permiten ataques precomputados por estados-nación contra grandes porciones del tráfico de Internet.

Riesgo CVSS	Estado(s)	Etiquetas	Título	Descripción corta
ALTO	Abierto	Seguridad	Vulnerabilidad de inyección SQL en móvil	La vulnerabilidad de inyección SQL en aplicaciones móviles permite a un atacante ejecutar consultas SQL maliciosas en la base de datos de una aplicación móvil, lo que puede derivar en acceso no autorizado, fugas de datos o manipulación de datos.
MEDIO	Abierto	No hay etiquetas disponibles.	Privilegio de administrador determinado por un flag booleano del lado del cliente sin verificación del servidor	BinaryPatchingActivity controla el acceso a la funcionalidad de la interfaz de administración mediante un campo booleano local que nunca es validado por una API de backend, un token criptográfico ni un desafío de reautenticación.
MEDIO	Abierto	No hay etiquetas disponibles.	El Certificate Pinning en TrafficActivity falla silenciosamente debido a un manejo de excepciones demasiado amplio	El CertificatePinner de OkHttp en <code>TrafficActivity.doPinning()</code> se anula porque <code>SSLPeerUnverifiedException</code> es capturada por un bloque catch amplio y suprimida, lo que permite ataques Man-in-the-Middle silenciosos.
MEDIO	Abierto	No hay etiquetas disponibles.	Permisos inseguros de almacenamiento externo filtran credenciales (InsecureStorageSDCardActivity)	<code>InsecureStorageSDCardActivity</code> escribe las credenciales de usuario en un archivo temporal en el almacenamiento externo y establece explícitamente el archivo como world-readable y world-writable.
MEDIO	Abierto	No hay etiquetas disponibles.	Pantallas sensibles que permiten capturas de pantalla no autorizadas por ausencia de FLAG_SECURE	Todas las 25 actividades de owasp.sat.agoat carecen de FLAG_SECURE, lo que permite la captura y grabación sin restricciones de pantallas sensibles, como la introducción del PIN, la entrada de credenciales y la generación de OTP.
MEDIO	Abierto	No hay etiquetas disponibles.	Flags inseguros de backup y debug exponen el sandbox de la aplicación	El manifiesto de la aplicación habilita <code>android:debuggable="true"</code> y <code>android:allowBackup="true"</code> , lo que permite a cualquier usuario con acceso a la depuración USB extraer, modificar y restaurar todos los datos privados de la aplicación o adjuntar un depurador al proceso en ejecución.
MEDIO	Abierto	No hay etiquetas disponibles.	Credenciales de autenticación embebidas en el código fuente de la aplicación	Múltiples secretos hardcoded —incluyendo un nombre de usuario, una contraseña y una clave de autenticación— están embebidos directamente en el código fuente de la aplicación y expuestos en tiempo de ejecución a través de un BroadcastReceiver exportado.
MEDIO	Abierto	Seguridad	Vulnerabilidad criptográfica: Algoritmo de hash débil	El cliente utiliza funciones de hash débiles que presentan debilidades criptográficas conocidas.
MEDIO 9,1	Abierto	Seguridad	Solicitud HTTP en texto plano	La aplicación está accediendo a datos a través de un canal no cifrado, comprometiendo la confidencialidad y la integridad de los datos en tránsito.
MEDIO	Abierto	Privacidad Seguridad	CERTIFICATE_EXPIRED	Un certificado SSL/TLS caducado ya no es confiable para navegadores y clientes, lo que provoca advertencias de seguridad, interrupciones de servicio y vulnerabilidades potenciales, incluidos ataques man-in-the-middle.
MEDIO	Abierto	Seguridad	Configuración insegura de encabezados HTTP	El encabezado de la aplicación está ausente o establece valores de encabezado inseguros o no seguros.
MEDIO	Abierto	Seguridad	Configuración insegura de cabecera HTTP: HTTP Strict Transport Security (HSTS)	Configuración insegura de encabezado HTTP: HTTP Strict Transport Security (HSTS).
MEDIO	Abierto	Seguridad	Cifrados TLS inseguros admitidos	El cliente admite combinaciones de cipher suites con debilidades criptográficas conocidas.
MEDIO	Abierto	Privacidad Seguridad	TLS_FALLBACK_SCSV no soportado	El servidor no admite TLS_FALLBACK_SCSV, lo que lo deja vulnerable a ataques de downgrade de protocolo.

Riesgo CVSS	Estado(s)	Etiquetas	Título	Descripción corta
MEDIO	Abierto	Privacidad Seguridad	Clave criptográfica y algoritmo de firma débiles en certificado SSL/TLS	El certificado SSL/TLS utiliza tanto una clave criptográfica débil como un algoritmo de firma débil, lo cual podría comprometer la seguridad de las conexiones cifradas y permitir la falsificación de certificados u otros ataques criptográficos.
MEDIO	Abierto	Privacidad Seguridad	Abuso de la conexión de red móvil	La aplicación obliga a utilizar la conexión de red móvil.
MEDIO	Abierto	Seguridad	Inyección XML	La inyección XML es una vulnerabilidad que se produce cuando la entrada de usuario no confiable no se valida ni se sanitiza correctamente antes de incorporarse a datos XML. Los atacantes pueden explotar esta vulnerabilidad inyectando contenido XML malicioso, lo que puede provocar acceso no autorizado, manipulación de datos o ataques de denegación de servicio.
BAJO 5,4	Abierto	Seguridad	Atributo usesCleartextTraffic establecido	La aplicación establece el atributo usesCleartextTraffic.
BAJO	Abierto	Seguridad	Uso de versión obsoleta del protocolo TLS/SSL	La aplicación configura protocolos SSL/TLS obsoletos con problemas conocidos de seguridad y privacidad.
BAJO	Abierto	Seguridad	Google Calendar – Exposición	Se ha detectado un Google Calendar de acceso público embebido en el objetivo que podría exponer información sensible, incluyendo detalles de reuniones, nombres de asistentes, horarios de eventos y datos internos de la organización.
BAJO 5,4	Abierto	Privacidad	Atributo hasFragileUserData no definido	La aplicación no establece explícitamente el atributo hasFragileUserData.
BAJO 5,4	Abierto	Seguridad	Versión de API de destino obsoleta	La aplicación establece el atributo targetSdkVersion para permitir el uso de un nivel de API anterior de Android, lo que podría exponer a los usuarios a vulnerabilidades de seguridad.
BAJO	Abierto	Seguridad Malware Exploit	Task Hijacking	La aplicación no protege contra el task hijacking, lo que permite que una aplicación maliciosa en el dispositivo secuestre su tarea en primer plano
POTENCIALMENTE	Abierto	No hay etiquetas disponibles.	Configuración insegura de seguridad de red y falta de protecciones de transporte en ANALYTICS_LIBRARY	El sustituto ANALYTICS_LIBRARY (TrafficActivity) transmite el tráfico analítico mediante un OkHttpClient predeterminado sin pinning, mientras que la Network Security Configuration permite globalmente HTTP en texto claro y confía en CAs instaladas por el usuario. El pipeline carece de cifrado de payload, firma de solicitudes y controles de integridad, lo que permite de forma trivial la interceptación MitM, la reproducción y la manipulación.
POTENCIALMENTE	Abierto	Seguridad	Información secreta almacenada en la aplicación	Contraseñas, tokens u otra información sensible se almacenan en la aplicación.
POTENCIALMENTE 3,9	Abierto	Privacidad	Modo de respaldo activado	La aplicación está habilitando el modo de respaldo.
POTENCIALMENTE	Abierto	Seguridad	Travesía de rutas de archivo	La presencia de un path traversal en la aplicación indica una vulnerabilidad mediante la cual un atacante puede manipular parte de la ruta del archivo. Esta ruta manipulada se utiliza posteriormente en las APIs del sistema de archivos sin una validación o revisión adecuada.
POTENCIALMENTE	Abierto	Seguridad	Almacenamiento inseguro de datos de la aplicación	Almacenamiento inseguro de datos de la aplicación que permite a otras aplicaciones leerlos y/o alterarlos.

Riesgo CVSS	Estado(s)	Etiquetas	Título	Descripción corta
POTENCIALMENTE	Abierto	Seguridad	Source to Sink	Se invoca un método Sink peligroso utilizando parámetros potencialmente controlados por el usuario o un método Source no fiable.
POTENCIALMENTE	Abierto	Seguridad	Manipulación de URL	Un atacante puede controlar la URL que una aplicación utiliza para obtener contenido; esto supone un riesgo de seguridad, ya que puede manipular dicho contenido y dar lugar a la ejecución de código malicioso o al robo de información sensible.
POTENCIALMENTE 3,4	Abierto	Seguridad	Servicios declarados sin permisos	Los servicios declarados no están protegidos con permisos globales.
REFUERZO 9,1	Abierto	Privacidad Seguridad	Configuración de red insegura	La aplicación no especifica una configuración de seguridad de red o establece ajustes inseguros.
REFUERZO 5,4	Abierto	Seguridad	Modo de depuración activado	La aplicación se compiló con el modo de depuración habilitado.

Lista de hallazgos detallados

ALTO OTP de baja entropía expuesto mediante el portapapeles del sistema y un PRNG débil

`ClipboardActivity` genera un OTP de cuatro dígitos mediante un PRNG no criptográfico, lo copia al portapapeles global y lo muestra en un Toast, haciendo que el token sea predecible y accesible para cualquier otra aplicación instalada o IME.

El OTP se extrae de un conjunto de tan solo 9.000 valores posibles mediante `CollectionsKt.shuffled`, que delega en `java.util.Random`. Este generador basado en LCG permite recuperar la semilla a partir de observaciones secuenciales. El token resultante se copia en el portapapeles del sistema sin mecanismo de expiración ni eliminación y se muestra simultáneamente en un Toast. No existe un backend de verificación del lado del servidor, por lo que el OTP es puramente un artefacto del lado del cliente que puede interceptarse mediante lectores de portapapeles o predecirse observando la secuencia del PRNG.

Detalles técnicos

Hallazgo #1

Causa raíz

El desarrollador utilizó un método de conveniencia `shuffle` respaldado por `java.util.Random` en lugar de `java.security.SecureRandom`. El token se colocó en el portapapeles global y se mostró en un Toast de la UI sin ninguna lógica de expiración, validación o anti-replay.

Código vulnerable

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

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

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

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

Evidencia de explotación

[Static Analysis] El análisis estático identificó el espacio de tokens `IntRange(1000, 9999)` y rastreó el PRNG hasta `java.util.Random` a través de la instancia `Random` predeterminada de Kotlin. La llamada a `ClipboardManager.setPrimaryClip` se confirmó en la línea 70 sin llamadas posteriores a `clearPrimaryClip` ni tiempo de expiración. La actividad no contiene lógica de red ni de validación.

Evidencia de validación

Causa raíz

La generación de OTP en `owasp.sat.agoat.ClipboardActivity` depende de `CollectionsKt.shuffled(new IntRange(1000, 9999))`, que delega a través de la biblioteca estándar de Kotlin a `java.util.Collections.shuffle(List)`. Esta API de Java construye una instancia predeterminada de `java.util.Random` con una semilla derivada del tiempo del sistema en lugar de `java.security.SecureRandom`. La actividad expone el token generado a través de dos canales inseguros: el portapapeles global de Android mediante `ClipboardManager.setPrimaryClip` y un Toast de la UI mediante `Toast.makeText`. No existe lógica de expiración, limpieza ni validación del lado del servidor en ninguna parte de la actividad.

Código vulnerable

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

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

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

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

Delegación de la biblioteca estándar de Kotlin a `java.util.Random` (traza OTPCLIP-003):

```
// CollectionsKt_MutableCollectionsJVMKt.java (lines 56-61)
public static final <T> List<T> shuffled(Iterable<? extends T> shuffled) {
    Intrinsic.checkParameterNotNull(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) {
    Intrinsic.checkParameterNotNull(shuffle, "$this$shuffle");
    Intrinsic.checkParameterNotNull(random, "random");
    for (int i = CollectionsKt.getLastIndex(shuffle); i >= 1; i--) {
        int j = random.nextInt(i + 1);
        ...
    }
}
```

Evidencia de explotación

[Static Analysis] La validación estática exhaustiva (OTPClip-001 a OTPClip-003) confirmó lo siguiente:

- Ubicación exacta del código fuente:** `owasp/sat/agoat/ClipboardActivity.java` en `/workspace/source_code_analysis_ffd42589f4eb4cf4a51427480ebe2d69/base.apk/sources/owasp/sat/agoat/ClipboardActivity.java`, 79 líneas en total.
- Generación de OTP confirmada en la línea 63:** `CollectionsKt.shuffled(new IntRange(1000, 9999))` produce exactamente 9.000 valores posibles (1000–9999 incluidos), generando aproximadamente **13,14 bits de entropía** (`log2(9000)`).
- Cadena PRNG completamente trazada:** La sobrecarga sin argumentos de `CollectionsKt.shuffled` llama a `Collections.shuffle(List)`, que instancia un `java.util.Random` predeterminado con una semilla derivada del tiempo. Los archivos `CollectionsKt_MutableCollectionsJVMKt.java` (líneas 56–61) y `CollectionsKt_MutableCollectionsKt.java` (líneas 296–305) demuestran la cadena de delegación. No existe importación ni uso de `SecureRandom` en `ClipboardActivity.java`.
- Exposición en el portapapeles confirmada en las líneas 68–70:** Se llama a `ClipboardManager.setPrimaryClip(clip)` con `ClipData.newPlainText("CC Card", String.valueOf(otp))`. No se invoca `clearPrimaryClip()`, `Handler.postDelayed()` ni se realiza limpieza del ciclo de vida posteriormente para eliminar el contenido del portapapeles.
- Exposición mediante Toast confirmada en la línea 71:** `Toast.makeText(ClipboardActivity.this, "OTP Generated and Copied: " + otp, 1).show()` muestra el entero OTP en bruto en pantalla, haciéndolo accesible a servicios de accesibilidad y grabación de pantalla.
- Sin lógica de validación:** Las búsquedas en toda la actividad y en las importaciones confirman la ausencia de APIs de E/S de red (`URLConnection`, `OkHttp`, `Retrofit`, `Volley`) y de palabras clave de validación. El OTP es puramente del lado del cliente sin verificación del servidor, expiración ni protección anti-replay.
- Restricciones de la superficie de ataque:** `AndroidManifest.xml` (línea 39) muestra que la actividad no tiene `intent-filters` ni `android:exported="true"`, lo que significa que las aplicaciones externas no pueden disparar la generación de OTP directamente. Sin embargo, una vez que un usuario interactúa con la UI de la aplicación, el OTP está inmediatamente disponible para cualquier aplicación que lea el portapapeles o para cualquier servicio de accesibilidad.
- Sin PRNG seguro alternativo:** Las importaciones de `ClipboardActivity.java` incluyen únicamente colecciones de `java.util.*` y tipos de la biblioteca estándar de Kotlin; `java.security.SecureRandom`, `kotlin.random.Random` y `Math.random()` están todos ausentes.

Recomendación

Corrección inmediata

Reemplace el generador inseguro por `SecureRandom` y aumente la longitud del OTP a al menos seis dígitos:

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

Evite copiar el OTP en el portapapeles del sistema; si es inevitable, límpielo tras un breve intervalo mediante un `Handler`.

Corrección permanente

Implemente un endpoint en el servidor para la generación y verificación de OTP. El cliente nunca debe generar ni validar tokens de recuperación de forma local. Exija uso único, una vigencia limitada a aproximadamente cinco minutos y una limitación de tasa a nivel de cuenta.

Verificación

Descompile el APK parchado y confirme que `CollectionsKt.shuffled` y `ClipboardManager.setPrimaryClip` ya no estén presentes en el flujo de OTP. Busque `java.util.Random` en la base de código para asegurarse de que haya sido reemplazado por `java.security.SecureRandom`.

ALTO Inyección SQL mediante rawQuery sin sanitizar en SQLInjectionActivity

SQLInjectionActivity construye una consulta SELECT concatenando directamente la entrada del usuario de un EditText en rawQuery() con selectionArgs como null, permitiendo la ejecución arbitraria de SQL, la enumeración de esquemas y el bypass de autenticación local.

La actividad SQLInjectionActivity realiza una consulta de solo lectura de un nombre de usuario en la base de datos SQLite local `agoat`. Dado que la cadena de consulta se construye mediante concatenación con `StringBuilder` a partir de una entrada `EditText` sin sanitizar y se pasa a `rawQuery(qry, null)`, un atacante puede inyectar sintaxis SQL arbitraria en la cláusula `WHERE`. Una inyección exitosa altera la semántica de la consulta para devolver todas las filas incondicionalmente, lo que la aplicación interpreta como un inicio de sesión válido debido a la lógica `cursor.getCount() > 0`. Esto expone todos los nombres de usuario y contraseñas en texto plano almacenados en la tabla `users` y permite extraer el esquema mediante `UNION SELECT`. Además, la cadena de consulta completa se registra en Logcat a nivel `VERBOSE`, por lo que los payloads inyectados pueden persistir en los logs del dispositivo.

Detalles técnicos

Hallazgo #1

Causa raíz

En la línea 69 de `owasp/sat/agoat/SQLInjectionActivity.java`, la aplicación construye dinámicamente una sentencia SQL `SELECT` concatenando directamente la entrada del usuario sin sanitizar en la cadena de la consulta. El método `rawQuery()` se invoca con `null` como parámetro `selectionArgs`, eliminando cualquier posibilidad de parametrización en SQLite.

Código vulnerable

El patrón de construcción de consultas vulnerable identificado durante el análisis estático es el siguiente:

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

La variable `username` proviene de `R.id.userName`, un widget `EditText` que acepta entrada de texto libre sin validación, lista de permitidos ni sanitización.

Evidencia de explotación

[Simulado] Dado que no se dispuso de un emulador de Android ni de herramientas `adb`, la explotación se validó contra una base de datos SQLite simulada localmente que replica el esquema exacto de `agoat` (tabla `users` con `ID INTEGER PRIMARY KEY AUTOINCREMENT`, `username VARCHAR`, `password VARCHAR`). La consulta reconstruida se ejecutó con entradas controladas por el atacante:

- **Payload de ruptura de sintaxis:** la entrada `'` inyectada en la consulta produjo un error `unrecognized token`, lo que confirma que la entrada escapa del contexto del literal de cadena.
- **Payload de bypass booleano:** la entrada `' OR '1'='1' --` transformó la consulta en `SELECT * FROM users WHERE username=' ' OR '1'='1' --`. La simulación devolvió **todas las filas** de la tabla `users`, demostrando una recuperación incondicional de filas.
- **Extracción de esquema basada en UNION:** la entrada `' UNION SELECT null, sql, null FROM sqlite_master --` volcó con éxito las sentencias CREATE de `sqlite_master`, confirmando la enumeración del esquema.

La lógica de la aplicación verifica `cursor.getCount() > 0` para determinar el éxito. Dado que el bypass booleano fuerza al cursor a contener filas, la aplicación ingresa a la rama de éxito y muestra todas las credenciales coincidentes en un Toast.

Evidencia de validación

Estado de validación

Esta ejecución de validación no ejecutó ninguna tarea de pruebas dinámicas contra el APK objetivo. No se generó evidencia de explotación nueva más allá del análisis estático original.

Causa raíz

[Análisis estático] En la línea 69 del archivo decompilado `owasp/sat/agoat/SQLInjectionActivity.java`, se construye dinámicamente una sentencia SQL `SELECT` concatenando directamente la entrada del usuario sin sanitizar en la cadena de la consulta. El método `rawQuery()` se invoca con `null` como parámetro `selectionArgs`, eliminando cualquier posibilidad de parametrización en SQLite.

Código vulnerable

[Análisis estático] Se identificó el siguiente patrón inseguro de construcción de consultas en el APK decompilado:

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

El valor de `username` proviene de `R.id.userName`, un widget `EditText` que acepta entrada de texto libre sin validación, lista de permitidos ni sanitización.

Evidencia de explotación

[Análisis estático] El patrón de código vulnerable permite que la entrada controlada por el atacante escape del contexto del literal de cadena entre comillas simples en la cláusula `WHERE`, alterando la semántica de la consulta. [Simulado] Una simulación previa contra un esquema replicado de `aGoat` demostró que:

- La entrada `'` rompe la sintaxis SQL, produciendo un error `unrecognized token`
- La entrada `' OR '1'='1' --` modifica la consulta para devolver todas las filas de la tabla `users` incondicionalmente
- La entrada `' UNION SELECT null, sql, null FROM sqlite_master --` extrae información del esquema desde `sqlite_master`. La lógica de la aplicación depende de `cursor.getCount() > 0` para determinar el éxito de la autenticación, por lo que devolver todas las filas anula esta comprobación y expone las credenciales en texto plano almacenadas en la tabla `users`.

Recomendación

Mitigación inmediata

Eliminar `android:debuggable="true"` de `AndroidManifest.xml` y deshabilitar el registro detallado de consultas SQL que imprime la entrada del usuario en Logcat.

Corrección permanente

Reemplazar la concatenación dinámica de cadenas en `SQLInjectionActivity.java` por una llamada parametrizada a `rawQuery()` que utilice marcadores de posición `?` y un array `selectionArgs`:

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

Esto garantiza que la entrada del usuario se trate estrictamente como un valor literal de cadena y no pueda alterar la estructura de la consulta.

Verificación

- Reejecutar el análisis estático para confirmar que no existan llamadas a `rawQuery()` con `selectionArgs` en `null` donde haya entrada del usuario.
- Proporcionar payloads como `' OR '1'='1' --` y verificar que la consulta devuelva cero filas en lugar de todas las filas.
- Inspeccionar la salida de Logcat para confirmar que ya no se escriban cadenas SQL completas que contengan entrada del usuario a nivel `VERBOSE`.

ALTO Implicitly Exported Protected Activity Enables Authentication Bypass (AccessControlViewActivity)

`AccessControlViewActivity` se exporta implícitamente mediante un intent-filter de deep link y no contiene comprobaciones de autenticación internas, lo que permite que cualquier aplicación de terceros lance la vista protegida sin PIN.

Esta vulnerabilidad anula por completo el mecanismo de verificación por PIN implementado en `AccessControlIssue1Activity`. Un atacante con acceso físico ADB o una aplicación maliciosa co-instalada puede acceder al panel protegido de descarga de facturas sin credenciales. En un escenario de producción, esto elude los controles principales de confidencialidad e integridad que protegen la funcionalidad sensible del usuario.

Detalles técnicos

Hallazgo #1

Causa raíz

`AccessControlViewActivity` se declara en `AndroidManifest.xml` con un `<intent-filter>` que contiene un esquema URL personalizado (`androgoat://vulnapp`), pero carece del atributo explícito `android:exported="false"`. Dado que `targetSdkVersion="26"` (comportamiento previo a Android 12), la presencia de un intent-filter exporta implícitamente la Activity. El análisis estático del código fuente de la Activity confirma la ausencia total de lógica de autenticación dentro de `onCreate()`.

Código vulnerable

AndroidManifest.xml (declaración de exportación implícita):

```
<activity android:label="@string/activity" android:name="owasp.sat.agoat.AccessControlViewActivity">
  <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 (líneas 45–54) — sin control de autenticación:

```

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();
        }
    });
}

```

Evidencia de explotación

[Análisis estático] El manifiesto y el código fuente confirman que cualquier aplicación de terceros o comando `adb` puede iniciar esta Activity directamente. El flujo normal de la aplicación exige un PIN verificado a través de `AccessControlIssue1Activity`, pero dicho control se elude por completo al invocar el deep link directamente.

Comando de prueba de concepto:

```

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

```

Confirmación esperada de la pila de actividades:

```

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

```

Evidencia de validación

Causa raíz

La validación estática del APK decompilado confirmó que `AccessControl1ViewActivity` se declara en `base.apk/AndroidManifest.xml` con un `<intent-filter>` que expone el esquema de deep link personalizado `androgoat://vulnapp`. La etiqueta `<activity>` carece tanto de `android:exported="false"` como de cualquier atributo `android:permission`. La auditoría de `base.apk/apktool.yml` verificó `targetSdkVersion: 26`; bajo el comportamiento previo a Android 12, esta omisión provoca la exportación implícita del componente.

Código vulnerable

AndroidManifest.xml (declaración exacta validada):

```

<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`, líneas 38–48) — renderiza la interfaz protegida incondicionalmente:

```

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_access_control1_view);
    Button downloadInvoice = (Button) findViewById(R.id.download);
    downloadInvoice.setOnClickListener(new View.OnClickListener() {
        @Override

```

```

        public final void onClick(View it) {
            AccessControl1ViewActivity.this.startService();
        }
    });
}

```

La clase extiende `android.support.v7.app.AppCompatActivity`; ninguna clase base personalizada impone autenticación. El archivo completo de 54 líneas no contiene llamadas a `getSharedPreferences`, comparaciones MD5, solicitudes repetidas de PIN ni validación de sesión.

Evidencia comparativa — aislamiento del control de PIN: Toda la lógica de autenticación reside exclusivamente en `AccessControlIssue1Activity.java` (~líneas 88–124), que almacena un hash MD5 del PIN en `pinDetails` de `SharedPreferences` y solo inicia `AccessControl1ViewActivity` tras la verificación exitosa:

```

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

```

Evidencia de explotación

[Análisis estático] El manifiesto y el código fuente confirman que cualquier aplicación de terceros o shell de ADB puede iniciar la Activity directamente. El intent del deep link resuelve al componente exportado implícitamente:

```

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

```

La validación descubrió un **segundo sink no autenticado**: `DownloadInvoiceService` está exportado explícitamente (`android:exported="true"`, sin permiso) y puede invocarse directamente, omitiendo incluso el inicio de la Activity:

```

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

```

En ambas rutas, la aplicación alcanza la funcionalidad de descarga de facturas sin presentar jamás el desafío del PIN.

Recomendación

Solución inmediata

Agregue `android:exported="false"` a `AccessControl1ViewActivity` en `AndroidManifest.xml`, o elimine el `<intent-filter>` si no se requiere el deep link.

```

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

```

Corrección permanente

Si debe conservarse la funcionalidad de deep link, implemente una verificación de autenticación en tiempo de ejecución dentro de

`AccessControl1ViewActivity.onCreate()` antes de inflar la vista protegida:

```

@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
}

```

Verificación

1. Intente lanzar la actividad mediante `adb shell am start -d "androgoat://vulnapp"`; verifique que el lanzamiento se bloquee o que la aplicación no acceda a la vista protegida.
2. Inspeccione el manifiesto para confirmar que `android:exported="false"` esté presente.

3. Ejecute `drozer` o `am` para confirmar que la actividad ya no sea accesible desde un UID de terceros.

ALTO Servicio exportado permite la descarga no autorizada de archivos (DownloadInvoiceService)

`DownloadInvoiceService` se exporta explícitamente en el manifiesto y descarga archivos al almacenamiento externo público sin realizar comprobaciones de autorización del llamador.

Esta vulnerabilidad convierte el Service exportado en un gateway de descarga de archivos sin autenticación. Dado que el destino es el almacenamiento externo público, cualquier aplicación con el permiso `READ_EXTERNAL_STORAGE` puede leer el contenido descargado. En un entorno de producción donde las facturas contienen PII o datos financieros, esto constituiría una brecha directa de confidencialidad.

Detalles técnicos

Hallazgo #1

Causa raíz

`DownloadInvoiceService` se declara con `android:exported="true"` y su método `onStartCommand()` invoca `downloadFile()` de inmediato sin validar la identidad del llamante, sus permisos ni su estado de autenticación. Esto permite que cualquier aplicación de terceros desencadene descargas de red y escriba archivos en el almacenamiento externo público.

Código vulnerable

AndroidManifest.xml:

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

DownloadInvoiceService.java (líneas 48-83):

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

Evidencia de explotación

[Análisis estático] El manifiesto exporta explícitamente el servicio y el código fuente confirma que no existe lógica de autorización antes de encolar la descarga. Un atacante puede iniciar este servicio directamente sin interactuar jamás con `AccessControl1ViewActivity` ni proporcionar un PIN.

Comando de prueba de concepto:

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

Impacto: El archivo descargado se escribe en `/sdcard/Download/<timestamp>`, un directorio público. El nombre de archivo basado en timestamp es predecible, y `allowScanningByMediaScanner()` hace que el archivo sea visible para la biblioteca multimedia del dispositivo. Las invocaciones repetidas pueden agotar el almacenamiento o el ancho de banda (DoS local).

Evidencia de validación

Causa raíz

`DownloadInvoiceService` está explícitamente exportado en `AndroidManifest.xml` con `android:exported="true"` y sin el atributo `android:permission`. El método del ciclo de vida `onStartCommand()` (líneas 36–42) realiza una invocación directa de `downloadFile()` sin inspeccionar el paquete llamante, los permisos ni el contenido del Intent. Un escaneo regex a nivel de proyecto en busca de APIs de validación del llamante (`checkCallingPermission`, `Binder.getCallingUid`, `getCallingPackage`, etc.) no arrojó coincidencias en el paquete `owasp/sat/agoat`, lo que confirma que no existe lógica de autorización oculta en superclases ni en código auxiliar.

Código vulnerable

AndroidManifest.xml:

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

DownloadInvoiceService.java (líneas 36–42):

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

DownloadInvoiceService.java (líneas 64–80):

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

res/xml/network_security_config.xml:

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

Evidencia de explotación

[Análisis estático] La auditoría del manifiesto confirmó que el servicio se exporta sin permisos, y el código fuente decompilado coincide exactamente con el código vulnerable reportado. La validación introdujo los siguientes hallazgos adicionales:

- **Parámetro Intent sin uso:** `onStartCommand()` recibe `Intent intent` pero nunca lo referencia, lo que demuestra que no existe validación de entrada ni control de acciones.
- **Ausencia de APIs de autorización:** Un escaneo a nivel de proyecto confirmó que no se utiliza ninguna de las APIs `checkCallingPermission`, `checkCallingOrSelfPermission`, `Binder.getCallingUid`, `Binder.getCallingPid` o `getCallingPackage`.
- **Nombre de archivo predecible:** El nombre de archivo es `String.valueOf(System.currentTimeMillis())`, lo que genera aproximadamente 120.000 valores posibles dentro de una ventana de ± 1 minuto; trivialmente explotable por fuerza bruta.
- **Exposición al escáner multimedia:** `allowScanningByMediaScanner()` fuerza la indexación en `MediaStore.Downloads.EXTERNAL_CONTENT_URI`, permitiendo su descubrimiento sin acceso directo al sistema de archivos.

- **Canal lateral de temporización:** `setNotificationVisibility(1)` se mapea a `VISIBILITY_VISIBLE_NOTIFY_COMPLETED`, mostrando una notificación del sistema que filtra el tiempo exacto de finalización de la descarga a la pantalla de bloqueo y a cualquier aplicación con acceso al servicio de escucha de notificaciones.
- **Confianza de red agravada:** `network_security_config.xml` confía en CAs instaladas por el usuario (`<certificates src="user" />`) y permite tráfico en claro, facilitando la interceptación MITM o la sustitución de payload.

Cualquier aplicación de terceros o shell de ADB puede iniciar el servicio:

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

El archivo descargado se almacena en `/sdcard/Download/<epoch_ms>` (almacenamiento externo público) y puede ser leído por cualquier aplicación que disponga del permiso `READ_EXTERNAL_STORAGE`.

Recomendación

Corrección inmediata

Establezca `android:exported="false"` en el servicio `DownloadInvoiceService` dentro de `AndroidManifest.xml`:

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

Corrección permanente

Si se requiere invocación externa, proteja el servicio con un permiso personalizado:

```
<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" />
```

Además, agregue una comprobación de autenticación dentro de `onStartCommand()` antes de encolar la descarga y cambie el destino al almacenamiento interno privado de la aplicación o a un directorio de caché cifrado en lugar de `Environment.DIRECTORY_DOWNLOADS`.

Verificación

1. Intente ejecutar `adb shell am startservice -n owasp.sat.agoat/.DownloadInvoiceService` desde un shell con un UID diferente y confirme que la llamada se rechaza (`SecurityException`).
2. Inspeccione el manifiesto para confirmar la presencia de `exported="false"` o de un permiso requerido.
3. Verifique que los archivos descargados ya no se escriban en el almacenamiento externo público.

ALTO Credenciales sensibles almacenadas en texto plano en SharedPreferences, SQLite y almacenamiento externo

La aplicación persiste nombres de usuario, contraseñas y hashes de PIN en múltiples mecanismos de almacenamiento local sin cifrado, protección KeyStore ni permisos de archivo restrictivos.

El análisis del código fuente revela que `InsecureStorageSharedPrefs` escribe credenciales en `SharedPreferences("users")` en claro, `InsecureStorageSQLiteActivity` las almacena en una base de datos SQLite sin cifrar llamada `aGoat`, e `InsecureStorageTempActivity` las escribe en archivos temporales legibles por cualquier aplicación tanto en almacenamiento interno como externo. Como la aplicación también declara `android:allowBackup="true"`, estos archivos pueden extraerse mediante `adb backup` o por cualquier aplicación con el permiso `READ_EXTERNAL_STORAGE`. Esto expone información equivalente a la identidad del usuario ante cualquier atacante local o aplicación maliciosa, permitiendo la suplantación completa de la cuenta sin necesidad de eludir un PIN.

Detalles técnicos

Hallazgo #1

Causa raíz

Varias Activities del ejercicio de almacenamiento escriben datos sensibles del usuario (nombres de usuario, contraseñas, hashes de PIN) en almacenamiento persistente usando formatos de texto plano y permisos de archivo débiles. No se utiliza `Cipher` , `KeyStore` ni `EncryptedSharedPreferences` en ningún punto de la base de código.

Código vulnerable

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

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

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

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

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

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

Evidencia de explotación

[Análisis estático] La revisión del código fuente confirmó la persistencia en texto plano en `SharedPreferences`, `SQLite` y archivos temporales. El manifiesto también declara `android:allowBackup="true"` , permitiendo la extracción mediante `adb backup` . `InsecureStorageSDCardActivity` escribe credenciales en `Environment.getExternalStorageDirectory()` , accesible para cualquier aplicación que disponga del permiso `READ_EXTERNAL_STORAGE` .

Evidencia de validación

Causa raíz

Varias Activities del ejercicio de almacenamiento en `owasp.sat.agoat` escriben credenciales sensibles del usuario (pares de nombre de usuario y contraseña) en almacenamiento persistente sin cifrado, consultas parametrizadas ni logging seguro. El `AndroidManifest.xml` debilita aún más el sandbox al habilitar copias de seguridad completas y la depuración en tiempo de ejecución.

Código vulnerable

`InsecureStorageSharedPrefs.java` (líneas 49–56 descompiladas):

```
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` (líneas 33–38 descompiladas):

```
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` (líneas 62–73 descompiladas):

```

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` (línea 86 descompilada) — fuga en el log del sistema:

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

`AndroidManifest.xml` (línea 5):

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

Evidencia de explotación

[Análisis estático] La descompilación del APK y la revisión del código fuente confirmaron los siguientes hallazgos validados:

1. **Extracción de SharedPreferences en texto plano:** `InsecureStorageSharedPrefs` escribe el nombre de usuario y la contraseña en `/data/data/owasp.sat.agoat/shared_prefs/users.xml` en XML de texto plano usando las claves `"username"` y `"password"`.
2. **Base de datos SQLite sin cifrar:** `InsecureStorageSQLiteActivity` crea `/data/data/owasp.sat.agoat/databases/aGoat` con el siguiente esquema:

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

Las credenciales se insertan mediante concatenación directa con `StringBuilder` sin parametrización, lo que confirma tanto el almacenamiento en texto plano como la inyección SQL.

3. **Fuga de credenciales a través del log del sistema:** En la línea 86, la consulta SQL completa —incluida la contraseña en texto plano— se escribe en el log del sistema de Android mediante `Log.v("Query", qry)`. Cualquier aplicación con el permiso `READ_LOGS` o una sesión de `logcat` puede capturar las credenciales sin necesidad de acceder al sistema de archivos.
4. **Vector de extracción mediante adb backup:** El manifiesto en la línea 5 declara `android:allowBackup="true"`, permitiendo que un atacante con acceso físico por USB ejecute:

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

El archivo resultante contiene el sandbox completo de la aplicación, incluyendo `shared_prefs/users.xml` y `databases/aGoat`, sin requerir root ni omitir el PIN.

5. **Extracción en tiempo de ejecución mediante depuración:** En la misma línea del manifiesto se declara `android:debuggable="true"`, habilitando la depuración JDWP. Un atacante puede conectar `jdb` o Android Studio para volcar la memoria, establecer puntos de interrupción en `SharedPreferences.Editor.putString()` o `SQLiteDatabase.execSQL()`, e interceptar las credenciales en tiempo de ejecución.
6. **Componentes exportados que expanden la superficie de ataque:** La auditoría del manifiesto identificó dos componentes explícitamente exportados sin protecciones de permisos:
 - Receptor `owasp.sat.agoat.ShowDataReceiver` (línea 27)
 - Servicio `owasp.sat.agoat.DownloadInvoiceService` (línea 37)
7. **Inyección SQL en INSERT:** La concatenación con `StringBuilder` en `InsecureStorageSQLiteActivity` permite que una entrada malformada altere la estructura SQL. Por ejemplo, un nombre de usuario como `admin', 'x'); DROP TABLE users; --` destruiría la base de datos o modificaría el comportamiento de la consulta.

[Limitación de la evidencia] La ejecución de la validación no incluyó tareas dinámicas para `InsecureStorageTempActivity` e `InsecureStorageSDCardActivity`; sus comportamientos de almacenamiento en texto plano reportados siguen basándose en el análisis estático original.

Recomendación

Inmediato

Deshabilite las copias de seguridad estableciendo `android:allowBackup="false"` en el manifiesto de la aplicación. Cifre los datos locales sensibles existentes mediante `EncryptedSharedPreferences` y `EncryptedFile` de AndroidX, o SQLCipher para el cifrado de bases de datos.

Corrección permanente

Migre todo el almacenamiento de credenciales a `EncryptedSharedPreferences` respaldado por el `KeyStore` de Android:

```
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
);
```

Nunca escriba credenciales en almacenamiento externo ni en archivos temporales legibles por cualquier usuario.

Verificación

Busque en el código fuente descompilado usos de `SharedPreferences`, `SQLiteDatabase` y `FileOutputStream`. Asegúrese de que no se pasen cadenas de nombres de usuario o contraseñas en texto plano a estas APIs. Intente realizar un `adb backup` y verifique que los archivos de credenciales sean ilegibles.

ALTO Credenciales de autenticación protegidas con hash MD5 sin sal (AccessControllssue1Activity)

La aplicación almacena los PIN como hashes MD5 sin sal en las `SharedPreferences` internas, lo que permite una recuperación trivial del PIN en texto plano mediante fuerza bruta offline en milisegundos.

Como el hash no utiliza sal y es de una sola iteración, un atacante que extraiga `pinDetails.xml` mediante `adb backup` (`allowBackup="true"`) o `run-as` (`debuggable="true"`) puede realizar un ataque de diccionario offline contra el espacio de claves limitado del PIN. El PIN recuperado permite entonces el acceso no autorizado a través del flujo de verificación legítimo, o puede combinarse con el bypass de deep-link para escalar el acceso.

Detalles técnicos

Hallazgo #1

Causa raíz

`AccessControllssue1Activity.hashPIN()` utiliza `MessageDigest.getInstance("MD5")` sin salt, recuento de iteraciones ni mecanismo de key-stretching. El digest resultante se escribe en `SharedPreferences("pinDetails", 0)` bajo la clave `"pin"`. Un PIN numérico de 4 a 6 dígitos cuyo hash se calcula mediante MD5 sin salt permite una búsqueda exhaustiva en menos de un segundo.

Código vulnerable

AccessControllssue1Activity.java (líneas 111–131):

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

Uso de SharedPreferences (líneas 85–89):

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

Evidencia de explotación

[Static Analysis] La auditoría del código fuente confirmó que la rutina de hashing carece de salt y utiliza el algoritmo MD5, criptográficamente roto. Se ejecutó un script de fuerza bruta como prueba de concepto contra un hash generado con el mismo algoritmo para cuantificar la debilidad:

```
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
```

Salida real:

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

El hash recalculado `hashlib.md5("7391".encode('utf-8')).hexdigest()` reproduce exactamente el digest objetivo, confirmando que cualquier atacante que obtenga el hash almacenado puede recuperar el PIN en texto plano de forma instantánea.

Evidencia de validación

Causa raíz

`AccessControlIssue1Activity.hashPIN()` invoca `MessageDigest.getInstance("MD5")` sin salt, recuento de iteraciones ni mecanismo de key-stretching. El digest resultante se escribe en `SharedPreferences("pinDetails", 0)` bajo la clave `"pin"`. La validación reveló que ni `activity_access_control_issue1.xml` ni `activity_access_verify_pin.xml` aplican `android:inputType`, `android:digits`, `android:maxLength` ni validación programática a los campos de PIN, lo que significa que la aplicación no impone restricciones de longitud ni de conjunto de caracteres en la entrada. El `AndroidManifest.xml` declara `android:allowBackup="true"` y `android:debuggable="true"`, permitiendo a un atacante extraer `pinDetails.xml` mediante `adb backup` o `run-as` sin acceso root.

Código vulnerable

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

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

```
    return md;
}
```

Sink de escritura en SharedPreferences — `createPIN()` (líneas 76–82):

```
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();
}
```

Recuperación y comparación de SharedPreferences — `isPinCorrect()` (líneas 85–88):

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

Fallo de null-safety: si la clave `"pin"` está ausente mientras `"pinSet"` permanece `true`, `isPinCorrect()` falla con una `NullPointerException` porque `.equals()` se invoca sobre una referencia nula.

Componente exportado para bypass — `AndroidManifest.xml` (líneas 20–24):

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

Evidencia de explotación

[Static Analysis] La auditoría del manifiesto confirmó que ambos requisitos previos para la extracción están activos:

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

La copia de seguridad completa predeterminada de Android incluye todos los archivos SharedPreferences en `/data/data/owasp.sat.agoat/shared_prefs/`, por lo que `pinDetails.xml` se captura en cualquier archivo de `adb backup`.

[Static Analysis] Un script de fuerza bruta como prueba de concepto ejecutado contra un hash generado con el mismo algoritmo vulnerable recuperó el PIN en texto plano de forma instantánea:

```
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
```

Salida real:

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

El hash recalculado reproduce exactamente el digest objetivo, confirmando que un atacante que obtenga el hash almacenado puede recuperar el PIN sin conexión en milisegundos.

[Static Analysis] `AccessControl1ViewActivity` se exporta mediante el esquema de deep-link `androgoat://vulnapp`, proporcionando un bypass directo:

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

Esto permite a un atacante acceder a la pantalla protegida sin conocer el PIN. Combinado con la extracción del hash, el PIN recuperado puede, como alternativa, introducirse a través del flujo legítimo de verificación de `isPinCorrect()`.

Recomendación

Corrección inmediata

Implemente limitación de tasa y bloqueo de cuentas en `AccessControlIssue1Activity.verifyPINView()` para ralentizar los ataques de adivinación en línea mientras se reconstruye la implementación del hash.

Corrección permanente

Reemplace MD5 por una función moderna de hash de contraseñas como bcrypt, scrypt o Argon2id. Utilice una sal aleatoria criptográficamente segura por usuario y un factor de trabajo alto.

Ejemplo (bcrypt con JBCrypt):

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

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

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

Verificación

1. Inspeccione `AccessControlIssue1Activity.java` y confirme que se ha eliminado `MessageDigest.getInstance("MD5")`.
2. Verifique que `hashPIN()` devuelva una cadena bcrypt/Argon2id que contenga una sal aleatoria y un factor de costo.
3. Confirme que `isPinCorrect()` utilice el verificador de tiempo constante de la biblioteca (p. ej., `BCrypt.checkpw`).
4. Asegúrese de que `pinDetails.xml` ya no almacene cadenas hexadecimales MD5 de 32 caracteres.

ALTO Clickjacking confirmado en actividades sensibles en owasp.sat.agoat

Una aplicación maliciosa con SYSTEM_ALERT_WINDOW puede dibujar superposiciones interactivas sobre cuatro actividades sensibles, interceptando los toques del usuario para desencadenar de forma no autorizada descargas de facturas, verificaciones de promociones, generaciones de OTP y envíos de PIN.

Pruebas dinámicas con una aplicación atacante complementaria confirmaron que los eventos táctiles se propagan desde ventanas superpuestas a botones sensibles de `AccessControl1ViewActivity`, `HardCodeActivity`, `ClipboardActivity` y `AccessControlIssue1Activity`, desencadenando acciones privilegiadas sin consentimiento informado. La ausencia total de `android:filterTouchesWhenObscured` y de filtrado de toques en tiempo de ejecución en las 25 actividades deja todos los elementos interactivos vulnerables a tapjacking. Además, `AccessControl1ViewActivity` se exporta mediante el deep link `androgoat://vulnapp`, permitiendo a un atacante navegar directamente a una pantalla post-autenticación y ejecutar clickjacking sobre ella sin verificación previa del PIN.

Detalles técnicos

Hallazgo #1

Causa raíz

La aplicación carece por completo de defensas contra clickjacking en sus 25 actividades. Ninguna vista interactiva declara

`android:filterTouchesWhenObscured="true"` y ninguna actividad implementa `onFilterTouchEventForSecurity()` con validación de `MotionEvent.FLAG_WINDOW_IS_OBSCURED`. Una aplicación maliciosa a la que se haya concedido `android.permission.SYSTEM_ALERT_WINDOW` puede crear ventanas `TYPE_APPLICATION_OVERLAY` que interceptan y propagan eventos táctiles a las vistas sensibles subyacentes.

Código vulnerable

[Análisis estático] Una búsqueda exhaustiva con grep en el código fuente decompilado no arrojó coincidencias para flags de protección de ventana ni filtrado táctil:

```
# 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
```

El `AndroidManifest.xml` expone `AccessControl1ViewActivity` a través de un esquema URL personalizado, permitiendo la navegación directa a una acción posterior a la autenticación sin verificación de PIN:

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

Evidencia de explotación

Se desplegó en el emulador una aplicación de overlay del atacante (`com.attacker.overlay`) con permiso `SYSTEM_ALERT_WINDOW` y ventanas `TYPE_APPLICATION_OVERLAY` . Las pruebas dinámicas confirmaron clickjacking exitoso en cuatro pantallas sensibles; la aplicación atacante registró `CLICKJACKED!` en cada toque interceptado y la aplicación objetivo ejecutó la acción subyacente:

Actividad objetivo	Acción sensible	Coordenadas del overlay	Resultado
<code>AccessControl1ViewActivity</code>	Descargar factura	x=540,y=393	CLICKJACKED confirmado — acción del botón ejecutada
<code>HardCodeActivity</code>	Verificar código promocional	x=540,y=935	CLICKJACKED confirmado — verificación disparada
<code>ClipboardActivity</code>	Generar OTP	x=540,y=779	CLICKJACKED confirmado — OTP generado
<code>AccessControl1Issue1Activity</code>	Verificación de PIN	x=540,y=859	CLICKJACKED confirmado — PIN aceptado

Configuración del overlay del atacante utilizada para la confirmación:

```
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
);
```

Artefactos de evidencia preservados:

- `/workspace/overlay_attack_logcat.txt` — entradas `CLICKJACKED!` con marca temporal
- `/workspace/final_attack_demo.mp4` — grabación de pantalla del clickjacking en `HardCodeActivity`
- `/workspace/clickjacking_pin.mp4` — grabación de pantalla del clickjacking en la verificación de PIN
- `/workspace/screen_overlay_test2.png` — captura de pantalla del overlay de Descargar factura
- `/workspace/screen_hardcode_overlay.png` — captura de pantalla del overlay del código promocional
- `/workspace/screen_clipboard_overlay.png` — captura de pantalla del overlay de OTP
- `/workspace/screen_pin_overlay.png` — captura de pantalla del overlay de PIN

Evidencia de validación

Intento de validación 1

La aplicación de overlay del atacante registró `CLICKJACKED!` para el botón Descargar factura de `AccessControl1ViewActivity` en x=540,y=393 y la aplicación objetivo ejecutó la acción de descarga

Intento de validación 2

La aplicación de overlay del atacante registró `CLICKJACKED!` para el botón Verificar código promocional de `HardCodeActivity` en x=540,y=935 y la aplicación objetivo ejecutó la verificación

Intento de validación 3

La aplicación de overlay del atacante registró `CLICKJACKED!` para el botón Generar OTP de `ClipboardActivity` en x=540,y=779 y la aplicación objetivo generó el OTP

Intento de validación 4

La aplicación de overlay del atacante registró `CLICKJACKED!` para la verificación de PIN de `AccessControlIssue1Activity` en x=540,y=859 y la aplicación objetivo aceptó el PIN

Intento de validación 5

Las grabaciones de pantalla `/workspace/final_attack_demo.mp4` y `/workspace/clickjacking_pin.mp4` capturaron las cadenas completas de explotación

Intento de validación 6

Las capturas de pantalla `/workspace/screen_overlay_test2.png`, `/workspace/screen_hardcode_overlay.png`, `/workspace/screen_clipboard_overlay.png`, `/workspace/screen_pin_overlay.png` documentan el posicionamiento del overlay

Intento de validación 7

Causa raíz

La aplicación carece por completo de defensas contra clickjacking en sus 25 actividades. El análisis estático confirmó cero instancias de `android:filterTouchesWhenObscured="true"`, `onFilterTouchEventForSecurity()` o `FLAG_SECURE` en ninguna parte del código fuente decompilado. Además, `AccessControl1ViewActivity` se exporta mediante un intent-filter en `AndroidManifest.xml` y no realiza comprobaciones de autenticación en sus métodos del ciclo de vida, permitiendo el lanzamiento externo directo sin verificación de PIN.

Código vulnerable

`AccessControl1ViewActivity.java` no contiene ningún control de autenticación en `onCreate()` ni sobrescrituras de `onResume()` o `onNewIntent()` para validar el estado de la sesión:

```
@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);
}
```

El `AndroidManifest.xml` expone la actividad sin `android:exported="false"`:

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

Evidencia de explotación

Pruebas dinámicas ampliadas de overlay: Las pruebas sistemáticas en las 25 actividades confirmaron que **22/25 son vulnerables** a tapjacking. Solo

`SplashActivity` y `XSSActivity` (sin elementos interactivos nativos) y `BinaryPatchingActivity` (botón deshabilitado de forma predeterminada) no fueron explotables. No se detectaron mitigaciones parciales. Los flujos de trabajo de alto riesgo confirmados como explotables incluyen:

- `InputValidationsOSCMDInjectionMain2Activity` — ejecución de comandos del sistema operativo mediante clickjacking del botón RUN
- `TrafficActivity` — clickjacking del botón HTTP/HTTPS/Certificate Pinning
- `RootDetectionActivity` / `EmulatorDetectionActivity` — bypass de la comprobación de integridad mediante overlay

- `InsecureStorageSdCardActivity` / `InsecureStorageSQLiteActivity` / `InsecureStorageSharedPrefs` / `InsecureStorageTempActivity` — acciones de almacenamiento de credenciales activables mediante overlay
- `SQLInjectionActivity` — ejecución de consultas SQL mediante overlay

Bypass de autenticación por deep link: Las pruebas dinámicas con ADB confirmaron el acceso directo a `AccessControl1ViewActivity` posterior a la autenticación sin introducir el PIN:

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

`dumpsys activity activities` mostró inmediatamente:

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

La validación de inicio en frío también tuvo éxito tras `am force-stop` :

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

Pila de tareas resultante:

```
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}
```

No apareció ningún indicador de PIN ni diálogo de autenticación ni en el inicio en caliente ni en el inicio en frío. Logcat no contenía mensajes de verificación de PIN ni de comprobación de sesión durante el lanzamiento.

Configuración del overlay del atacante:

```
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
);
```

Artefactos de evidencia preservados de la validación:

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

Recomendación

Mitigación inmediata

Agregue `android:filterTouchesWhenObscured="true"` a cada vista interactiva crítica en los archivos XML de layout sensibles. Esto bloquea de inmediato los eventos de toque provenientes de ventanas superpuestas:

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

Corrección permanente

Aplice `WindowManager.LayoutParams.FLAG_SECURE` en el método `onCreate()` de cada actividad sensible para bloquear la captura de pantalla y el reconocimiento por superposición:

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

Como alternativa, aplíquelo mediante un tema en `AndroidManifest.xml` :

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

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

Implemente el filtrado de eventos de toque en tiempo de ejecución como defensa en profundidad:

```
@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);
}
```

Proteja el deep link exportado añadiendo un control de autorización en `AccessControl1ViewActivity.onCreate()` :

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

Verificación

1. Vuelva a ejecutar la aplicación complementaria de superposición e intente tocar cada botón sensible. Verifique que la aplicación objetivo no responda y que los registros muestren `FLAG_WINDOW_IS_OBSCURED` .
2. Ejecute `adb shell screencap` sobre las actividades protegidas y confirme que aparece una pantalla negra o una excepción de seguridad.
3. Ejecute `grep -r "filterTouchesWhenObscured" /workspace/decompiled_source/` y confirme que existen coincidencias en las vistas críticas.

ALTO Credenciales de autenticación en bruto y consultas SQL filtradas al Logcat de Android

Múltiples actividades escriben incondicionalmente nombres de usuario, contraseñas y sentencias SQL en texto plano en los logs del sistema de Android mediante `Log.e` , `Log.v` y `System.out.println` , lo que permite la exfiltración de credenciales por cualquier aplicación con acceso a los logs o por un atacante con `adb` .

El análisis estático de `owasp.sat.agoat` identificó tres actividades en las que datos sensibles introducidos por el usuario fluyen directamente a sinks de registro sin redactar. `InsecureLoggingActivity` emite el nombre de usuario y la contraseña en bruto a logcat y la salida estándar en cada clic del botón.

`InsecureStorageSQLiteActivity` registra la sentencia SQL INSERT completa que contiene credenciales en texto plano, mientras que `SQLInjectionActivity` filtra tanto la consulta SELECT en bruto como los registros de base de datos resultantes. Dado que los registros de Android son legibles por cualquier usuario en muchos dispositivos y fácilmente accesibles mediante `adb logcat` , esto crea un canal de divulgación de información de alto impacto que no requiere complejidad alguna para su explotación.

Detalles técnicos

Causa raíz

Los desarrolladores utilizaron `android.util.Log` y `System.out.println` para depurar las entradas de usuario y las operaciones de base de datos. No se aplica sanitización, control condicional ni supresión antes de que las cadenas se escriban en el búfer de registro del sistema.

Código vulnerable

`InsecureLoggingActivity.java` (líneas 59 y 70):

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

`InsecureStorageSQLiteActivity.java` (línea 87):

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

`SQLInjectionActivity.java` (líneas 82–83 y 93–94):

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

Evidencia de explotación

[Análisis estático] La auditoría estática trazó un flujo de datos incondicional desde `EditText.getText()` a través de la concatenación de `StringBuilder` directamente hacia los sinks de registro. Los fragmentos anteriores se extrajeron textualmente del árbol de fuentes de compilado en `sources/owasp/sat/agoat/`. No se realizó ejecución dinámica, por lo que no se obtuvo captura de logcat en tiempo real; sin embargo, la ruta de código es incondicional y emitirá los datos en cada invocación.

Evidencia de validación

Causa raíz

El análisis estático source-to-sink confirmó que tres actividades en `owasp.sat.agoat` pasan credenciales de usuario sin procesar y cadenas de consulta SQL directamente a los sinks de `android.util.Log` y `System.out.println` sin ninguna sanitización, control condicional ni supresión. Un grep dirigido en el código fuente de compilado verificó la ausencia de referencias a `BuildConfig.DEBUG`, comprobaciones de `Log.isLoggable()` y cualquier función de enmascaramiento (`replace`, `mask`, `redact`, `substring`) en los tres archivos afectados. El `AndroidManifest.xml` además declara `android:debuggable="true"`, lo que garantiza que el acceso a adb logcat esté trivialmente disponible y amplifica el canal de divulgación.

Código vulnerable

`InsecureLoggingActivity.java` (líneas 45–70) — `onClick` dentro de `InsecureLoggingActivity$onCreate$1` :

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

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

`InsecureStorageSQLiteActivity.java` (líneas 64–87) — Registro incondicional de un INSERT SQL fuera del `try/catch` :

```
// Lines 64-74: Raw SQL construction
StringBuilder sb = new StringBuilder();
```

```

sb.append("INSERT INTO users (username, password) VALUES('");
sb.append(username2.getText().toString()); // line 68
sb.append(",");
sb.append(password2.getText().toString()); // line 72
sb.append("'");
String qry = sb.toString(); // line 74

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

```

`SQLInjectionActivity.java` (líneas 56–94) — Registro de SELECT sin procesar y del conjunto de resultados:

```

// 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

```

Evidencia de explotación

[Análisis estático] La validación realizó trazas formales source-to-sink en las tres actividades, confirmando flujos de datos incondicionales desde

`EditText.getText()` hacia los sinks de registro:

1. **InsecureLoggingActivity** — Las líneas 54 y 58 introducen el nombre de usuario y la contraseña sin procesar en `StringBuilder sb`; la línea 59 emite mediante `Log.e`. Las líneas 65 y 69 alimentan un segundo `StringBuilder` emitido a `System.out` en la línea 70. Ninguno de los dos sinks está protegido por una condición `if (DEBUG)`.
2. **InsecureStorageSQLiteActivity** — La línea 87 (`Log.v("Query", qry)`) se encuentra **después** del bloque `try/catch` (líneas 75–86), por lo que se ejecuta tanto si `execSQL` tiene éxito como si lanza una excepción. La cadena SQL INSERT completa contiene la contraseña en texto plano.
3. **SQLInjectionActivity** — Las líneas 82–83 registran el conjunto de resultados completo (incluidas las contraseñas de la base de datos en texto plano) cuando `getCount() > 0`. Las líneas 93–94 se ejecutan **incondicionalmente** en cada clic de botón, emitiendo la consulta SELECT sin procesar y un búfer que contiene registros del conjunto de resultados o la entrada del usuario más un sufijo `"not found"`.

La auditoría del `AndroidManifest.xml` confirmó además `android:debuggable="true"` (línea 5), lo que trivializa la explotación en dispositivos físicos porque un atacante con acceso USB/adb puede ejecutar `adb logcat` y capturar las credenciales y sentencias SQL emitidas sin necesidad de rootear el dispositivo ni eludir las restricciones de acceso a logcat.

Limitación de la evidencia: No se realizó ejecución dinámica ni captura de logcat en tiempo real durante esta validación; la explotabilidad se confirmó enteramente mediante análisis estático del código fuente decompilado.

Recomendación

Acción inmediata

Elimine o comente todas las llamadas a `Log.e`, `Log.d`, `Log.v` y `System.out.println` en los archivos identificados que procesan credenciales de usuario o consultas SQL.

Corrección permanente

Implemente una utilidad de registro centralizada que inspeccione los mensajes en busca de patrones sensibles y los reemplace por `[REDACTED]` antes de su emisión. Utilice reglas de ProGuard o R8 para eliminar las llamadas a `Log.*` de las compilaciones de lanzamiento:

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

Verificación

Compile la aplicación en modo de lanzamiento, realice los flujos de entrada de credenciales y ejecute `adb logcat | grep -i "Error\|Query"`. Confirme que no aparecen nombres de usuario, contraseñas ni textos de consultas SQL en los logs.

ALTO Inyección SQL mediante execSQL sin sanitizar en InsecureStorageSQLiteActivity

InsecureStorageSQLiteActivity construye una sentencia INSERT concatenando dos valores EditText sin sanitizar en execSQL(), lo que permite la inyección de consultas apiladas, la manipulación arbitraria de registros y el compromiso total de la integridad de la base de datos local.

La actividad InsecureStorageSQLiteActivity persiste las credenciales del usuario en la base de datos SQLite local `agoat` construyendo una sentencia `INSERT` mediante concatenación directa de cadenas de dos campos `EditText` controlados por el atacante. La cadena resultante se pasa a `SQLiteDatabase.execSQL()`, que acepta múltiples sentencias SQL separadas por punto y coma. Esto permite a un atacante finalizar de forma anticipada el `INSERT` original y ejecutar sentencias secundarias arbitrarias, incluyendo `INSERT`, `UPDATE`, `DELETE` y `CREATE TABLE`. Tanto los campos de nombre de usuario como de contraseña son puntos de inyección vulnerables, y las escrituras persistentes implican que el compromiso de la integridad es inmediato y duradero. La actividad comparte el mismo archivo de base de datos que SQLInjectionActivity, lo que permite que los datos inyectados influyan en consultas orientadas a la lectura.

Detalles técnicos

Hallazgo #1

Causa raíz

En `owasp/sat/agoat/InsecureStorageSQLiteActivity.java` en la línea 80, la aplicación construye una sentencia `INSERT` concatenando directamente dos entradas de la interfaz sin sanitizar en la cadena SQL y la ejecuta mediante `execSQL()`. A diferencia de `rawQuery()`, `execSQL()` no restringe la ejecución a una única sentencia, lo que permite la inyección de consultas apiladas.

Código vulnerable

El patrón de construcción de consultas vulnerable identificado durante el análisis estático es:

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

Tanto `userName` como `password` provienen de widgets `EditText` (`R.id.userName` y `R.id.password`) sin validación de entradas ni parametrización.

Evidencia de explotación

[Simulado] Se utilizó una base de datos SQLite simulada en local con el esquema exacto de `agoat` para ejecutar payloads reconstruidos. Los siguientes resultados confirman la ejecución arbitraria de sentencias:

- Escape de comillas / terminación anticipada mediante el campo username:** La entrada `test','test'); --` provocó que la base de datos ejecutara `INSERT INTO users (username, password) VALUES('test','test'); --')`, insertando efectivamente `(test, test)` y comentando el resto.
- Inyección multi-sentencia mediante el campo username:** La entrada `test','test'); INSERT INTO users (username, password) VALUES('attacker','attacker'); --` provocó la ejecución de **dos sentencias discretas**. La simulación confirmó que se insertaron **dos filas**, demostrando la ejecución arbitraria de sentencias secundarias.
- Inyección multi-sentencia mediante el campo password:** La entrada `test'); INSERT INTO users (username, password) VALUES('attacker2','attacker2'); --` también provocó la inserción de **dos filas**, confirmando que ambos campos de entrada son vulnerables de forma independiente.
- Compromiso de integridad mediante UPDATE:** Un payload apilado agregó `UPDATE users SET password='hacked' WHERE username='admin'; --` después del `INSERT`. La simulación confirmó que la contraseña de `admin` fue modificada permanentemente a `hacked`.
- Exfiltración de datos mediante CREATE TABLE:** Un payload apilado creó una nueva tabla `stolen_data` e insertó todas las credenciales existentes en ella, confirmando la exfiltración completa de datos mediante objetos de esquema controlados por el atacante.

Evidencia de validación

Causa raíz

[Análisis estático] El código fuente descompilado con JADX en

`/workspace/source_code_analysis_b456ad45d2f249a09fc5ea8e695da9e6/base.apk/sources/owasp/sat/agoat/InsecureStorageSQLiteActivity.java` confirma el sink vulnerable en la **línea 80**. El método `onCreate()` (líneas 44–90) obtiene los handles de `EditText` para `R.id.userName` (línea 58) y `R.id.password` (línea 59), asocia un listener de clic al botón `Save` (`R.id.SQLButton`) (línea 60) y, dentro de `onClick()` (línea 62), construye la consulta mediante concatenación directa de cadenas antes de pasarla a `this.mDB.execSQL(qry)`. No hay placeholders `?`, `bindArgs` ni sanitización intermedia.

Código vulnerable

El código fuente Java descompilado (transpilado de Kotlin por JADX) muestra el patrón de concatenación exacto:

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

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

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

La validación también confirmó que `Log.v("Query", qry)` (línea 81) y `Log.d("Error", ...)` (línea 84) escriben la cadena de consulta completa —incluido cualquier payload inyectado— en los registros del sistema, proporcionando un canal lateral adicional para un atacante con permiso `READ_LOGS` o acceso físico a ADB.

Evidencia de explotación

[Análisis estático] La validación confirmó que ambos campos de entrada alcanzan el sink vulnerable de forma independiente sin validación, escape ni lógica de lista blanca. El layout XML (`activity_insecure_storage_sqlite.xml`) vincula `R.id.userName` y `R.id.password` directamente a los widgets `EditText` origen, y el trazado inverso desde la línea 80 revela cero controles defensivos.

El impacto entre actividades se validó analizando `SQLInjectionActivity.java` (líneas 45–97), que abre el archivo de base de datos idéntico:

```
// 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);
```

Dado que ambas actividades referencian la misma base de datos `"aGoat"` y `execSQL()` aplica los cambios inmediatamente en disco, cualquier payload apilado que comprometa la integridad (p. ej., `UPDATE`, `DELETE`, `CREATE TABLE`) ejecutado mediante `InsecureStorageSQLiteActivity` es visible instantáneamente para las consultas `SELECT` de `SQLInjectionActivity`.

La auditoría del manifiesto amplifica aún más el impacto: `android:allowBackup="true"` y `android:debuggable="true"` permiten la extracción y manipulación offline de la base de datos `aGoat` mediante `adb backup` o `run-as`, mientras que `clearTextTrafficPermitted="true"` y la confianza en CA de usuario en `network_security_config.xml` habilitan una interceptación de red trivial.

Recomendación

Mitigación inmediata

Deshabilitar el botón `Save` o agregar restricciones de longitud de entrada del lado del cliente como solución temporal mientras se despliega la solución definitiva. Eliminar `android:debuggable="true"` del manifiesto.

Solución definitiva

Reemplazar la llamada a `execSQL()` por `SQLiteDatabase.insert()` mediante `ContentValues`, lo que elimina por completo la concatenación de cadenas:

```
ContentValues values = new ContentValues();
values.put("username", userName.getText().toString());
```

```
values.put("password", password.getText().toString());
sqliteDatabase.insert("users", null, values);
```

Si la aplicación debe usar `execSQL()` para la gestión del esquema, asegurarse de que `execSQL()` **nunca** se invoque con cadenas que contengan entrada del usuario. Toda la manipulación de datos debe utilizar APIs parametrizadas (`insert()`, `update()`, `delete()`) o `SQLiteStatement` con parámetros enlazados.

Verificación

1. Volver a ejecutar el análisis estático para confirmar que no existen ocurrencias de `execSQL()` que concatenen entrada del usuario.
2. Intentar payloads de múltiples sentencias como `'); DELETE FROM users; --` contra la build corregida y verificar que solo se produce la inserción prevista de una única fila.
3. Revisar la tabla `users` tras cada interacción de prueba para confirmar que no se producen cambios inesperados en el esquema ni filas adicionales.

ALTO Firebase Database expuesta públicamente

exponer públicamente una base de datos Firebase, lo que puede provocar acceso no autorizado, manipulación de datos, divulgación de información, ataques maliciosos y pérdida de reputación.

Firebase es una plataforma de desarrollo de aplicaciones móviles y web que proporciona diversas herramientas y servicios a los desarrolladores, incluida una base de datos en tiempo real. Si una base de datos Firebase está expuesta públicamente, cualquier persona puede acceder y manipular los datos almacenados sin autenticación ni autorización.

El impacto en la seguridad de exponer públicamente una base de datos Firebase puede ser grave, ya que puede conducir a los siguientes riesgos de seguridad:

- Acceso no autorizado: Cualquier persona puede acceder a los datos de la base de datos sin autenticación ni autorización, lo que permite que usuarios no autorizados obtengan información sensible almacenada en ella.
- Manipulación de datos: Los atacantes pueden modificar, agregar o eliminar datos en la base de datos, lo que puede provocar pérdida de datos o generar información inexacta con consecuencias graves para la aplicación o el negocio.
- Divulgación de información: Los atacantes pueden acceder a información sensible como contraseñas, datos personales, información financiera y otros datos confidenciales almacenados en la base de datos.
- Ataques maliciosos: Los atacantes pueden ejecutar diversos ataques maliciosos, como ataques de inyección, cross-site scripting y otros tipos de ataques para explotar las vulnerabilidades de la aplicación conectada a la base de datos Firebase.
- Pérdida de reputación: Si se expone información sensible, puede generarse una pérdida de confianza y credibilidad entre los clientes, lo cual impacta significativamente en la reputación del negocio.

Para verificar si una base de datos Firebase está expuesta públicamente, puede utilizar el siguiente comando curl:

<> BASH

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

Reemplace `<project-id>` por el ID del proyecto de Firebase que desea verificar. Este comando intentará recuperar el nodo raíz de la base de datos Firebase en formato JSON.

Si la base de datos Firebase está expuesta públicamente, debería poder recuperar los datos sin autenticación ni autorización.

Si la base de datos está protegida y configurada correctamente, recibirá un mensaje de error indicando que no está autorizado para acceder a los datos.

Detalles técnicos

Hallazgo #1

Firebase `https://androgoat-42597.firebaseio.com` es **públicamente** accesible y puede validarse mediante el siguiente comando: `curl --request GET --url https://androgoat-42597.firebaseio.com/.json`. La clave se encontró en `res/values/strings.xml`.

Recomendación

Para hacer privada una base de datos de Firebase, puede seguir estos pasos técnicos:

- Habilitar Firebase Authentication: permite autenticar usuarios y controlar el acceso a su base de datos. Puede utilizar esta función para autenticar usuarios mediante diversos proveedores, como correo electrónico y contraseña, Google, Facebook, entre otros.
- Configurar Firebase Security Rules: le permiten definir cómo se accede a los datos y qué usuarios pueden leer o escribir en la base de datos. Puede establecer reglas basadas en la autenticación del usuario y en rutas de datos específicas de la base de datos.

Referencias

- [Firebase Security Rules](#)

ALTO Vulnerabilidad de primos comunes LOGJAM

Uso de primos Diffie-Hellman ampliamente compartidos que permiten ataques precomputados por estados-nación contra grandes porciones del tráfico de Internet.

Esta vulnerabilidad indica que el servidor utiliza números primos Diffie-Hellman ampliamente compartidos, lo que permite ataques de precomputación eficientes y facilita que adversarios estatales descifren pasivamente grandes volúmenes de tráfico de Internet.

LOGJAM-common_primes se produce cuando los servidores utilizan los mismos parámetros DH estándar (números primos) compartidos por millones de instalaciones. Si bien el uso de primos compartidos se consideraba seguro, el algoritmo de criba de campos numéricos permite a los atacantes precomputar costosas tablas criptográficas una vez por primo y, posteriormente, romper rápidamente cualquier conexión que utilice ese primo.

Cómo funciona:

1. El atacante identifica primos DH de uso extendido (p. ej., parámetros por defecto de Apache/OpenSSL)
2. Realiza una precomputación costosa (semanas de procesamiento) para el primo objetivo
3. Captura pasivamente los intercambios de claves DH de cualquier servidor que use ese primo
4. Usa las tablas precomputadas para romper conexiones individuales en minutos
5. Escala el ataque a millones de servidores que comparten el mismo primo

Requisitos:

- El servidor utiliza números primos DH comunes o por defecto (de 1024 bits o menos)
- Recursos computacionales significativos para la precomputación inicial (~100 millones de dólares para 1024 bits)
- Capacidad para capturar tráfico de red de las conexiones objetivo
- No se requiere capacidad de man-in-the-middle; es un ataque puramente pasivo

Ejemplo de escenario: Un importante proveedor de nube utiliza parámetros DH de 1024 bits por defecto en miles de servidores. Un actor estatal dedica meses a precomputar tablas criptográficas para este primo específico. Una vez finalizado el proceso, puede monitorear pasivamente el tráfico de Internet y descifrar cualquier conexión TLS a estos servidores en tiempo real, afectando a millones de usuarios sin interferencia activa ni detección.

Romper el primo de 1024 bits más común permitiría escuchar pasivamente el tráfico del 18 % de los principales dominios HTTPS, mientras que romper los dos primos más comunes comprometería el 66 % de los servidores VPN y el 26 % de los servidores SSH a nivel mundial.

Detalles técnicos

Hallazgo #1

LOGJAM-common_primes detectado en `demo.testfire.net`

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

Recomendación

Para mitigar los ataques de primos comunes de LOGJAM:

Defensa principal: genere parámetros DH únicos:

```
# 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;
```

Evite primos por defecto o comunes:

Nunca utilice estos primos ampliamente comprometidos:

- Primo por defecto de Apache de 512 bits (usado por millones de servidores)
- Primo por defecto de OpenSSL de 512 bits (presente en SSLeay desde 1995)
- Primos estándar de RFC 5114 (posiblemente influidos por la NSA)

Migre a ECDHE:

Solución óptima: elimine la vulnerabilidad de DH por completo:

```
# 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;
```

Use parámetros de 2048 bits como mínimo:

Si requiere DHE:

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

Pruebas para detectar primos comunes:

```
# 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
```

Protecciones adicionales:

- Use TLS 1.3, que elimina completamente DH de campo finito
- Implemente certificate pinning para detectar intentos de MITM
- Supervise patrones de conexión inusuales que indiquen análisis de tráfico masivo
- Considere las implicaciones de Perfect Forward Secrecy al elegir cipher suites

Observación crítica: incluso los parámetros DH robustos de 1024 bits se vuelven vulnerables al compartirse entre muchos servidores, ya que el costo de precomputación puede amortizarse. La unicidad es tan importante como el tamaño.

Referencias

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

ALTO Vulnerabilidad de inyección SQL en móvil

La vulnerabilidad de inyección SQL en aplicaciones móviles permite a un atacante ejecutar consultas SQL maliciosas en la base de datos de una aplicación móvil, lo que puede derivar en acceso no autorizado, fugas de datos o manipulación de datos.

La inyección SQL móvil es una vulnerabilidad que permite a los atacantes inyectar sentencias SQL maliciosas en aplicaciones móviles, lo que potencialmente puede otorgarles acceso no autorizado a datos sensibles o permitirles manipular la base de datos.

Ejemplos

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()
}

```

Detalles técnicos

Hallazgo #1

El parámetro en la posición `0` del método `execSQL` de la clase `SQLiteDatabase` puede ser vulnerable a inyección SQL. Taint trazado desde el componente no alcanzable: `owasp.sat.agoat.InsecureStorageSQLiteActivity`. El TaintElement proviene de: Clase `InsecureStorageSQLiteActivity` Método `onCreate`

Posición `0` Firma `(Landroid/os/Bundle;)V`. Traza de llamadas:

```

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

```

En `android.database.sqlite.SQLiteDatabase.execSQL`

Unable to extract source code.

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

```

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

```

En `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;
}

```

En `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;
}

```

En `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;
}

```

En `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;
}

```

En `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;
}

```

En `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;
    }
}

```

En `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;
}

```

En `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;
}
}

```

Hallazgo #2

Un parámetro controlado por el usuario se utiliza para construir un parámetro SQL vulnerable a inyección SQL Trazado de taint desde componente no alcanzable:

`owasp.sat.agoat.InsecureStorageSQLiteActivity` encontrado en `owasp.sat.agoat.InsecureStorageSQLiteActivity`. Desde el parámetro `0` hasta el sink `void android.database.sqlite.SQLiteDatabase.execSQL(java.lang.String)`. Traza de taint:

```

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()
```

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

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

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

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

```

    }
    v2_2.execSQL(v0_2);
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    android.util.Log.v(Query, v0_2);
    return;
}

```

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

```

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

```

Método `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;  
}
```

Método `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;  
}
```

Método `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;  
}
```

Método `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;
    }
}

```

Hallazgo #3

El parámetro en la posición **0** del método **execSQL** en la clase **SQLiteDatabase** puede ser vulnerable a inyección SQL. Taint rastreado desde un componente no alcanzable: **owasp.sat.agoat.InsecureStorageSQLiteActivity**. El TaintElement proviene de: Clase **InsecureStorageSQLiteActivity** Método **onCreate** Posición **0** Firma **(Landroid/os/Bundle;)V**. Traza de llamadas:

```

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

```

at **android.database.sqlite.SQLiteDatabase.execSQL**

Unable to extract source code.

at **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;
}

```

at **android.database.sqlite.SQLiteDatabase.execSQL**

Unable to extract source code.

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

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

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

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

```

    return;
}

```

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

```

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

```

at `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;
}

```

at `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;
}

```

at `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;
}

```

at `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;
}

```

Se utiliza un parámetro controlado por el usuario para construir un parámetro SQL vulnerable a inyección SQL Taint rastreado desde un componente inalcanzable:

`owasp.sat.agoat.SQLInjectionActivity` encontrado en `owasp.sat.agoat.SQLInjectionActivity`. Desde el parámetro `0` hasta el sink `android.database.Cursor android.database.sqlite.SQLiteDatabase.rawQuery(java.lang.String, java.lang.String[])`. Trazo de taint:

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

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

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

```
}
```

Método `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;
}
```

Método `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;
}
```

Método `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;
}
```

Método `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;
}
```

Método `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;
}
```

Método `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;
}
```

Método `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;
}
```

Método `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;
}
```

Hallazgo #5

Un parámetro controlado por el usuario se utiliza para construir un parámetro SQL vulnerable a inyección SQL Trazado de taint desde componente no alcanzable:

`owasp.sat.agoat.SQLInjectionActivity` encontrado en `owasp.sat.agoat.SQLInjectionActivity` . Desde el parámetro `0` hasta el sink `android.database.Cursor android.database.sqlite.SQLiteDatabase.rawQuery(java.lang.String, java.lang.String[])` . Trazado de taint:

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

Método `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$(this.this$0, this.this$0.openOrCreateDatabase(aGoat, 0, 0));
        android.database.Cursor v3_2 = owasp.sat.agoat.SQLInjectionActivity.access$getMDB$(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;
}
```

Método `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()
        } 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;
}

```

Método `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;
}

```

Método `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;
}

```

Método `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;
}

```

Método `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;
}

```

```
}
```

Método `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;
}
```

Hallazgo #6

Se utiliza un parámetro controlado por el usuario para construir un parámetro SQL vulnerable a inyección SQL Taint trazado desde un componente no alcanzable:

`owasp.sat.agoat.SQLInjectionActivity` encontrado en `owasp.sat.agoat.SQLInjectionActivity` . Desde el parámetro `0` hasta el sink `android.database.Cursor android.database.sqlite.SQLiteDatabase.rawQuery(java.lang.String, java.lang.String[])` . Traza de taint:

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

Método `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;
}

```

Método `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$0.getApplicationContext();
        StringBuilder v4_10 = new StringBuilder();
        v4_10.append(Data not saved: Error ocured - );
        v4_10.append(v2_13.getMessage());
        android.widget.Toast.makeText(v3_8, ((CharSequence) v4_10.toString()), 1).show();
        android.util.Log.v(Query, v0_2);
        return;
    }
    if (v3_2 == null) {
        kotlin.jvm.internal.Intrinsics.throwNpe();
    }
    android.database.Cursor v3_3 = v3_2.rawQuery(v0_2, 0);
    StringBuilder v4_5 = new StringBuilder();
    if (v3_3 == null) {
        String v5_13 = new StringBuilder();
        v5_13.append(User: ());
        android.widget.EditText v6_2 = this.$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;
}

```

Método `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;
}

```

Método `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;
}

```

Método `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;
}

```

Método `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;
    }

```

Método `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;
}

```

Método `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;
}

```

Recomendación

Para mitigar vulnerabilidades de SQL Injection en aplicaciones móviles, tenga en cuenta lo siguiente:

- Utilice consultas parametrizadas o prepared statements para separar el código SQL de la entrada del usuario.
- Sanitice y valide la entrada del usuario antes de insertarla en la base de datos para ayudar a mitigar el SQL Injection de segundo orden.
- Actualice periódicamente el driver SQL para solucionar vulnerabilidades conocidas.

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],  
        );  
    });  
}
```

Referencias

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

MEDIO

Privilegio de administrador determinado por un flag booleano del lado del cliente sin verificación del servidor

BinaryPatchingActivity controla el acceso a la funcionalidad de la interfaz de administración mediante un campo booleano local que nunca es validado por una API de backend, un token criptográfico ni un desafío de reautenticación.

La Activity inicializa un campo `private final boolean isAdmin` a `false` y lo utiliza en `onCreate()` para decidir si habilita un botón de administrador y muestra texto de administrador. Dado que no existe verificación de rol del lado del servidor, verificación de firma ni autenticación step-up, un atacante puede escalar privilegios modificando este flag en tiempo de ejecución mediante Frida o parcheando el bytecode Dalvik del APK. Esto demuestra un vector de escalada de privilegios que depende enteramente de la confianza del lado del cliente, sin que se requiera verificación secundaria.

Detalles técnicos

Hallazgo #1

Causa raíz

`BinaryPatchingActivity` controla el acceso a los elementos de la interfaz de administrador mediante un campo `private final boolean isAdmin` inicializado en `false`. La Activity verifica este indicador durante `onCreate()`; si es `true`, habilita un botón de administrador y actualiza un TextView. No existe ninguna llamada a la API del backend, atestación criptográfica ni autenticación step-up para validar el estado de administrador.

Código vulnerable

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

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

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

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

Evidencia de explotación

[Static Analysis] El campo es una constante en tiempo de compilación sin verificación remota. Un atacante puede utilizar Frida para interceptar

`BinaryPatchingActivity.isAdmin()` mediante un hook y forzar que devuelva `true`, o aplicar un parche binario al APK para modificar el inicializador Dalvik de `const/4 v0, 0x0` a `0x1`. El backend nunca se consulta y no se desencadena ninguna reautenticación cuando se habilita la interfaz de administrador.

Evidencia de validación

Causa raíz

`BinaryPatchingActivity` (paquete `owasp.sat.agoat`) controla el acceso a los elementos de la interfaz de administración mediante un campo `private final boolean isAdmin` declarado en la línea 17 del código fuente Java decompilado (`base.apk/sources/owasp/sat/agoat/BinaryPatchingActivity.java`). El campo recibe el valor predeterminado de la JVM `false` y es inmutable tras la construcción debido al modificador `final`. Durante `onCreate()` (línea 48), la Activity verifica `this.isAdmin`; cuando es `true`, habilita `adminButton` y actualiza `isAdminText`. La validación confirmó que no existen vectores de entrada externos, solicitudes de red, atestación criptográfica, verificación de firma ni autenticación step-up en la Activity ni en su grafo de llamadas alcanzable.

Código vulnerable

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

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

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

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

Cuando `isAdmin` es `false` (valor predeterminado), el layout `activity_binary_patching.xml` muestra `isAdminText` como "Lo sentimos, no tiene acceso a esta función." y `adminButton` permanece deshabilitado (`android:enabled="false"`). No existe ninguna ruta de código alternativa que valide el estado de administrador contra una autoridad remota.

Evidencia de explotación

[Static Analysis] La auditoría del código fuente verificó que `isAdmin` se asigna una sola vez (de forma implícita a `false` en el constructor sintético) y nunca se modifica posteriormente. Una búsqueda en todo el proyecto de `isAdmin\s*=''` devolvió cero coincidencias, lo que confirma que ningún setter, reflexión o código nativo modifica el indicador.

La auditoría del manifiesto (`base.apk/AndroidManifest.xml` , línea 5) reveló:

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

`android:debuggable="true"` permite que Frida u otras herramientas de instrumentación se adjunten **sin acceso root**, reduciendo drásticamente la barrera para que un atacante intercepte `BinaryPatchingActivity.isAdmin()` mediante un hook en tiempo de ejecución y fuerce que devuelva `true` . La Activity no está exportada (`android:exported` toma el valor predeterminado `false`), pero una vez que la aplicación se ejecuta localmente, un atacante puede modificar el inicializador Dalvik de `const/4 vX, 0x0` a `const/4 vX, 0x1` para habilitar de forma permanente la interfaz de administrador en cada inicio. No se desencadena ninguna validación del backend, reautenticación ni verificación de integridad cuando se activa la interfaz de administración.

Recomendación

Inmediato

Elimine el indicador `isAdmin` del lado del cliente de `BinaryPatchingActivity` . En su lugar, consulte un endpoint `/user/role` del backend mediante TLS autenticado y almacene el rol en caché en `EncryptedSharedPreferences` con un TTL breve.

Corrección permanente

Implemente la aplicación de roles del lado del servidor para cada acción privilegiada. El cliente debe enviar un JWT firmado o un token de sesión al backend, que validará el claim de rol antes de devolver datos de administrador o habilitar funciones administrativas:

```
// Lado del cliente: solo control de UI; el backend aplica la autorización
// Lado del servidor: validar token.role == "admin" antes de procesar
```

Verificación

Utilice Frida para hacer un hook en `BinaryPatchingActivity.isAdmin()` que devuelva `true` . Confirme que las acciones de administrador intentadas sigan fallando porque el backend rechaza la solicitud. Revise el código fuente para asegurarse de que ningún booleano local por sí solo pueda habilitar elementos privilegiados de la UI.

MEDIO El Certificate Pinning en TrafficActivity falla silenciosamente debido a un manejo de excepciones demasiado amplio

El CertificatePinner de OkHttp en `TrafficActivity.doPinning()` se anula porque `SSLPeerUnverifiedException` es capturada por un bloque catch amplio y suprimida, lo que permite ataques Man-in-the-Middle silenciosos.

`TrafficActivity.doPinning()` implementa `CertificatePinner` de OkHttp para `owasp.org` con tres pins SHA-256. Sin embargo, toda la solicitud de red está envuelta en `try { ... } catch (Exception e) { e.printStackTrace(); }` , lo que suprime silenciosamente `SSLPeerUnverifiedException` cuando falla el certificate pinning. Combinado con la confianza global en CA instaladas por el usuario, un atacante puede interceptar esta conexión con pinning mediante un certificado proxy sin notificar al usuario.

Detalles técnicos

Hallazgo #1

Causa raíz

El desarrollador agregó *certificate pinning* mediante `CertificatePinner.Builder` de OkHttp, pero envolvió la solicitud con pin dentro de un controlador de excepciones demasiado amplio que imprime el *stack trace* en lugar de fallar de forma segura. Esto anula la garantía de seguridad del *pinning*, ya que la aplicación continúa su ejecución en silencio tras una discrepancia de pin.

Código vulnerable

`TrafficActivity.java` , líneas 140–141:

```
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` , líneas 147–148:

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

Evidencia de explotación

[Análisis Estático] El código fuente descompilado muestra que la lógica de *pinning* está aislada en `doPinning()` y solo aplica *pinning* a `owasp.org` . Dado que la *Network Security Config* de la aplicación ya confía en CA instaladas por el usuario, el *handshake* TLS acepta un certificado proxy. El *pinner* lanza entonces `SSLPeerUnverifiedException` , pero el bloque `catch (Exception e)` demasiado amplio de las líneas 147–148 lo suprime en silencio. Un *hook* de Frida sobre `okhttp3.CertificatePinner.check()` también puede neutralizar la verificación del pin:

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

Evidencia de validación

Causa raíz

La validación confirma que `TrafficActivity.doPinning()` en `owasp/sat/agoat/TrafficActivity.java` , líneas 139–149, es el único punto de aplicación de *certificate pinning* en toda la aplicación. Construye un `CertificatePinner` de OkHttpClient con tres pins SHA-256 para `owasp.org` , pero envuelve la llamada síncrona `execute()` en un único bloque `try` demasiado amplio cerrado por `catch (Exception e) { e.printStackTrace(); }` . Dado que `SSLPeerUnverifiedException` es una subclase de `Exception` , cualquier discrepancia de pin se suprime de forma determinista en lugar de fallar de forma segura. El `AndroidManifest.xml` de la aplicación referencia `android:networkSecurityConfig="@xml/network_security_config"` , y el archivo correspondiente `res/xml/network_security_config.xml` contiene `<certificates src="user"/>` , lo que indica al sistema operativo que confíe en CA instaladas por el usuario. Esto permite que un certificado proxy supere el *handshake* TLS antes de que el *pinner* de OkHttpClient lo rechace.

Código vulnerable

Validado en el árbol de código fuente descompilado

(`/workspace/source_code_analysis_dfcb434d7ae9421693bfb4c0774f91126/base.apk/sources/owasp/sat/agoat/TrafficActivity.java`):

Líneas 139–146:

```
try {
    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();
    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();
}
```

El mismo patrón se refleja en la descompilación alternativa (`TrafficActivity$doPinning$1.java` , línea 28).

Evidencia de explotación

[Análisis Estático] Una búsqueda exhaustiva en todas las fuentes de la aplicación descompilada (`/workspace/jadx_out/sources` , `/workspace/jadx_out2/sources` , `/workspace/decompiled_source/sources` , y directorios de análisis asociados) confirma que `CertificatePinner.Builder` se instancia **solo** dentro de `TrafficActivity.doPinning()` (y su clase interna anónima `TrafficActivity$doPinning$1.invoke2()`), y que el único nombre de host sobre el que se aplica *pinning* es `owasp.org` . No existen en ninguna parte de la base de código bibliotecas de *pinning* alternativas (TrustKit, AndroidPinning), ni manipulaciones personalizadas de `TrustManager` , `HostnameVerifier` o `SSLContext` . El único sitio de llamada a `OkHttpClient.Builder.certificatePinner()` se encuentra en la línea 141 de `TrafficActivity.java` .

El método se activa exclusivamente mediante un clic en un botón de la interfaz (`R.id.PinningButton`) que invoca `doPinning()` dentro de un hilo en segundo plano `doAsync` de Anko. El análisis *source-to-sink* confirma que la respuesta de red solo fluye hacia `android.util.Log.v("Response", body.string())` en la línea 146; no hay *sinks* de interfaz (`Toast` , `TextView`), `SharedPreferences` , archivo ni IPC. En consecuencia, cuando `client.newCall(request).execute()` lanza `SSLPeerUnverifiedException` debido a una discrepancia de pin, la excepción se captura en las líneas 147–148, se imprime en `System.err` y la ejecución continúa en silencio sin notificación alguna al usuario.

Limitación de la evidencia: No se ejecutó explotación dinámica mediante un proxy interceptador activo ni instrumentación en vivo con Frida durante esta fase de validación. La vulnerabilidad se confirma mediante análisis estático determinista de la jerarquía de manejo de excepciones y el aislamiento de alcance en la base de código.

Recomendación

Solución inmediata

Sustituir el bloque catch amplio en `doPinning()` por un manejo de excepciones específico que falle de forma segura:

```
} 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);
}
```

Corrección permanente

1. No silenciar `SSLPeerUnverifiedException` . Tratar cualquier fallo de pinning como un error de seguridad fatal para esa conexión.
2. Extender el pinning a todos los endpoints de producción, no solo a `owasp.org` , utilizando ya sea `CertificatePinner` de OkHttp o `<pin-set>` en `network_security_config.xml` .

Verificación

1. Interceptar la solicitud a `https://owasp.org` en `doPinning()` mediante una CA proxy.
2. Sin la corrección, la aplicación registra la excepción de forma silenciosa y no muestra datos.
3. Con la corrección aplicada, la aplicación debe registrar el fallo de pinning y abortar la conexión, sin que se complete ninguna solicitud de red.

MEDIO Permisos inseguros de almacenamiento externo filtran credenciales (InsecureStorageSDCardActivity)

`InsecureStorageSDCardActivity` escribe las credenciales de usuario en un archivo temporal en el almacenamiento externo y establece explícitamente el archivo como world-readable y world-writable.

Esta vulnerabilidad expone las credenciales de usuario a cualquier aplicación co-instalada. Si bien se confirmó la ausencia del vector de file path traversal previsto en esta compilación APK, la implementación de almacenamiento inseguro existente sigue constituyendo un riesgo de confidencialidad significativo por la persistencia de credenciales en texto claro en almacenamiento compartido.

Detalles técnicos

Hallazgo #1

Causa raíz

La Activity crea un archivo temporal en `Environment.getExternalStorageDirectory()` y, a continuación, invoca `setReadable(true)` y `setWritable(true)` , otorgando acceso de lectura y escritura a todas las aplicaciones del dispositivo. Además, la ruta absoluta del archivo se filtra a logcat a través de `System.out.println()` .

Código vulnerable

`InsecureStorageSDCardActivity.java` (líneas 68–76):

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

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

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

Evidencia de explotación

[Static Analysis] El código fuente confirma que cada vez que se pulsa el botón Save se crea un archivo legible por cualquier aplicación que contiene el nombre de usuario, la contraseña y la marca de tiempo concatenados. Cualquier aplicación de terceros que cuente con el permiso `READ_EXTERNAL_STORAGE` puede enumerar y leer estos archivos. La filtración mediante logcat (`System.out.println`) revela además la ruta absoluta, eliminando la necesidad de realizar una enumeración del sistema de archivos.

Escenario de ataque: Una aplicación maliciosa con el permiso `READ_EXTERNAL_STORAGE` monitoriza logcat o enumera el directorio raíz del almacenamiento externo en busca de archivos `users*.tmp` y extrae las credenciales en texto claro.

Evidencia de validación

Causa raíz

La aplicación apunta a `targetSdkVersion=26` (Android 8.0), lo que fuerza el comportamiento heredado del almacenamiento externo, donde `Environment.getExternalStorageDirectory()` devuelve un espacio de nombres compartido y globalmente accesible (p. ej., `/sdcard/` o `/storage/emulated/0/`). Los archivos creados en este directorio son inherentemente accesibles para cualquier aplicación co-instalada que cuente con el permiso `READ_EXTERNAL_STORAGE`, porque el montaje subyacente de sdcardfs/FUSE/FAT32 emulado no aplica los bits de permisos de Unix. La validación confirmó que `android:debuggable="true"` y `android:allowBackup="true"` están ambos habilitados en el manifiesto, agravando el riesgo para la confidencialidad mediante la inspección de memoria en tiempo de ejecución y la extracción de copias de seguridad adb.

Código vulnerable

La validación del código fuente decompilado (`base.apk/sources/owasp/sat/agoat/InsecureStorageSDCardActivity.java`) identificó los números de línea exactos de cada operación insegura y rastreó la construcción completa de los datos en texto claro:

Construcción de credenciales escrita en disco (líneas 45–51):

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

Creación de archivo en almacenamiento compartido (líneas 68–70):

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

Filtración de ruta en logcat (línea 71):

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

Invocaciones de permisos (líneas 72–73):

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

[Static Analysis] Se verificó que las invocaciones a `setReadable(boolean)` y `setWritable(boolean)` corresponden a las sobrecargas de un solo argumento que solo afectan al propietario. En el almacenamiento externo heredado de la API 26, esta distinción es funcionalmente irrelevante porque el sistema de archivos emulado no respeta los bits de `chmod`; en la práctica, cualquier archivo ubicado en el directorio raíz del almacenamiento externo compartido es legible por todas las aplicaciones que cuenten con `READ_EXTERNAL_STORAGE`.

Evidencia de explotación

[Static Analysis] **Contexto del manifiesto y el SDK:**

- `AndroidManifest.xml` declara `WRITE_EXTERNAL_STORAGE` y `READ_EXTERNAL_STORAGE`.
- `targetSdkVersion="26"` y `minSdkVersion="18"` se extrajeron de `base.apk/apktool.yml`.
- `android:debuggable="true"` y `android:allowBackup="true"` están ambos presentes.

[Static Analysis] **Ausencia de mitigaciones:**

- Una búsqueda en todo el proyecto de `setReadable(false)`, `setWritable(false)`, `chmod`, `setExecutable` y `setPermissions` dentro del paquete de la aplicación devolvió **cero coincidencias**. No existe en ningún punto de la base de código lógica de restricción de permisos, limpieza o aislamiento en subdirectorios.

[Static Analysis] **Sin validación de entradas:**

- `username.getText().toString()` y `password.getText().toString()` se concatenan directamente en la cadena `data` sin comprobaciones de nulidad, límites de longitud ni saneamiento.

[Static Analysis] **Descubrimiento predecible:**

- `File.createTempFile("users", "tmp", ...)` genera archivos `users*.tmp` en el directorio raíz del almacenamiento externo. El `System.out.println` incondicional filtra la ruta absoluta (p. ej., `userinfo /sdcard/users12345.tmp`) a logcat, eliminando la necesidad de enumerar el sistema de archivos. Una aplicación maliciosa co-instalada con `READ_EXTERNAL_STORAGE` puede apuntar directamente a esa ruta y extraer la cadena de credenciales en texto claro.

Recomendación

Corrección inmediata

Elimine las llamadas explícitas a `setReadable(true)` y `setWritable(true)` de `InsecureStorageSDCardActivity.java`.

Corrección permanente

Nunca almacene datos de autenticación sensibles en el almacenamiento externo. Mueva la escritura del archivo al directorio interno privado de la aplicación y confíe en el modelo de permisos predeterminado de Android:

```
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();
```

Además, elimine la sentencia de log `System.out.println` que filtra la ruta absoluta.

Verificación

- Intercepte la creación del archivo en un entorno de ejecución y confirme que el nuevo archivo se crea en `/data/data/owasp.sat.agoat/files/`.
- Verifique que el modo del archivo no otorgue acceso de lectura/escritura a otros UID(`ls -l` muestra `rw-----`).
- Confirme que logcat ya no emita la ruta absoluta del archivo cuando se toque el botón Save.

MEDIO Pantallas sensibles que permiten capturas de pantalla no autorizadas por ausencia de FLAG_SECURE

Todas las 25 actividades de `owasp.sat.agoat` carecen de `FLAG_SECURE`, lo que permite la captura y grabación sin restricciones de pantallas sensibles, como la introducción del PIN, la entrada de credenciales y la generación de OTP.

La ausencia de `windowManager.LayoutParams.FLAG_SECURE` en cada ventana de actividad permite capturar pantalla y grabar vídeo de flujos sensibles sin restricciones mediante `adb` y herramientas de terceros. Esto se confirmó dinámicamente en las pantallas de verificación de PIN, introducción de credenciales, generación de OTP y pago. Un atacante o un proceso comprometido puede recolectar credenciales y tokens sensibles directamente de las capturas de pantalla sin interacción del usuario, y utilizar este reconocimiento para diseñar ataques overlay más precisos.

Detalles técnicos

Hallazgo #1

Causa raíz

Toda actividad en `owasp.sat.agoat` inicializa su ventana sin `WindowManager.LayoutParams.FLAG_SECURE`. Este flag es el mecanismo estándar de Android para impedir que las utilidades de captura de pantalla del sistema, `adb` y las aplicaciones de grabación de pantalla de terceros capturen el contenido de la pantalla.

Código vulnerable

[Análisis estático] Ninguna actividad invoca `getWindow().addFlags(WindowManager.LayoutParams.FLAG_SECURE)` ni `getWindow().setFlags(...)` en `onCreate()`. Los archivos fuente Java y smali decompilados de las 25 actividades muestran cero referencias a `FLAG_SECURE`.

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

Evidencia de explotación

Las pruebas dinámicas en el emulador confirmaron que era posible capturar y grabar la pantalla sin restricciones en todas las pantallas sensibles:

```
# Éxito en las pantallas de verificación de PIN, introducción de credenciales, OTP y pago
adb shell screencap /sdcard/screen_pin_overlay.png
adb shell screenrecord /sdcard/record.mp4
```

El análisis de `dumpsys window` confirmó que no había ningún flag `FLAG_SECURE` presente en ninguna ventana de la aplicación objetivo.

La evidencia registrada incluye:

- `/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`

Esto permite que una aplicación maliciosa o un dispositivo comprometido recopile credenciales, PINs, OTPs y números de tarjetas de crédito directamente a partir de capturas de pantalla sin el conocimiento del usuario.

Evidencia de validación

Intento de validación 1

El comando `adb shell screencap` se ejecutó con éxito en todas las pantallas sensibles, incluidas las de verificación de PIN e introducción de credenciales

Intento de validación 2

`adb shell screenrecord` capturó ataques de superposición en tiempo real sin restricciones

Intento de validación 3

El análisis de `dumpsys window` confirmó que no hay `FLAG_SECURE` presente en las ventanas de la aplicación objetivo

Intento de validación 4

Las capturas de pantalla `/workspace/screen_pin_overlay.png`, `/workspace/screen_hardcode_overlay.png`, `/workspace/screen_clipboard_overlay.png`, `/workspace/screen_overlay_test2.png` mostraron contenido de pantalla sin restricciones

Intento de validación 5

Causa raíz

`AccessControl1ViewActivity` en `owasp.sat.agoat` está implícitamente exportada porque su declaración en el manifiesto incluye un `<intent-filter>` con `action.VIEW` y un esquema personalizado `androgoat://vulnapp`, pero omite `android:exported="false"`. Además, los puntos de entrada de la actividad no contienen lógica de autenticación, aunque la aplicación requiere la verificación de PIN en la actividad complementaria `AccessControlIssue1Activity`.

Código vulnerable

Declaración del manifiesto de `/workspace/agoat_decode/AndroidManifest.xml`:

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

```

        <data android:host="vulnapp" android:scheme="androgoat"/>
    </intent-filter>
</activity>

```

El análisis estático de `/workspace/agoat_analysis/sources/owasp_sat_agoat_AccessControl1ViewActivity.java` y el smali correspondiente confirman que:

- `onCreate(Bundle)` establece la vista de contenido y adjunta un `OnClickListener` que invoca `startService()`.
- `onNewIntent(Intent)` **no está sobrescrito**.
- `onResume()` **no está sobrescrito**.
- **No hay referencias** a `verifyIfPinSet()`, `isPinCorrect()`, `verifyPINView()`, ni a la validación de PIN en preferencias compartidas.

Evidencia de explotación

El dispositivo tenía un PIN configurado (`pinSet=true`), hash MD5 de `1234`:

```

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

```

La explotación dinámica mediante ADB tuvo éxito sin ninguna solicitud de PIN:

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

Salida:

```

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` confirmó que la actividad estaba enfocada:

```

mCurrentFocus=Window{10344ff u0 owasp.sat.agoat/owasp.sat.agoat.AccessControl1ViewActivity}
mFocusedApp=AppWindowToken{b8bff17 token=Token{77c6796 ActivityRecord{30936b1 u0 owasp.sat.agoat/.AccessControl1ViewActivity t195}}}

```

Logcat registró el lanzamiento directo:

```

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

```

La captura de pantalla almacenada en `/workspace/deepLink_accesscontrol1.png` proporciona evidencia visual de la actividad lanzada directamente mediante un enlace profundo sin verificación de PIN.

Recomendación

Mitigación inmediata

Aplicar `FLAG_SECURE` primero en las actividades de mayor riesgo (verificación de PIN, entrada de credenciales, generación de OTP):

```

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

```

Corrección permanente

Crear una clase base de actividad utilizada por todas las pantallas sensibles para centralizar la protección de ventanas:

```
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
        );
    }
}
```

Registrar la clase base en `AndroidManifest.xml` para todas las actividades sensibles:

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

Verificación

1. Ejecutar `adb shell screencap /sdcard/test.png` mientras cada actividad sensible esté en primer plano. Verificar que la imagen capturada se muestre en negro.
2. Ejecutar `adb shell screenrecord /sdcard/test.mp4` y confirmar que la grabación excluye el contenido de la actividad protegida.
3. Revisar la salida de `dumpsys window` para verificar que `FLAG_SECURE` (0x00002000) aparezca en los atributos de la ventana.

MEDIO Flags inseguros de backup y debug exponen el sandbox de la aplicación

El manifiesto de la aplicación habilita `android:debuggable="true"` y `android:allowBackup="true"`, lo que permite a cualquier usuario con acceso a la depuración USB extraer, modificar y restaurar todos los datos privados de la aplicación o adjuntar un depurador al proceso en ejecución.

Estas flags del manifiesto eliminan protecciones críticas del sandbox. `android:allowBackup="true"` permite la extracción completa de datos y su manipulación mediante `adb backup` y `adb restore`, mientras que `android:debuggable="true"` permite que usuarios de shell asuman el UID de la aplicación mediante `run-as` y adjunten depuradores JDWP. En conjunto, ofrecen múltiples vías triviales para sobrescribir datos de autenticación e inspeccionar la memoria en tiempo de ejecución.

Detalles técnicos

Hallazgo #1

Causa raíz

El manifiesto de la aplicación declara `android:debuggable="true"` y `android:allowBackup="true"`. Estos flags permiten que cualquier usuario con acceso a depuración USB extraiga todos los datos privados de la aplicación mediante `adb backup`, restaure datos alterados mediante `adb restore` y ejecute comandos como el UID de la aplicación a través de `run-as`. Esto otorga acceso de lectura/escritura sin restricciones a SharedPreferences, bases de datos SQLite y otros archivos del sandbox.

Evidencia de explotación

[Análisis estático] `AndroidManifest.xml` contiene ambos flags:

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

Mediante `run-as`, un atacante puede escribir directamente en el sandbox de la aplicación:

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

Mediante `adb backup`, un atacante puede extraer todos los datos de la aplicación, modificar archivos sin conexión y restaurar el archivo alterado:

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

Evidencia de validación

Causa raíz

La validación amplió la causa raíz más allá de los propios flags del manifiesto para incluir la ausencia de controles de integridad en la capa de almacenamiento y la presencia de almacenes de datos de alto valor en claro. El análisis estático confirmó que `android:fullBackupContent` no está presente en `AndroidManifest.xml`, lo que implica que el alcance del backup predeterminado abarca todo el directorio `/data/data/owasp.sat.agoat/`. Además, se determinó que toda la lógica de persistencia de datos sensibles carece de cifrado, HMAC o firmas digitales, lo que provoca que la aplicación acepte silenciosamente archivos del sandbox alterados durante la restauración.

Código vulnerable

La validación identificó cinco objetivos concretos residentes en el sandbox creados por componentes específicos de la aplicación:

1. `pinDetails.xml` — Almacenamiento de PIN en MD5 sin sal `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();
}
```

El método `hashPIN()` utiliza `MessageDigest.getInstance("MD5")` (equivalente a la línea ~113), generando un digest criptográficamente inseguro sin sal.

2. `users.xml` — Almacenamiento de credenciales en texto plano `InsecureStorageSharedPrefs.java:50-57` :

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

3. Base de datos SQLite `aGoat` — Repositorio de credenciales en texto plano `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` — Fuga de credenciales en archivos temporales de lectura global `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` — Estado de juego de solo integridad `InsecureStorageSharedPrefs1Activity.java:91-95` :

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

Evidencia de explotación

La validación confirmó los flags del manifiesto y construyó cadenas de explotación con referencias al código fuente:

Confirmación estática del manifiesto:

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

No está presente el atributo `android:fullBackupContent`, y la búsqueda de expresiones regulares en `res/xml/` no devolvió ninguna regla de exclusión de backup.

Cadena de `adb backup` / `restore` validada:

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

El análisis estático post-restauración confirmó que `AccessControlIssue1Activity.java:72-88` lee `pinDetails.xml` sin controles de integridad. El método `isPinCorrect()` compara los PIN proporcionados por el usuario contra el digest MD5 almacenado; inyectar `5f4dcc3b5aa765d61d8327deb882cf99` (MD5 de "password") permite autenticarse con "password".

Cadena de lectura/escritura con `run-as` validada:

```
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'
```

Los comandos de escritura para inyectar datos del atacante también fueron validados:

```
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"
```

Registro de credenciales: La validación inicializó un par PIN/hash controlado por el atacante:

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

Recomendación

Corrección inmediata

Eliminar `android:debuggable="true"` y establecer `android:allowBackup="false"` en `AndroidManifest.xml`:

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

Corrección permanente

Asegurar que los builds de release nunca se distribuyan con `debuggable="true"`. Implementar una revisión lint en tiempo de compilación o una puerta de CI/CD que falle si cualquiera de las flags está habilitada en variantes de release. Si los backups son necesarios por comodidad del usuario, implementar el mecanismo de backup cifrado de Android (AutoBackup con una clave del lado del cliente) en lugar del modelo heredado de backup completo.

Verificación

1. Compilar el APK de release y ejecutar:

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

Confirmar que ninguno de los atributos esté presente o que ambos estén establecidos explícitamente en `false`.

2. Intentar `adb backup -f test.ab owasp.sat.agoat` y verificar que falle o genere un archivo vacío.

3. Intentar `adb shell run-as owasp.sat.agoat` y verificar que el comando sea rechazado.

MEDIO Credenciales de autenticación embebidas en el código fuente de la aplicación

Múltiples secretos hardcodeados —incluyendo un nombre de usuario, una contraseña y una clave de autenticación— están embebidos directamente en el código fuente de la aplicación y expuestos en tiempo de ejecución a través de un `BroadcastReceiver` exportado.

Las credenciales hardcodeadas socavan la seguridad de cualquier sistema que dependa del secreto para la autenticación o la autorización. Dado que `ShowDataReceiver` está exportado, una aplicación maliciosa de terceros puede filtrar estos secretos en tiempo de ejecución sin interacción del usuario. Si estas credenciales se reutilizan en distintos entornos backend, representan una vía directa al acceso no autorizado.

Detalles técnicos

Hallazgo #1

Causa raíz

`ShowDataReceiver.java` y `HardCodeActivity.java` contienen cadenas literales con credenciales de autenticación y secretos que se compilan en el APK. `ShowDataReceiver` está exportado, lo que permite que cualquier aplicación del dispositivo active un `Toast` que filtra estos valores en tiempo de ejecución.

Código vulnerable

`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>
```

Evidencia de explotación

La descompilación estática del APK confirmó los valores exactos sin necesidad de ejecución dinámica:

Secreto	Valor	Ubicación en el código fuente
Nombre de usuario	CrazyUser	ShowDataReceiver.java
Contraseña	CrazyPassword	ShowDataReceiver.java
Clave	123myKey456	ShowDataReceiver.java
Código promocional	NEW2019	HardCodeActivity.java
URL de Firebase	https://androgoat-42597.firebaseio.com	res/values/strings.xml

Estos valores se pueden recuperar fácilmente mediante ingeniería inversa del APK (JADX/apktool) o enviando un broadcast Intent al `ShowDataReceiver` exportado, que muestra las credenciales en un mensaje `Toast`.

Evidencia de validación

Causa raíz

El APK compila literales de credenciales en texto plano directamente en el bytecode DEX. `ShowDataReceiver` se declara en `AndroidManifest.xml` como `android:exported="true"` sin requerimiento de permiso, lo que crea un punto de entrada desprotegido al que puede acceder cualquier otra aplicación instalada.

Código vulnerable

Código fuente descompilado de `/workspace/my_source/sources/owasp/sat/agoat/ShowDataReceiver.java` :

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

`HardCodeActivity.java` almacena el código promocional como un literal compilado utilizado en una comparación local:

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

Declaración del manifiesto que confirma que el receiver está exportado y desprotegido:

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

Evidencia de explotación

Se confirmó la explotación dinámica enviando un broadcastIntent mediante ADB:

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

Salida de la herramienta:

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

El broadcast se entregó correctamente (`result=0`), lo que demuestra que cualquier aplicación de terceros puede activar el Toast que filtra las credenciales sin interacción del usuario ni permisos.

Una búsqueda exhaustiva con `ripgrep` en toda la base de código confirmó que los secretos existen exclusivamente en las ubicaciones reportadas y **no tienen sinks de backend**:

Secreto	Archivos encontrados	Uso fuera de Toast / UI
<code>CrazyUser</code>	<code>ShowDataReceiver.java:17</code>	Ninguno
<code>CrazyPassword</code>	<code>ShowDataReceiver.java:17</code>	Ninguno
<code>123myKey456</code>	<code>ShowDataReceiver.java:17</code>	Ninguno
<code>NEW2019</code>	<code>HardCodeActivity.java:49</code>	Ninguno (solo <code>String.equals</code> local)
<code>androgoat-42597</code>	<code>res/values/strings.xml:130</code>	Ninguno (nunca se carga en el código)

[Prueba externa] Aunque no se usa en el código de la aplicación (no hay importaciones del SDK de Firebase ni referencias a `R.string.firebase_url`), la URL de Firebase embebida en el código `https://androgoat-42597.firebaseio.com` respondió a una solicitud HTTP no autenticada con `200 OK` y cuerpo `{"password":"androgoat","user":"androgoat"}`, confirmando que el endpoint es de lectura pública.

Recomendación

Inmediato

- Rote todas las credenciales hardcodeadas de inmediato; trate cualquier valor incrustado en el APK como público.
- Elimine o deshabilite el `ShowDataReceiver` exportado si no es necesario para la funcionalidad legítima de la aplicación.

Corrección permanente

- Cargue los secretos desde un servidor remoto de configuración seguro autenticado mediante atestación de dispositivo, o recupérellos del Android Keystore o de la biblioteca Tink en tiempo de ejecución.
- Nunca compile contraseñas, claves de API ni tokens de autenticación en el código fuente o los recursos de la aplicación.

Verificación

1. Vuelva a decompilar el APK tras la remediación.
2. Busque en el árbol de código fuente las cadenas conocidas en texto plano (`CrazyUser` , `CrazyPassword` , `123myKey456` , `NEW2019`).
3. Confirme que no existan coincidencias en fuentes Java, fuentes Kotlin, `strings.xml` , `assets/` y `res/raw/` .
4. Verifique que `ShowDataReceiver` ya no esté exportado en `AndroidManifest.xml` .

MEDIO

Vulnerabilidad criptográfica: Algoritmo de hash débil

El cliente utiliza funciones de hash débiles que presentan debilidades criptográficas conocidas.

Una función hash es una función que se utiliza para codificar datos y devolver un valor hash de -tamaño fijo- con el objetivo de minimizar la probabilidad de colisión (la posibilidad de que dos entradas distintas produzcan exactamente el mismo valor hash es infinitesimal). Estas funciones pueden utilizarse para verificar la integridad de un mensaje o de archivos, identificar un archivo o datos, o comprobar un secreto o una contraseña.

Algoritmos de hash comunes:

- **MD5** : Message Digest Algorithm (MD5) fue diseñado en 1991 como sustituto de MD4. En marzo de 2005 se presentaron dos archivos distintos con el mismo hash MD5. Un año después se publicó un algoritmo capaz de encontrar colisiones en menos de un minuto. En cuanto a los algoritmos hash, encontrar dos entradas distintas con el mismo hash debería ser computacionalmente inviable; MD5 incumple este requisito de forma catastrófica. En 2008, el Software Engineering Institute concluyó que MD5 está «roto criptográficamente e inadecuado para su uso posterior».
- **SHA1** : Secure Hash Algorithm 1 (SHA1) fue diseñado por la NSA. Al igual que MD5, SHA1 es vulnerable a ataques de colisión. En 2005 ya se consideraba inseguro; en 2011, el National Institute of Standards and Technology (NIST) lo declaró formalmente obsoleto.
- **SHA2** : es un conjunto de algoritmos hash diseñados también por la NSA. SHA-2 difiere significativamente de SHA1 y se considera altamente seguro.
- **SHA-256** : es un miembro del conjunto SHA-2. El sufijo 256 indica 256 bits, la longitud del valor hash generado. SHA-256 es un estándar de la industria seguro y consolidado, con una probabilidad de colisión prácticamente nula (2^{256}).
- **SHA3** : SHA3 es el miembro más reciente de la familia de estándares Secure Hash Algorithm, publicado por el NIST en 2015. Aunque forma parte de la misma serie de estándares, SHA3 difiere internamente de la estructura tipo MD5 de SHA1 y SHA2, y también se considera altamente seguro.
- **Argon2** : es el ganador de la Password Hashing Competition de 2015 y actualmente se considera uno de los algoritmos de hash de contraseñas más seguros disponibles. Ha sido adoptado por diversos estándares de seguridad y criptografía, entre ellos el Internet Engineering Task Force (IETF), la Password Hashing Competition (PHC) y The Open Worldwide Application Security Project (OWASP).
- **Bcrypt** : Introducido en 1999, Bcrypt es un algoritmo lento y adaptativo diseñado para ser altamente resistente a la fuerza bruta. Su carácter adaptativo otorga al usuario el control sobre el costo computacional, por lo que puede ajustarse a medida que el hardware evoluciona. Bcrypt también ha sido adoptado por numerosos estándares de seguridad, como el Open Web Application Security Project (OWASP) y el Payment Card Industry Data Security Standard (PCI DSS).
- **HMAC** : El código de autenticación de mensajes basado en hash no es un algoritmo hash, sino una técnica criptográfica de autenticación que utiliza un algoritmo hash junto con una clave secreta. La robustez de HMAC depende del algoritmo hash subyacente, por lo que no debe utilizarse con algoritmos hash obsoletos ni débiles.

MD5 y **SHA-1** son inseguros y no deben utilizarse, ni siquiera con HMAC.

Detalles técnicos

Hallazgo #1

La aplicación utiliza MD5. MD2, MD4 y MD5 son funciones hash que han sido rotas por colisiones. Traza de llamadas:

```
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$mperformClickInternal
at android.view.View$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
```

Hallazgo #2

La aplicación utiliza MD5. MD2, MD4 y MD5 son funciones hash que han sido comprometidas por colisiones. Traza de llamadas:

```

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`

No se ha podido extraer el código fuente.

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...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.checkNotNull(v0_2, pinValue);
    if (!v1_10.isPinCorrect(v0_2.getText().toString())) {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Incorrect PIN entered), 1).show();
    } else {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) PIN Verified), 1).show();
        this.this$0.startActivity(new android.content.Intent(((android.content.Context) this.this$0),
owasp.sat.agoat.AccessControl1ViewActivity));
    }
    return;
}

```

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

```

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

```

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

```

public final void onClick(android.view.View p5)
{
    android.widget.EditText v0_2 = ((android.widget.EditText) this.this$0.findViewById(2131165335));
    android.widget.Toast v1_9 = this.this$0;
    kotlin.jvm.internal.Intrinsics.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;
}

```

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;
}

```

Hallazgo #3

La aplicación utiliza MD5. MD2, MD4 y MD5 son funciones hash que han sido vulneradas por colisiones. Traza de llamadas:

```

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.checkParameterIsNotNull(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...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.checkParameterIsNotNull(p4, pinValue);
}

```

```
return this.getSharedPreferences(pinDetails, 0).getString(pin, 0).equals(this.hashPIN(p4));
}
```

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

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

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

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

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;
}
```

Hallazgo #4

La aplicación utiliza MD5. MD2, MD4 y MD5 son funciones hash que han sido comprometidas por colisiones. Traza de llamadas:

```
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
```

en `java.security.MessageDigest.getInstance`

Unable to extract source code.

en `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(MD5).digest(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);
}

```

en `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));
}

```

en `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;
}

```

en `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;
}

```

Hallazgo #5

La aplicación utiliza MD5. MD2, MD4 y MD5 son funciones hash que han sido vulneradas por colisiones. Traza de llamadas:

```

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

```

En `java.security.MessageDigest.getInstance`

Unable to extract source code.

En `owasp.sat.agoat.AccessControlIssue1Activity.hashPIN`

```

public final String hashPIN(String p13)
{
    kotlin.jvm.internal.Intrinsics.checkNotNull(p13, pinValue);
    String v0_6 = java.security.MessageDigest.getInstance(MD5);
}

```

```

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

```

En `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));
}

```

En `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;
}

```

Hallazgo #6

La aplicación utiliza MD5. MD2, MD4 y MD5 son funciones hash que han sido comprometidas por colisiones. Traza de llamadas:

```

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

```

en `java.security.MessageDigest.getInstance`

Unable to extract source code.

en `owasp.sat.agoat.AccessControlIssue1Activity.hashPIN`

```

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

```

en `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));
}
```

Hallazgo #7

La aplicación utiliza MD5. MD2, MD4 y MD5 son funciones hash que han sido rotas por colisiones. Traza de llamadas:

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

en `java.security.MessageDigest.getInstance`

No se pudo extraer el código fuente.

en `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);
}
```

Recomendación

Para mitigar los riesgos de utilizar algoritmos de hash débiles, tenga en cuenta lo siguiente:

- Evite MD5 y SHA-1, ya que son algoritmos inseguros y propensos a colisiones. No deben utilizarse para ningún propósito, incluido HMAC.
- Para el hash y la verificación de firmas, opte por SHA-256 o SHA-3. Ofrecen alta seguridad y son estándares de la industria ampliamente aceptados.
- Para el hash de datos sensibles (PII, PHI...), considere Argon2 o Bcrypt. Están diseñados para resistir ataques de fuerza bruta y son recomendados por los estándares de seguridad.
- Al utilizar HMAC para la autenticación de mensajes, asegúrese de combinarlo con algoritmos de hash seguros como SHA-256 o SHA-3 para mantener una seguridad robusta.

Referencias

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

MEDIO

Solicitud HTTP en texto plano

CVSS v3: 9,1

La aplicación está accediendo a datos a través de un canal no cifrado, comprometiendo la confidencialidad y la integridad de los datos en tránsito.

El tráfico HTTP en claro se refiere a los datos transmitidos entre una aplicación móvil y su servidor backend sin ningún tipo de cifrado, lo que los hace fácilmente legibles y susceptibles de ser interceptados por actores maliciosos. Esta ausencia de cifrado conlleva riesgos de seguridad significativos para las aplicaciones móviles. Cuando se envía información sensible, como credenciales de inicio de sesión, datos personales o detalles financieros, a través de conexiones HTTP en claro, esta se vuelve vulnerable a la interceptación de comunicaciones y a ataques de tipo man-in-the-middle.

Uno de los principales riesgos asociados al tráfico HTTP en claro en aplicaciones móviles es la posible exposición de los datos de los usuarios. Los atacantes pueden interceptar y extraer información valiosa, lo que puede derivar en robo de identidad, acceso no autorizado a cuentas y uso indebido de datos personales. Además, los atacantes pueden explotar estas vulnerabilidades para manipular el contenido de la aplicación o inyectar código malicioso en el canal de comunicación.

Asimismo, el tráfico HTTP en claro puede comprometer la privacidad de los usuarios. Las aplicaciones móviles suelen recopilar y transmitir datos de los usuarios con fines analíticos o de publicidad dirigida. Sin cifrado, estos datos son fácilmente accesibles para terceros, comprometiendo la confidencialidad de los usuarios y pudiendo derivar potencialmente en violaciones de privacidad.

Las aplicaciones móviles que no implementan protocolos de comunicación seguros también enfrentan el riesgo de ataques de suplantación de identidad. Los atacantes pueden crear redes Wi-Fi falsas o utilizar otras técnicas para interceptar el tráfico, haciéndose pasar por el servidor backend legítimo. Los usuarios

envían sus datos al atacante sin darse cuenta, quien puede utilizarlos con fines maliciosos.

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();

    // Simular una solicitud HTTP para iniciar sesión.
    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) {
      // Inicio de sesión exitoso
      setState(() {
        _loginStatus = 'Login successful!';
      });
    } else {
      // Inicio de sesión fallido
      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
    }

    // Simular una solicitud HTTP para iniciar sesión.
    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()  
  
            // Simular una solicitud HTTP para iniciar sesión.  
            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."  
            }  
        }  
    }  
}
```

Detalles técnicos

Hallazgo #1

Uso de una URL HTTP en texto plano sin cifrar: <http://demo.testfire.net/> Solicitud:

```
GET / HTTP/1.1  
Host: demo.testfire.net  
Connection: Keep-Alive
```

```
Accept-Encoding: gzip
User-Agent: okhttp/3.8.0
```

Respuesta:

```
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
```

Recomendación

Para mitigar los riesgos asociados al tráfico HTTP en claro en aplicaciones móviles, es imperativo implementar protocolos de comunicación seguros y buenas prácticas. A continuación se presentan algunas recomendaciones detalladas para reforzar la seguridad de las comunicaciones de la aplicación móvil:

- Adoptar el cifrado HTTPS: El paso más fundamental consiste en utilizar HTTPS (HTTP Secure) para toda la comunicación entre la aplicación móvil y el servidor backend. HTTPS cifra los datos durante la transmisión, garantizando que la información sensible permanezca confidencial y segura. Obtenga un certificado SSL/TLS de una Autoridad de Certificación (CA) de confianza y exija conexiones HTTPS para todos los endpoints de API y transferencias de datos.
- Certificate Pinning: Implemente certificate pinning para reforzar la seguridad de las conexiones HTTPS. Esto consiste en incluir de forma rígida el certificado SSL del servidor o su clave pública dentro de la aplicación móvil. De este modo, la app garantiza que solo se comunica con el servidor previsto, previniendo ataques man-in-the-middle en los que los atacantes intentan utilizar sus propios certificados.
- Habilitar HSTS (HTTP Strict Transport Security): Utilice encabezados HSTS para indicar al navegador web de la app que siempre use conexiones HTTPS, incluso si el usuario escribe erróneamente «http://» en la URL. Esto previene posibles ataques de degradación y garantiza que todas las comunicaciones estén cifradas.
- Utilizar la última versión de TLS: Asegúrese de que la aplicación móvil utilice la última versión del protocolo TLS (Transport Layer Security) para aprovechar las funciones de seguridad y los algoritmos de cifrado más robustos.
- Implementar la comprobación de certificados: Configure la app para verificar certificados SSL revocados o caducados y evitar la interacción con servidores potencialmente comprometidos.
- Cifrado de datos sensibles: Cifre los datos sensibles almacenados en el dispositivo o transmitidos al servidor. Utilice algoritmos de cifrado robustos y gestione las claves de cifrado de forma segura; esto puede añadir una capa adicional de seguridad a la transmisión de datos.
- Garantizar la transmisión segura de datos para todas las API: Evalúe todas las API utilizadas por la app y asegúrese de que también empleen HTTPS para la transmisión de datos. Esto incluye las API y servicios de terceros integrados en la aplicación.

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();

    // Simular una petición HTTPS segura para inicio de sesión (reemplace esto por el endpoint seguro real de su API).
```

```

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) {
  // Inicio de sesión exitoso
  setState(() {
    _loginStatus = 'Login successful!';
  });
} else {
  // Inicio de sesión fallido
  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
    }
  }
}

```

```

// Simular una petición HTTPS segura para inicio de sesión (reemplace esto por el endpoint seguro real de su API).
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()

            // Simular una petición HTTPS segura para inicio de sesión (reemplace esto por el endpoint seguro real de su API).
            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."
        }
    }
}
}
}

```

Detalles del vector CVSS

Vector CVSS V3:

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

Vector de ataque:	Red (N) Adyacente (A) Local (L) Físico (P)
Complejidad de ataque:	Bajo (L) Alto (H)
Privilegios requeridos:	Ninguno (N) Bajo (L) Alto (H)
Interacción del usuario:	Ninguno (N) Requerido (R)
Alcance:	Cambiado (C) Sin cambios (U)
Confidencialidad:	Alto (H) Bajo (L) Ninguno (N)
Integridad:	Alto (H) Bajo (L) Ninguno (N)
Disponibilidad:	Alto (H) Bajo (L) Ninguno (N)

Referencias

- [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 \(CWE-319\)](#)

MEDIO

CERTIFICATE_EXPIRED

Un certificado SSL/TLS caducado ya no es confiable para navegadores y clientes, lo que provoca advertencias de seguridad, interrupciones de servicio y vulnerabilidades potenciales, incluidos ataques man-in-the-middle.

Un certificado SSL/TLS caducado ya no es confiable para los navegadores y clientes modernos, lo que genera roturas en las cadenas de confianza, advertencias de seguridad y exposición a diversas vulnerabilidades potenciales, como ataques man-in-the-middle (MitM).

Cuando un certificado caduca, los navegadores muestran advertencias de seguridad destacadas como «Tu conexión no es privada» o «El certificado ha caducado». Estas advertencias interrumpen la comunicación segura entre el cliente y el servidor, afectando directamente la confianza del usuario y la continuidad del negocio. En entornos de comercio electrónico y otros sensibles, esto suele provocar abandono de usuarios, pérdidas financieras y daños reputacionales.

Los riesgos de seguridad de un certificado caducado trascienden el impacto en los usuarios. Los atacantes pueden aprovechar la ausencia de un certificado válido para ejecutar ataques man-in-the-middle (MitM), interceptando o alterando datos sensibles durante la transmisión. Sin un certificado válido que autentique el servidor, las comunicaciones cifradas pierden su protección, dejando los datos vulnerables a la interceptación, modificación o falsificación.

Por ejemplo, un sitio web con un certificado caducado ya no puede garantizar la confidencialidad y la integridad de los datos del usuario. Un atacante puede suplantar el sitio web e interceptar credenciales de acceso o información de pago mediante spoofing de la conexión, lo que puede derivar en acceso no autorizado o fraude financiero.

Los certificados caducados también representan incumplimiento de diversos estándares de seguridad, incluidos PCI DSS, HIPAA y GDPR. Estas regulaciones exigen certificados SSL/TLS válidos para proteger los datos sensibles. El incumplimiento puede acarrear sanciones legales y daños reputacionales adicionales.

Detalles técnicos

Hallazgo #1

CERTIFICATE_EXPIRED detectado en `demo.testfire.net`

- **Hallazgo:** expira en <60 días (33)

Recomendación

Para mitigar los riesgos asociados a certificados SSL/TLS vencidos, las organizaciones deben implementar diversas estrategias proactivas:

- **Herramientas de monitoreo automatizado:** Despliegue herramientas que supervisen continuamente el estado de los certificados SSL/TLS y envíen alertas cuando estén próximos a vencer. Esto ayuda a prevenir interrupciones del servicio causadas por certificados vencidos que pasan desapercibidos.
- **Procedimientos de renovación de emergencia:** Establezca procedimientos de emergencia claros para la renovación rápida de certificados en caso de vencimiento inesperado. Esto garantiza un tiempo de inactividad mínimo y protege contra posibles riesgos de seguridad.
- **Auditorías periódicas:** Realice auditorías periódicas del inventario de certificados SSL/TLS para identificar y renovar oportunamente cualquier certificado vencido. Mantener un inventario actualizado ayuda a evitar lagunas de seguridad.
- **Aprovisionamiento automatizado de certificados:** Implemente sistemas que automaticen la emisión y renovación de certificados SSL/TLS para mantener la validez continua. Herramientas como cert-manager en Kubernetes agilizan este proceso.

Aprovisionamiento automatizado de certificados en Kubernetes:

En Kubernetes, puede automatizar la gestión de certificados SSL/TLS mediante cert-manager. Esta herramienta interactúa con Certificate Authorities (CAs) como Let's Encrypt para emitir y renovar certificados automáticamente.

- Instale cert-manager:

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

A continuación se muestra una configuración YAML de cert-manager para automatizar el aprovisionamiento de certificados:

```
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
```

Monitoreo con Certbot:

Para entornos que no utilizan Kubernetes, puede usar Certbot para automatizar la emisión y renovación de certificados SSL.

```
# Automatically issue or renew SSL certificates using Certbot
domain="example.com"
email="admin@example.com"

# Run Certbot in standalone mode to obtain the certificate
certbot certonly --standalone -d $domain --email $email --agree-tos
```

Renovación automatizada de certificados con cronjob:

Puede automatizar el proceso de renovación mediante un cronjob:

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

Este cronjob se ejecutará diariamente a medianoche para verificar los certificados que deban renovarse, garantizando cobertura continua sin intervención manual.

Referencias

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

MEDIO

Configuración insegura de encabezados HTTP

El encabezado de la aplicación está ausente o establece valores de encabezado inseguros o no seguros.

Configuración insegura de cabeceras:

- **Content Security Policy:** Mitiga el riesgo de cross-site scripting (XSS) al especificar orígenes de contenido confiables.
- **Cookie:** Mejora la seguridad estableciendo atributos como HttpOnly y Secure para proteger los datos de las cookies contra accesos no autorizados.
- **Cross-Origin Resource Sharing:** Controla cómo pueden compartirse los recursos entre distintos dominios para evitar accesos maliciosos.
- **HTTP Public Key Pinning:** Protege contra ataques man-in-the-middle al especificar qué claves públicas son válidas para un sitio determinado.
- **Redirection:** Garantiza que las redirecciones sean seguras y solo conduzcan a destinos de confianza para prevenir vulnerabilidades de open redirect.
- **Referrer Policy:** Define cuánta información de referrer se transmite al navegar de un sitio a otro, mejorando la privacidad.
- **Subresource Integrity:** Verifica que los recursos cargados desde dominios de terceros no hayan sido manipulados mediante la comprobación de su hash criptográfico.
- **X-Content-Type-Options:** Evita que los navegadores realicen MIME-sniffing del tipo de contenido, reduciendo el riesgo de ataques basados en el tipo de contenido.
- **X-Frame-Options:** Previene ataques de clickjacking controlando si una página puede incrustarse en un frame.
- **X-XSS-Protection:** Activa el filtro XSS integrado del navegador para bloquear ataques de cross-site scripting detectados.
- **Permissions-Policy:** Controla qué funciones y API pueden utilizarse en el navegador, reforzando la seguridad al limitar las capacidades disponibles para contenido no confiable.
- **Clear-Site-Data:** Permite que los sitios soliciten al navegador el borrado de datos almacenados (cookies, almacenamiento local, cachés) para un origen específico, ayudando a mitigar el impacto de fugas de datos o problemas de privacidad.

Detalles técnicos

Hallazgo #1

MISSING_SECURITY_HEADERS detectado en `demo.testfire.net`

- Hallazgo: --

Recomendación

Para asegurarse de no tener una configuración insegura de cabeceras, tenga en cuenta lo siguiente:

1. Content Security Policy (CSP):

Restrinja las fuentes de contenido para mitigar los riesgos asociados a ataques de cross-site scripting (XSS) y la carga no autorizada de recursos.

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

2. Cabeceras de seguridad de cookies:

Implemente los atributos Secure y HttpOnly para evitar el robo y la manipulación de cookies, mejorando la seguridad de las sesiones de usuario.

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

3. Cross-Origin Resource Sharing (CORS):

Configure correctamente las políticas CORS para restringir el acceso a recursos desde orígenes diferentes, mitigando el cross-site request forgery (CSRF) y la fuga de datos cross-origin.

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

4. HTTP Public Key Pinning (HPKP):

Utilice HPKP para vincular claves públicas a servidores web específicos y protegerse contra ataques Man-in-the-Middle (MitM) con certificados fraudulentos.

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

5. Cabeceras de redirección:

Garantice redirecciones seguras implementando controles estrictos que prevengan vulnerabilidades de open redirection, protegiendo así a los usuarios contra ataques de phishing.

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

6. ReferrerPolicy:

Establezca políticas de referrer apropiadas para controlar cuánta información se envía en la cabecera Referer, reduciendo el riesgo de exposición de datos sensibles.

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

7. Subresource Integrity (SRI):

Implemente SRI para verificar la integridad de recursos externos, como scripts y hojas de estilo, protegiéndose contra modificaciones no autorizadas y ataques a la cadena de suministro.

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

8. X-Content-Type-Options:

Habilite la directiva 'nosniff' para evitar que los navegadores realicen MIME-sniffing en una respuesta, mitigando los riesgos asociados a ataques de confusión de tipos de contenido.

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

9. X-Frame-Options:

Configure X-Frame-Options para restringir la incrustación de contenido web en frames, protegiendo contra ataques de clickjacking y garantizando la integridad de las páginas web.

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

10. X-XSS-Protection:

Habilite los mecanismos de protección XSS para mitigar ataques XSS indicando a los navegadores que saneen o bloqueen scripts potencialmente maliciosos.

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

Referencias

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

MEDIO

Configuración insegura de cabecera HTTP: HTTP Strict Transport Security (HSTS)

Configuración insegura de encabezado HTTP: HTTP Strict Transport Security (HSTS).

HTTP Strict Transport Security (HSTS) es un mecanismo de política de seguridad web mediante el cual un servidor web declara que los agentes de usuario conformes (como un navegador web) deben interactuar con él utilizando únicamente conexiones seguras (HTTPS). El servidor comunica la política HSTS al agente de usuario mediante un campo de encabezado de respuesta HTTP denominado "Strict-Transport-Security". La política de HSTS especifica un período durante el cual el agente de usuario debe acceder al servidor únicamente de forma segura.

Cuando una aplicación web emite una política HSTS a los agentes de usuario, los agentes de usuario conformes se comportan de la siguiente manera:

- Convierten automáticamente cualquier enlace inseguro (HTTP) que haga referencia a la aplicación web en enlaces seguros (HTTPS). (Por ejemplo, <http://example.com/some/page/> se modificará a <https://example.com/some/page/> antes de acceder al servidor.)
- Si no se puede garantizar la seguridad de la conexión (p. ej., el certificado TLS del servidor es autofirmado), los agentes de usuario muestran un mensaje de error y no permiten que el usuario acceda a la aplicación web.

Detalles técnicos

Hallazgo #1

HSTS detectado en `demo.testfire.net`

- **Hallazgo:** no ofrecido

Recomendación

Configure el servidor web para redirigir las solicitudes HTTP a HTTPS.

Por ejemplo, para Apache, debe modificar el archivo httpd.conf. Para más configuraciones, consulte la sección Referencias externas.

```
# 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>
```

Referencias

- [Strict-Transport-Security](#)

MEDIO

Cifrados TLS inseguros admitidos

El cliente admite combinaciones de cipher suites con debilidades criptográficas conocidas.

Transport Layer Security (TLS) comienza cuando el cliente solicita una conexión segura y presenta una lista de suites de cifrado compatibles.

Una suite de cifrado es un conjunto de algoritmos de autenticación, cifrado, código de autenticación de mensajes (MAC) e intercambio de claves. Cada algoritmo cumple una función específica durante la negociación del protocolo. Por ejemplo, el uso de un algoritmo débil puede comprometer de forma crítica la seguridad de todo el canal.

Se ha identificado que el endpoint admite una combinación de suites de cifrado y protocolos Secure Sockets Layer / Transport Layer Security (SSL/TLS) que presentan debilidades criptográficas conocidas. Por tanto, no debería confiarse en ellos para garantizar una seguridad efectiva de la capa de transporte. Un atacante capaz de interceptar la conexión podría influir en el tráfico o descifrarlo.

Se han demostrado debilidades criptográficas en todas las configuraciones siguientes:

- Soporte del protocolo SSL obsoleto (v2 o v3)
- Soporte de suites de cifrado que ofrecen claves de menos de 128 bits
- Soporte de suites de cifrado que ofrecen el modo `CBC` en combinación con un protocolo TLS inferior a la versión 1.1, vulnerable al ataque BEAST
- Uso de compresión en TLS vulnerable al ataque CRIME
- Soporte de suites de cifrado que ofrecen `RC4` como cifrado
- Soporte de suites de cifrado que ofrecen `DES` como cifrado

- Soporte de suites de cifrado que ofrecen **MD5** como algoritmo de firma

Detalles técnicos

Hallazgo #1

cipherlist_OBSOLETEdetectado en **demo.testfire.net**

- **Hallazgo:** suites de cifrado obsoletas ofrecidas

Hallazgo #2

cipherlist_OBSOLETEdetectado en **owasp.org**

- **Hallazgo:** Suites de cifrado obsoletas ofrecidas

Recomendación

La configuración TLS recomendada debe cumplir con las siguientes recomendaciones:

- Restringir a **TLSv1** o superior, siendo **TLS1.2** el preferido
- Si se requiere **SSLv3**, se recomienda implementar la función **TLS Fallback SCSV** para prevenir ataques de downgrade de protocolo
- Deshabilitar Diffie-Hellman anónimo (**ADH**)
- Deshabilitar las suites de cifrado **aNULL** y **eNULL**
- Deshabilitar las suites de intercambio de claves de exportación
- Eliminar el soporte de **RC4**
- Eliminar el soporte de **DES**
- Eliminar el soporte de **MD5**
- Preferir **SHA256** sobre **SHA1**
- Preferir **AES128** sobre **AES256**, ya que 256 ofrece pocas ventajas de seguridad y es menos robusto ante ataques de temporización
- Deshabilitar la renegociación iniciada por el cliente
- Deshabilitar la compresión TLS
- Ofrecer únicamente cifrados con una longitud de clave superior a 128 bits
- Ofrecer suites de cifrado con propiedades de secreto directo perfecto (**DHE**, **ECDHE**)
- Usar un grupo Diffie-Hellman personalizado para protegerse contra el ataque Logjam
- Implementar el campo de encabezado HTTP Strict Transport Security
- Implementar **OCSP stapling**

Ejemplo de una configuración TLS segura para Nginx generada con [Mozilla SSL Configuration Generator](#)

<> NGINX

```
server {
    listen 443 ssl;

    # certs sent to the client in SERVER HELLO are concatenated in ssl_certificate
    ssl_certificate /path/to/signed_cert_plus_intermediates;
    ssl_certificate_key /path/to/private_key;
    ssl_session_timeout 1d;
    ssl_session_cache shared:SSL:50m;
    ssl_session_tickets off;

    # Diffie-Hellman parameter for DHE ciphersuites, recommended 2048 bits
    ssl_dhparam /path/to/dhparam.pem;

    # modern configuration. tweak to your needs.
    ssl_protocols TLSv1.1 TLSv1.2;
    ssl_ciphers 'ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!3DES:!MD5:!PSK';
    ssl_prefer_server_ciphers on;

    # HSTS (ngx_http_headers_module is required) (15768000 seconds = 6 months)
    add_header Strict-Transport-Security max-age=15768000;

    # OCSP Stapling ---
    # fetch OCSP records from URL in ssl_certificate and cache them
    ssl_stapling on;
    ssl_stapling_verify on;
```

```
## verify chain of trust of OCSP response using Root CA and Intermediate certs
ssl_trusted_certificate /path/to/root_CA_cert_plus_intermediates;

resolver <IP DNS resolver>;

....
}
```

Referencias

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

MEDIO TLS_FALLBACK_SCSV no soportado

El servidor no admite TLS_FALLBACK_SCSV, lo que lo deja vulnerable a ataques de downgrade de protocolo.

Esta vulnerabilidad indica que el servidor no admite **TLS_FALLBACK_SCSV** (Signaling Cipher Suite Value), un mecanismo diseñado para prevenir ataques de downgrade de protocolo en conexiones SSL/TLS.

TLS_FALLBACK_SCSV es fundamental porque:

1. Permite que los clientes indiquen cuándo retroceden a una versión de protocolo inferior debido a fallos de conexión.
2. Permite que los servidores detecten e impidan intentos de downgrade maliciosos.
3. Ayuda a mantener el mayor nivel de seguridad posible en conexiones SSL/TLS.

Cuando no se admite **TLS_FALLBACK_SCSV**, el servidor se vuelve vulnerable a:

1. Ataques de downgrade de protocolo: un atacante puede forzar el uso de versiones de protocolo más antiguas y menos seguras.
2. Ataques Man-in-the-Middle (MitM): el downgrade a un protocolo más débil puede facilitar que los atacantes intercepten y descifren las comunicaciones.
3. Explotación de vulnerabilidades en protocolos antiguos: las versiones anteriores de SSL/TLS pueden contener vulnerabilidades conocidas explotables una vez producido el downgrade.

Escenario de ejemplo:

Un atacante intercepta el handshake inicial entre un cliente y un servidor. El atacante manipula la conexión para que falle, provocando que el cliente reintente con una versión de protocolo inferior. Sin **TLS_FALLBACK_SCSV**, el servidor acepta este downgrade, lo que puede exponer la conexión a vulnerabilidades de la versión de protocolo anterior. La ausencia de soporte para **TLS_FALLBACK_SCSV** contraviene las mejores prácticas de seguridad SSL/TLS y puede afectar al cumplimiento de diversos estándares y normativas de seguridad.

Detalles técnicos

Hallazgo #1

fallback_SCSV detectado en **demo.testfire.net**

- **Hallazgo:** NO soportado

Recomendación

Cuando no se admite TLS_FALLBACK_SCSV (Signaling Cipher Suite Value), puede dar lugar a ataques de downgrade. A continuación se presentan estrategias para mitigar este riesgo:

Estrategias proactivas

1. **Deshabilitar el fallback de versiones SSL/TLS:** Configure los servidores para deshabilitar el fallback a versiones anteriores de SSL/TLS. Esto evita que los atacantes obliguen a las conexiones a utilizar protocolos vulnerables.
2. **Exigir una versión mínima de TLS:** Establezca una versión mínima aceptable de TLS (por ejemplo, TLS 1.2) tanto en el cliente como en el servidor para evitar el downgrade a versiones vulnerables.
3. **Escaneos de seguridad periódicos:** Realice escaneos periódicos de su infraestructura para identificar y corregir cualquier debilidad en la configuración de TLS.
4. **Monitorear patrones inusuales de negociación TLS:** Implemente monitoreo para detectar intentos de forzar el downgrade de protocolos.

Ejemplos de implementación

1. Configuración de Nginx

Para deshabilitar el fallback de versiones SSL/TLS y exigir una versión mínima de TLS en Nginx:

```
server {
    listen 443 ssl;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;
    ssl_ciphers 'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256';
    # Other SSL settings...
}
```

2. Configuración de Apache

Para Apache, utilice la siguiente configuración para obtener resultados similares:

```
<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. Comando de OpenSSL para pruebas

Utilice OpenSSL para probar la configuración TLS de su servidor:

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

Este comando intenta establecer una conexión TLS 1.2. Si tiene éxito, indica que su servidor está configurado correctamente para utilizar versiones modernas de TLS.

Monitoreo de intentos de downgrade

- Uso de fail2ban

Puede utilizar `fail2ban` para monitorear los registros en busca de intentos potenciales de downgrade:

1. Cree un filtro personalizado en `/etc/fail2ban/filter.d/tls-downgrade.conf` :

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

2. Agregue un jail en `/etc/fail2ban/jail.local` :

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

3. Reinicie fail2ban:

```
sudo systemctl restart fail2ban
```

Esta configuración bloqueará las direcciones IP que realicen intentos repetidos de conexión mediante versiones no admitidas (probablemente antiguas) de SSL/TLS.

Al implementar estas medidas, puede reducir significativamente el riesgo de ataques de downgrade de TLS, incluso cuando no se admite TLS_FALLBACK_SCSV.

Referencias

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

Clave criptográfica y algoritmo de firma débiles en certificado SSL/TLS

El certificado SSL/TLS utiliza tanto una clave criptográfica débil como un algoritmo de firma débil, lo cual podría comprometer la seguridad de las conexiones cifradas y permitir la falsificación de certificados u otros ataques criptográficos.

Los certificados SSL/TLS con claves criptográficas o algoritmos de firma débiles conllevan riesgos de seguridad significativos, ya que dejan los sistemas vulnerables a ataques criptográficos.

Puntos clave sobre claves criptográficas y algoritmos de firma débiles:

1. Claves criptográficas débiles:

- Claves RSA: las claves RSA de menos de 2048 bits se consideran débiles. La recomendación actual es utilizar al menos 2048 bits, con 3072 bits o más para una seguridad a largo plazo.
- Claves DSA: al igual que las claves RSA, las claves DSA de menos de 2048 bits se consideran débiles y deben evitarse.
- Avances computacionales: a medida que aumenta la capacidad computacional, las claves de menor longitud se vuelven cada vez más vulnerables a los ataques.
- Estándares del sector: muchos estándares del sector y requisitos de cumplimiento exigen longitudes mínimas de clave para los certificados SSL/TLS.

2. Algoritmos de firma débiles:

- Algoritmos obsoletos: MD5 y SHA1 se consideran criptográficamente inseguros y no deben utilizarse.
- Ataques de colisión: los algoritmos débiles son vulnerables a ataques de colisión, en los que un atacante puede crear un certificado fraudulento con la misma firma que uno legítimo.
- Estándares del sector: muchos estándares del sector y requisitos de cumplimiento prohíben el uso de algoritmos de firma débiles.
- Compatibilidad retroactiva: algunos sistemas pueden seguir utilizando algoritmos débiles por compatibilidad retroactiva, pero esta práctica se desaconseja encarecidamente.

Implicaciones en el mundo real: Un sitio web que utilice un certificado con una clave RSA de 1024 bits o firmado con SHA1 podría ser objetivo de atacantes con recursos computacionales significativos. Estos podrían descifrar el tráfico interceptado o crear certificados falsificados, permitiendo ataques de tipo man-in-the-middle y comprometiendo la seguridad de las comunicaciones. Por tanto, el cumplimiento de los estándares del sector que exigen claves y algoritmos robustos es esencial para mantener la seguridad de SSL/TLS.

Detalles técnicos

Hallazgo #1

WEAK_SIGNATURE_ALGORITHMS detectados en `demo.testfire.net`

- **Hallazgo:** Algoritmos de firma débiles detectados en TLS 1.2: RSA+SHA1.

Recomendación

Para mitigar las vulnerabilidades de claves débiles:

1. Utilice longitudes de clave adecuadas:

- Para RSA y DSA, utilice una longitud mínima de 2048 bits.
- Considere el uso de 3072 bits o superior para una seguridad a largo plazo.

2. Considere la criptografía de curva elíptica (ECC):

- ECC proporciona una seguridad equivalente con longitudes de clave más cortas, lo que ofrece un mejor rendimiento.
- Utilice curvas como P-256 o P-384 para una seguridad robusta.

3. Rotación periódica de claves:

- Implemente una política para rotar las claves periódicamente, especialmente al actualizar a longitudes de clave más robustas.

4. Audite los certificados existentes:

- Realice auditorías periódicas de sus certificados SSL/TLS para identificar y reemplazar aquellos con claves débiles.

5. Utilice Autoridades de Certificación modernas:

- Elija CAs que exijan requisitos de clave robustos y sigan las mejores prácticas de la industria.

6. Implemente prácticas sólidas de generación de claves:

- Utilice generadores de números aleatorios criptográficamente seguros al generar claves.
- Considere el uso de módulos de seguridad de hardware (HSMs) para la generación y el almacenamiento de claves.

Referencias

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

MEDIO

Abuso de la conexión de red móvil

La aplicación obliga a utilizar la conexión de red móvil.

El gestor de descargas es un servicio del sistema que gestiona descargas HTTP de larga duración. La aplicación no debería forzar el uso exclusivo de la conexión de red móvil.

Detalles técnicos

Hallazgo #1

Solo se permite la conexión mediante red móvil Taint rastreado desde el componente alcanzable: `owasp.sat.agoat.DownloadInvoiceService$downloadFile$1` encontrado en `owasp.sat.agoat.DownloadInvoiceService`. Desde la constante `1` hasta el sink `android.app.DownloadManager$Request` `android.app.DownloadManager$Request.setAllowedNetworkTypes(int)`. Trazo de taint:

```
at android.app.DownloadManager$Request.setAllowedNetworkTypes()
at owasp.sat.agoat.DownloadInvoiceService$downloadFile$1.invoke()
at android.app.DownloadManager$Request.setNotificationVisibility()
at owasp.sat.agoat.DownloadInvoiceService$downloadFile$1.invoke()
```

Método `owasp.sat.agoat.DownloadInvoiceService$downloadFile$1.invoke()`:

```
public final void invoke(org.jetbrains.anko.AnkoAsyncContext p10)
{
    kotlin.jvm.internal.Intrinsics.checkNotNull(p10, $receiver);
    android.net.Uri v1 = android.net.Uri.parse(https://github.com/satishpatnayak/MyTest/blob/master/AndroGoatInvoice.txt);
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1, Uri.parse(url1));
    android.app.DownloadManager$Request v2_2 = new android.app.DownloadManager$Request(v1);
    String v3 = v1.getLastPathSegment();
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v3, url1.lastPathSegment);
    int v5 = 0;
    v2_2.setAllowedNetworkTypes(3);
    v2_2.setTitle(((CharSequence) v3));
    v2_2.setDescription(((CharSequence) The File is downloading...));
    v2_2.allowScanningByMediaScanner();
    v2_2.setNotificationVisibility(1);
    v2_2.setDestinationInExternalPublicDir(android.os.Environment.DIRECTORY_DOWNLOADS, String.valueOf(System.currentTimeMillis()));
    kotlin.TypeCastException v6_8 = this.this$0.getSystemService(download);
    if (v6_8 == null) {
        throw new kotlin.TypeCastException(null cannot be cast to non-null type android.app.DownloadManager);
    } else {
        ((android.app.DownloadManager) v6_8).enqueue(v2_2);
        return;
    }
}
```

Método `owasp.sat.agoat.DownloadInvoiceService$downloadFile$1.invoke()`:

```
public final void invoke(org.jetbrains.anko.AnkoAsyncContext p10)
{
    kotlin.jvm.internal.Intrinsics.checkNotNull(p10, $receiver);
    android.net.Uri v1 = android.net.Uri.parse(https://github.com/satishpatnayak/MyTest/blob/master/AndroGoatInvoice.txt);
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1, Uri.parse(url1));
    android.app.DownloadManager$Request v2_2 = new android.app.DownloadManager$Request(v1);
    String v3 = v1.getLastPathSegment();
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v3, url1.lastPathSegment);
    int v5 = 0;
    v2_2.setAllowedNetworkTypes(3);
    v2_2.setTitle(((CharSequence) v3));
    v2_2.setDescription(((CharSequence) The File is downloading...));
    v2_2.allowScanningByMediaScanner();
    v2_2.setNotificationVisibility(1);
    v2_2.setDestinationInExternalPublicDir(android.os.Environment.DIRECTORY_DOWNLOADS, String.valueOf(System.currentTimeMillis()));
    kotlin.TypeCastException v6_8 = this.this$0.getSystemService(download);
    if (v6_8 == null) {
        throw new kotlin.TypeCastException(null cannot be cast to non-null type android.app.DownloadManager);
    }
}
```

```

    } else {
        ((android.app.DownloadManager) v6_8).enqueue(v2_2);
        return;
    }
}

```

Recomendación

La aplicación debería permitir que los usuarios utilicen tanto Wi-Fi como una conexión de red móvil. Una estrategia común en aplicaciones es descargar grandes volúmenes de datos únicamente cuando haya una red Wi-Fi disponible.

MEDIO Inyección XML

La inyección XML es una vulnerabilidad que se produce cuando la entrada de usuario no confiable no se valida ni se sanitiza correctamente antes de incorporarse a datos XML. Los atacantes pueden explotar esta vulnerabilidad inyectando contenido XML malicioso, lo que puede provocar acceso no autorizado, manipulación de datos o ataques de denegación de servicio.

La inyección XML es una vulnerabilidad que surge de la falta de validación y sanitización adecuadas de las entradas del usuario antes de incorporarlas a archivos o flujos de configuración XML. Esta omisión permite a un atacante explotar la aplicación inyectando contenido XML malicioso. Al manipular la estructura XML o introducir entidades maliciosas, un atacante puede alterar la lógica de la aplicación, manipular datos o extraer información sensible.

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: 'Demostración de inyección XML',
      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');

    // Crear datos XML
    final xmlData = '''
<root>
  <username>${_usernameController.text}</username>
  <password>${_passwordController.text}</password>
</root>
''';

```

```

// Guardar el XML en el archivo de configuración
await configFile.writeAsString(xmlData);
}

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

  // Ejecutar lógica de inicio de sesión
  print('Iniciando sesión con nombre de usuario: $username y contraseña: $password');
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Demostración de inyección XML'),
    ),
    body: Padding(
      padding: EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text('Ingresa el nombre de usuario:'),
          TextField(
            controller: _usernameController,
          ),
          SizedBox(height: 16.0),
          Text('Ingresa la contraseña:'),
          TextField(
            controller: _passwordController,
            obscureText: true,
          ),
          SizedBox(height: 16.0),
          ElevatedButton(
            onPressed: () async {
              await saveConfigFile();
              login();
            },
            child: Text('Iniciar sesión'),
          ),
        ],
      ),
    ),
  );
}

```

SWIFT

```

import SwiftUI

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

  var body: some View {
    VStack {
      Text("Ingresa el nombre de usuario:")
      TextField("Nombre de usuario", text: $username)
        .textFieldStyle(RoundedBorderTextFieldStyle())

      Text("Ingresa la contraseña:")
      SecureField("Contraseña", text: $password)
        .textFieldStyle(RoundedBorderTextFieldStyle())

      Button(action: {
        saveConfigFile()
        login()
      }) {
        Text("Iniciar sesión")
      }
    }
    .padding()
  }

  func saveConfigFile() {

```

```

// Crear datos XML
let xmlData = '''
<root>
  <username>\(username)</username>
  <password>\(password)</password>
</root>
'''

// Guardar el XML en el archivo de configuración
let configFilePath = "config.xml"
do {
    try xmlData.write(toFile: configFilePath, atomically: true, encoding: .utf8)
} catch {
    print("Error al escribir el XML en el archivo: \(error)")
}
}

func login() {
    // Ejecutar lógica de inicio de sesión
    print("Iniciando sesión con nombre de usuario: \(username) y contraseña: \(password)")
}

}

@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, "Error al guardar el archivo de configuración", Toast.LENGTH_SHORT).show()
    }
}

private fun login(username: String, password: String) {
    // Ejecutar lógica de inicio de sesión
    Toast.makeText(this, "Iniciando sesión con nombre de usuario: $username y contraseña: $password", Toast.LENGTH_SHORT).show()
}
}

```

Detalles técnicos

Hallazgo #1

El parámetro en la posición: **0** del método **loadData** de la clase **WebView** puede ser vulnerable a inyección XML. Taint trazado desde componente no alcanzable: **owasp.sat.agoat.XSSActivity**. El TaintElement proviene de: Clase **XSSActivity** Método **onCreate** Posición **0** Firma **(Ljava/lang/OSBundle;)V**. Traza de llamadas:

```

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

```

en **android.webkit.WebView.loadData**

No se pudo extraer el código fuente.

en **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;
}

```

en **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;
}

```

Recomendación

Para mitigar las vulnerabilidades de inyección XML, tenga en cuenta:

- Implemente técnicas adecuadas de validación y sanitización de entradas.
- Valide la entrada del usuario contra los formatos esperados y sanitícela eliminando o escapando los caracteres que podrían interpretarse como etiquetas o entidades XML.
- Considere utilizar bibliotecas o API seguras de análisis XML que gestionen el data binding de forma segura.

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: 'Demostración de inyección XML',
            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');

  // Crear datos XML
  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);

  // Guardar el XML en el archivo de configuración
  await configFile.writeAsString(xmlData);
}

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

  // Ejecutar la lógica de inicio de sesión
  print('Iniciando sesión con el nombre de usuario: $username y la contraseña: $password');
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Demostración de inyección XML'),
    ),
    body: Padding(
      padding: EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text('Nombre de usuario:'),
          TextField(
            controller: _usernameController,
          ),
          SizedBox(height: 16.0),
          Text('Contraseña:'),
          TextField(
            controller: _passwordController,
            obscureText: true,
          ),
          SizedBox(height: 16.0),
          ElevatedButton(
            onPressed: () async {
              await saveConfigFile();
              login();
            },
            child: Text('Iniciar sesión'),
          ),
        ],
      ),
    ),
  );
}
}

```

SWIFT

```

import Foundation

func main() {
  let configFilePath = "config.xml"

  // Cargar el archivo de configuración
  guard let configFileURL = Bundle.main.url(forResource: "config", withExtension: "xml") else {
    print("Error al cargar el archivo de configuración")
    return
  }
}

```

```

guard let xmlData = try? Data(contentsOf: configFileURL) else {
    print("Error al leer el archivo de configuración")
    return
}

// Analizar el XML
let xmlDoc: XMLDocument
do {
    xmlDoc = try XMLDocument(data: xmlData, options: .documentTidyXML)
} catch {
    print("Error al analizar el XML: \(error)")
    return
}

guard let root = xmlDoc.rootElement() else {
    print("Estructura XML no válida")
    return
}

// Leer los valores de configuración desde la entrada del usuario
guard let username = getUserInput(prompt: "Nombre de usuario:"), let password = getUserInput(prompt: "Contraseña:") else {
    print("Entrada no válida")
    return
}

// Actualizar los valores de configuración en el XML
if let usernameElement = root.elements(forName: "username").first {
    usernameElement.stringValue = username
}

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

// Guardar el XML modificado de nuevo en el archivo de configuración
guard let modifiedXmlData = xmlDoc.xmlData(options: .nodePrettyPrint) else {
    print("Error al generar el XML modificado")
    return
}

do {
    try modifiedXmlData.write(to: configFileURL)
} catch {
    print("Error al escribir el XML modificado en el archivo: \(error)")
    return
}

// Usar los valores de configuración
login(username: username, password: password)
}

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

func login(username: String, password: String) {
    // Ejecutar la lógica de inicio de sesión
    print("Iniciando sesión con el nombre de usuario: \(username) y la contraseña: \(password)")
}

// Iniciar el programa
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, "Error al guardar el archivo de configuración", Toast.LENGTH_SHORT).show()
        }
    }

    private fun login(username: String, password: String) {
        // Ejecutar la lógica de inicio de sesión
        Toast.makeText(this, "Iniciando sesión con el nombre de usuario: $username y la contraseña: $password", Toast.LENGTH_SHORT).show()
    }
}

```

Referencias

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

BAJO Atributo usesCleartextTraffic establecido

CVSS v3: 5.4

La aplicación establece el atributo usesCleartextTraffic.

El atributo `android:usesCleartextTraffic` indica si la aplicación utilizará tráfico de red en texto claro, como HTTP en texto claro.

El valor predeterminado para las aplicaciones dirigidas al nivel de API 27 o inferior es "true". En las dirigidas al nivel de API 28 o superior, el valor predeterminado es "false".

Detalles técnicos

Hallazgo #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.

Recomendación

Establezca explícitamente el atributo `android:usesCleartextTraffic` con valor `false` y defina una configuración de seguridad de red de Android. El valor predeterminado es `true` en aplicaciones dirigidas a API level 27 o inferior. Las aplicaciones dirigidas a API level 28 o superior usan `false` de forma predeterminada.

<> XML

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

Detalles del vector CVSS

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

Vector de ataque:

Red (N)Adyacente (A)Local (L)Físico (P)

Complejidad de ataque:

Bajo (L)Alto (H)

Privilegios requeridos:

Ninguno (N)Bajo (L)Alto (H)

Interacción del usuario:

Ninguno (N)Requerido (R)

Alcance:

Cambiado (C)Sin cambios (U)

Confidencialidad:

Alto (H)Bajo (L)Ninguno (N)

Integridad:

Alto (H)Bajo (L)Ninguno (N)

Disponibilidad:

Alto (H)Bajo (L)Ninguno (N)

Referencias

- [Storage updates in Android 11](#)

BAJO

Uso de versión obsoleta del protocolo TLS/SSL

La aplicación configura protocolos SSL/TLS obsoletos con problemas conocidos de seguridad y privacidad.

La familia de protocolos SSL/TLS cuenta con seis versiones: SSL v2, SSL v3, TLS v1.0, TLS v1.1, TLS v1.2 y TLS v1.3:

- SSL v2 es inseguro y no debe utilizarse. Esta versión del protocolo puede atacar claves RSA y sitios con el mismo nombre mediante el ataque DROWN.
- SSL v3 es inseguro cuando se usa con HTTP debido al ataque SSLv3 POODLE. Este protocolo se considera débil al emplearse con otros protocolos. Está obsoleto y no debe utilizarse.
- TLS v1.0 y TLS v1.1 son protocolos heredados vulnerables al ataque BEAST. Aunque las implementaciones modernas han aplicado algunas mitigaciones, ambos aún presentan debilidades conocidas y fueron declarados obsoletos por PCI DSS y los navegadores a partir de enero de 2020.
- TLS v1.2 y v1.3 no tienen problemas de seguridad conocidos.

Detalles técnicos

Hallazgo #1

TLS_1_0_DEPRECATED detected on demo.testfire.net

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

Recomendación

TLS v1.2 o TLS v1.3 debería ser el protocolo principal admitido, ya que esta versión ofrece cifrado autenticado moderno.

Referencias

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

Se ha detectado un Google Calendar de acceso público embebido en el objetivo que podría exponer información sensible, incluyendo detalles de reuniones, nombres de asistentes, horarios de eventos y datos internos de la organización.

Se ha detectado un Google Calendar accesible públicamente embebido en el objetivo que podría exponer información sensible, incluyendo detalles de reuniones, nombres de asistentes, horarios de eventos y datos internos de la organización.

Detalles técnicos

Hallazgo #1

Google Calendar – Exposure:

- <https://calendar.google.com/calendar/embed?src=hl6cjgs6ep1h7oniqgueu2bhbo%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=hl6cjs6ep1h7onigqueu2bhbo%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/"
}

```

Hallazgo #2

Google Calendar - Exposure:

- <https://calendar.google.com/calendar/embed?src=hl6cjs6ep1h7onigqueu2bhbo%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"
}
```

BAJO Atributo hasFragileUserData no definido

CVSSv3: 5,4

La aplicación no establece explícitamente el atributo hasFragileUserData.

`android:hasFragileUserData` es un atributo de Android que permite a los desarrolladores especificar si su aplicación contiene datos de usuario frágiles que deben protegerse.

Los datos de usuario frágiles son aquellos que podrían causar daño al usuario si se pierden, roban o utilizan de forma indebida.

Esto puede incluir información personal sensible, como números de la seguridad social, números de tarjetas de crédito, historiales médicos y otros datos sensibles.

Este valor debe establecerse de forma explícita en la aplicación para indicar si esta maneja o no datos importantes del usuario.

Si su valor es true, cuando el usuario desinstale la aplicación, se mostrará un mensaje preguntándole si desea conservar los datos de la aplicación.

Detalles técnicos

Hallazgo #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">
```

Recomendación

El flag `hasFragileUserData` puede agregarse al archivo `AndroidManifest.xml` de la aplicación.

<> XML

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

Detalles del vector CVSS

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

Vector de ataque:	Red (N) Adyacente (A) Local (L) Físico (P)
Complejidad de ataque:	Bajo (L) Alto (H)
Privilegios requeridos:	Ninguno (N) Bajo (L) Alto (H)
Interacción del usuario:	Ninguno (N) Requerido (R)
Alcance:	Cambiado (C) Sin cambios (U)
Confidencialidad:	Alto (H) Bajo (L) Ninguno (N)
Integridad:	Alto (H) Bajo (L) Ninguno (N)
Disponibilidad:	Alto (H) Bajo (L) Ninguno (N)

Referencias

- [Android Developer - Fragile User Data](#)

BAJO Versión de API de destino obsoleta

CVSS v3: 5,4

La aplicación establece el atributo `targetSdkVersion` para permitir el uso de un nivel de API anterior de Android, lo que podría exponer a los usuarios a vulnerabilidades de seguridad.

El atributo `android:targetSdkVersion` especifica el nivel de API de destino de Android requerido por la aplicación. Establecer un valor bajo de `targetSdkVersion` puede permitir que la aplicación se ejecute en versiones anteriores de Android, pero podría exponer a los usuarios a vulnerabilidades de seguridad.

Enlace al aviso de desuso: <https://support.google.com/googleplay/android-developer/answer/11926878?hl=en>

Detalles técnicos

Hallazgo #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"/>
```

Recomendación

Considere establecer el atributo `android:targetSdkVersion` en un valor que corresponda a un nivel de API de Android más reciente. Esto ayuda a garantizar que la aplicación se beneficie de mejoras de seguridad y a reducir el riesgo de vulnerabilidades asociadas a versiones anteriores de Android.

Detalles del vector CVSS

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

Vector de ataque:	Red (N) Adyacente (A) Local (L) Físico (P)
Complejidad de ataque:	Bajo (L) Alto (H)
Privilegios requeridos:	Ninguno (N) Bajo (L) Alto (H)
Interacción del usuario:	Ninguno (N) Requerido (R)
Alcance:	Cambiado (C) Sin cambios (U)
Confidencialidad:	Alto (H) Bajo (L) Ninguno (N)
Integridad:	Alto (H) Bajo (L) Ninguno (N)
Disponibilidad:	Alto (H) Bajo (L) Ninguno (N)

Referencias

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

BAJO Task Hijacking

La aplicación no protege contra el task hijacking, lo que permite que una aplicación maliciosa en el dispositivo secuestre su tarea en primer plano

Una tarea de Android es una colección de actividades con las que los usuarios interactúan al realizar un trabajo específico. Las actividades de diferentes aplicaciones pueden residir en la misma tarea, lo que podría utilizarse para reubicar una actividad maliciosa en la tarea de su aplicación manipulando los siguientes parámetros:

- **Task Affinity** controlada por el atributo `taskAffinity`
- **Task Reparenting** controlada por el atributo `allowTaskReparenting`

Task Affinity es un atributo de actividad definido en la etiqueta `<activity>` del archivo `AndroidManifest.xml`. Task Affinity especifica la tarea a la que la actividad desea unirse. De forma predeterminada, todas las actividades de una aplicación comparten la misma afinidad, que corresponde al nombre del paquete de la aplicación.

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`, cuando se establece en `true` para una actividad A, y cuando una nueva tarea con la misma afinidad se trae al frente, el sistema mueve la actividad A **reubicable** desde su tarea original a la nueva pila de tareas en primer plano.

Los ataques de Task Hijacking se presentan en diferentes variantes:

- **Task Affinity Control:** la aplicación tiene el nombre de paquete `com.mySecureApp.app` y la actividad **A1**. Una aplicación maliciosa tiene dos actividades **M1** y **M2**, donde `M2.taskAffinity = com.mySecureApp.app` y `M2.allowTaskReparenting = true`. Si la aplicación maliciosa está abierta en **M2**, una vez iniciada su aplicación, **M2** se reubica al frente y el usuario interactuará con la aplicación maliciosa.
- **Single Task Mode:** la aplicación ha establecido el modo de lanzamiento en `singleTask`. Una aplicación maliciosa con `M2.taskAffinity = com.mySecureApp.app` puede secuestrar la pila de tareas de la aplicación objetivo.
- **Task Reparenting:** la aplicación ha establecido `taskReparenting` en `true`. Una aplicación maliciosa puede mover la tarea de la aplicación objetivo a la pila de la aplicación maliciosa.

Task hijacking puede utilizarse para realizar phishing, ataques de denegación de uso, y ha sido explotado en el pasado por troyanos de malware bancario. Las nuevas variantes de estos ataques (StandHogg 2.0) son extremadamente difíciles de detectar, ya que se basan en código.

Task hijacking fue abordado en la versión 11 de Android como parte de la corrección de `CVE-2020-0267` `WindowManager` confused deputy.

Detalles técnicos

Hallazgo #1

The activity `owasp.sat.agoat.AccessControl1ViewActivity` has the `taskAffinity` attribute set to `None` and the `launchMode` attribute set to `None`, this activity is missing protection against task hijacking.

```
<activity xmlns:android="http://schemas.android.com/apk/res/android" android:label="@7F0B0035"
  android:name="owasp.sat.agoat.AccessControl1ViewActivity"><intent-filter><action android:name="android.intent.action.VIEW"/><category
  android:name="android.intent.category.DEFAULT"/><data android:scheme="androgoat" android:host="vulnapp"/></intent-filter></activity>
```

Hallazgo #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"/>
```

Hallazgo #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"/>
```

Hallazgo #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"/>
```

Hallazgo #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"/>
```

Hallazgo #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"/>
```

Hallazgo #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"/>
```

Hallazgo #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"/>
```

Hallazgo #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"/>
```

Hallazgo #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"/>
```

Hallazgo #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"/>
```

Hallazgo #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"/>
```

Hallazgo #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"/>
```

Hallazgo #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"/>
```

Hallazgo #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"/>
```

Recomendación

Las distintas formas de vulnerabilidades de Task Hijacking requieren correcciones distintas:

- Establezca el task affinity de las actividades de la aplicación como `""` (cadena vacía) en la etiqueta `<activity>` del `AndroidManifest.xml` para forzar que las actividades usen un task affinity generado aleatoriamente, o defínalo en la etiqueta `<application>` para aplicarlo a todas las actividades de la aplicación.

O

- Establezca `android:launchMode` en `singleInstance`. `singleInstance` garantiza que no se crearán otras actividades en la misma tarea.
- No defina el launch mode como `singleTask` o agregue soporte para un servicio de monitoreo que detecte la presencia de tareas maliciosas en primer plano.
- No establezca el flag `FLAG_ACTIVITY_NEW_TASK` en los intents de lanzamiento de actividades, o úselo junto con `FLAG_ACTIVITY_CLEAR_TASK`:

JAVA

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

```
i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK);
startActivity(i);
```

- No especifique `allowReparenting` con `taskAffinity` o agregue soporte para un servicio de monitoreo que detecte la presencia de tareas maliciosas en primer plano.
- Prefiera el uso de Explicit intent, que especifica qué aplicación satisfará el intent, proporcionando el nombre del paquete de la aplicación de destino o un nombre de clase de componente completamente calificado. El Implicit intent solo especifica la acción general.

Referencias

- [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](#)

POTENCIALMENTE

Configuración insegura de seguridad de red y falta de protecciones de transporte en ANALYTICS_LIBRARY

El sustituto ANALYTICS_LIBRARY (TrafficActivity) transmite el tráfico analítico mediante un OkHttpClient predeterminado sin pinning, mientras que la Network Security Configuration permite globalmente HTTP en texto claro y confía en CAs instaladas por el usuario. El pipeline carece de cifrado de payload, firma de solicitudes y controles de integridad, lo que permite de forma trivial la interceptación MitM, la reproducción y la manipulación.

El componente ANALYTICS_LIBRARY dentro del alcance (`owasp.sat.agoat.TrafficActivity`) realiza peticiones de red salientes mediante un `OkHttpClient` predeterminado inicializado sin pinning de certificados, gestión de confianza personalizada ni cifrado del payload. El archivo `network_security_config.xml` de la aplicación habilita explícitamente el tráfico en texto plano para todos los destinos y confía en autoridades de certificación instaladas por el usuario, lo cual neutraliza las protecciones TLS nativas del dispositivo. Un análisis estático exhaustivo de las 29 clases propias confirmó la ausencia total de cifrado del payload (AES/RSA/DES), firma HMAC, checksums de peticiones, interceptores OkHttp o generación de nonces en el pipeline de transmisión de analíticas. Si bien la compilación actual solo envía peticiones GET vacías sin parámetros y no exfiltra datos sensibles del usuario, la pila de transporte presenta una falla arquitectónica: cualquier atacante en la red puede interceptar, reproducir, modificar o sintetizar eventos de analíticas sin detección criptográfica. Si se introdujera un SDK de analíticas de producción en esta base de código, heredaría los mismos permisos de tráfico en texto plano y el canal TLS sin pinning, comprometiendo inmediatamente la confidencialidad y la integridad.

Detalles técnicos

Hallazgo #1

Root Cause

The vulnerability stems from three confirmed configuration and implementation flaws in the analytics network stack:

1. **Global cleartext permission:** `res/xml/network_security_config.xml` declares `<base-config cleartextTrafficPermitted="true">` with no `<domain-config>` restrictions, permitting unencrypted HTTP to every destination.
2. **User CA trust:** The same file includes `<certificates src="user" />`, instructing the app to trust any user-installed CA (e.g., Burp Suite, mitmproxy) for all TLS connections.
3. **Unpinned default client:** `TrafficActivity.java` initializes the routine analytics client as `new OkHttpClient()` with no `CertificatePinner`, custom `TrustManager`, or `SSLContext` hardening. Certificate pinning exists only inside an opt-in `doPinning()` method as a local variable and is never used for default traffic.

Vulnerable Code

Network Security Configuration (`base.apk/res/xml/network_security_config.xml`):

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
  <base-config cleartextTrafficPermitted="true">
    <trust-anchors>
      <certificates src="system" />
      <certificates src="user" />
    </trust-anchors>
  </base-config>
</network-security-config>
```

Default Unpinned OkHttpClient (`base.apk/sources/owasp/sat/agoat/TrafficActivity.java`):

```
private String httpurl = "http://demo.testfire.net";
private String httpsurl = "https://owasp.org";
private final OkHttpClient client = new OkHttpClient();
```

Bare Request Construction (No Payload, No Headers, No Signing):

```
public final void run(String url) {
    Intrinsics.checkNotNull(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/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();
```

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.

Recomendación

Inmediato

- **Restringir tráfico en texto claro:** Reemplace el `<base-config cleartextTrafficPermitted="true">` global por entradas `<domain-config>` explícitas que establezcan `cleartextTrafficPermitted="false"` para todos los endpoints de analítica. Si el texto claro es absolutamente necesario para un endpoint heredado específico, restrínjalo estrictamente a ese único dominio.
- **Eliminar la confianza en CA de usuario:** Elimine `<certificates src="user" />` de la configuración base en las compilaciones de producción. Si las pruebas internas requieren un proxy CA, restrínjalo a una variante de compilación de depuración mediante `<debug-overrides>` aplicado únicamente a dispositivos de prueba.

Corrección permanente

- **Implementar certificate pinning por defecto:** Configure el `OkHttpClient` predeterminado que usa `TrafficActivity` con un `CertificatePinner` que contenga al menos dos pines de respaldo SHA-256 para cada endpoint de analítica:

```
CertificatePinner pinner = new CertificatePinner.Builder()
    .add("analytics.example.com", "sha256/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=")
    .add("analytics.example.com", "sha256BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB=")
    .build();

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

- **Habilitar cifrado y firma de payload:** Si la biblioteca de analítica llega a transmitir datos sensibles, cifre el cuerpo de la solicitud mediante AES-GCM y añada una firma HMAC-SHA256 sobre el texto cifrado. Incluya un nonce monotónico o una marca de tiempo para prevenir ataques de reproducción.
- **Deshabilitar flags de depuración:** Establezca `android:debuggable="false"` y `android:allowBackup="false"` en el `AndroidManifest.xml` de producción.

Verificación

- Vuelva a descompilar el APK y verifique que `res/xml/network_security_config.xml` no contenga `cleartextTrafficPermitted="true"` en la configuración base ni `<certificates src="user" />`.
- Verifique que `TrafficActivity.java` inicialice su `OkHttpClient` predeterminado mediante `new OkHttpClient.Builder().certificatePinner(...).build()`.
- Busque en el código fuente descompilado `Cipher.getInstance()` y `Mac.getInstance()` para confirmar que las rutinas de cifrado y firma están presentes en la cadena de llamadas de analítica.
- Intercepte el tráfico con un proxy CA instalado por el usuario y confirme que la aplicación rechaza la conexión con una `SSLPeerUnverifiedException` debido a un desajuste de pinning.

POTENCIALMENTE

Información secreta almacenada en la aplicación

Contraseñas, tokens u otra información sensible se almacenan en la aplicación.

La filtración de secretos, contraseñas y claves de API puede tener graves consecuencias, entre ellas:

- **Pérdida de confidencialidad:** personas no autorizadas podrían acceder a información sensible si se filtran secretos o contraseñas.
- **Pérdida de integridad:** la filtración de secretos o contraseñas podría permitir el acceso no autorizado a sistemas o datos, con el consiguiente riesgo de alteración o corrupción de la información.
- **Pérdida de disponibilidad:** en algunos casos, el uso no autorizado de secretos o contraseñas filtrados podría denegar el acceso a usuarios legítimos y provocar una pérdida de disponibilidad en los sistemas o datos afectados.
- **Daño reputacional:** una empresa o individuo podría ver dañada su reputación si se revela que no pudo proteger adecuadamente la información sensible.
- **Consecuencias legales:** dependiendo de la naturaleza de la información filtrada y de las leyes de la jurisdicción correspondiente, la filtración de secretos o contraseñas podría acarrear consecuencias legales.
- **Sobrefacturación:** las claves de API filtradas podrían permitir que personas no autorizadas accedan a recursos de la API y realicen acciones facturables, provocando sobrefacturación.

Detalles técnicos

Hallazgo #1

Secret Detected: Google OAuth Access Token

File: `https://android.googleapis.com:443//auth`

Detected Secret: `ya29.m`

Hallazgo #2

Secret Detected: Generic Username/Password

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

Detected Secret: `Login": "satishpatnayak`

Hallazgo #3

Secret Detected: Google Cloud API Key

File: `https://www.googleapis.com:443//experimentsandconfigs/v1/getExperimentsAndConfigs?r=1&c=1`

Detected Secret: AIzaSyAJ_AuopqbXieQNXUJKLW5CluuGX7ujD1o

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_AuopqbXieQNXUJKLW5CluuGX7ujD1o
maps	✗	\$7/1.000 reqs	curl https://maps.googleapis.com/maps/api/js?key=AlzaSyAJ_AuopqbXieQNXUJKLW5CluuGX7ujD1o
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_AuopqbXieQNXUJKLW5CluuGX7ujD1o
Directions	✗	\$5/1.000 reqs	curl https://maps.googleapis.com/maps/api/directions/json?origin=Disneyland&destination=Universal+Studios+Hollywood4&key=AlzaSyAJ_AuopqbXieQNXUJKLW5CluuGX7ujD1o
Geocode	✗	\$5/1.000 reqs	curl https://maps.googleapis.com/maps/api/geocode/json?latlng=40.30&key=AlzaSyAJ_AuopqbXieQNXUJKLW5CluuGX7ujD1o
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_AuopqbXieQNXUJKLW5CluuGX7ujD1o
Autocomplete	✗	\$2.83/1.000 reqs	curl https://maps.googleapis.com/maps/api/place/autocomplete/json?input=Bingh&types=(cities)&key=AlzaSyAJ_AuopqbXieQNXUJKLW5CluuGX7ujD1o
Elevation	✗	\$5/1.000 reqs	curl https://maps.googleapis.com/maps/api/elevation/json?locations=39.7391536,-104.9847034&key=AlzaSyAJ_AuopqbXieQNXUJKLW5CluuGX7ujD1o
Timezone	✗	\$5/1.000 reqs	curl https://maps.googleapis.com/maps/api/timezone/json?location=39.6034810,-119.6822510&timestamp=1331161200&key=AlzaSyAJ_AuopqbXieQNXUJKLW5CluuGX7ujD1o
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_AuopqbXieQNXUJKLW5CluuGX7ujD1o
Geolocation	✗	\$5/1.000 reqs	curl https://www.googleapis.com/geolocation/v1/geolocate?key=AlzaSyAJ_AuopqbXieQNXUJKLW5CluuGX7ujD1o
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_AuopqbXieQNXUJKLW5CluuGX7ujD1o
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_AuopqbXieQNXUJKLW5CluuGX7ujD1o
Speed Limit- Roads	✗	\$20/1.000 reqs	curl https://roads.googleapis.com/v1/speedLimits?path=38.75807927603043,-9.03741754643809&key=AlzaSyAJ_AuopqbXieQNXUJKLW5CluuGX7ujD1o
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_AuopqbXieQNXUJKLW5CluuGX7ujD1o
Text Search- Places	✗	\$2.83/1.000 reqs	curl https://maps.googleapis.com/maps/api/place/queryautocomplete/json?input=pizza+nearpar&key=AlzaSyAJ_AuopqbXieQNXUJKLW5CluuGX7ujD1o
Playable Locations	✗	\$10/1.000 reqs	curl https://playablelocations.googleapis.com/v3/samplePlayableLocations?key=AlzaSyAJ_AuopqbXieQNXUJKLW5CluuGX7ujD1o
Google Custom Search	✗	\$5/1.000 reqs	curl https://www.googleapis.com/customsearch/v1?key=AlzaSyAJ_AuopqbXieQNXUJKLW5CluuGX7ujD1o&cx=016017638605632953755:bggehv7mz6v&q=test
Firebase Cloud Messaging	✗	NA	curl -i -s -X POST --header "Authorization: key=AlzaSyAJ_AuopqbXieQNXUJKLW5CluuGX7ujD1o" --header "Content-Type: application/json" --data '{"registration_ids": ["1"]}' https://fcm.googleapis.com/fcm/send

Hallazgo #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_FwkKIP_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_tDeuEmsRUsqyG83frY4&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+nearpar&key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w
Playable Locations	✗	\$10/1.000 reqs	curl https://playablelocations.googleapis.com/v3/samplePlayableLocations?key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w
Google Custom Search	✗	\$5/1.000 reqs	curl https://www.googleapis.com/customsearch/v1?key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w&cx=016017638605632953755:bggehv7mz6y&q=test
Firebase Cloud Messaging	✗	NA	curl -i -s -X POST --header "Authorization: key=AlzaSyAT_K5X0a1_ong7AZgtdm4SX6_FwkKlp_w" --header "Content-Type: application/json" --data '{"registration_ids": ["1"]}' https://fcm.googleapis.com/fcm/send

Recomendación

Los datos sensibles nunca deben incluirse en la propia aplicación. En su lugar, se deben utilizar métodos seguros para cifrar, almacenar y recuperar las credenciales de sus servicios y acceder a dichos datos según sea necesario.

Para mitigar el riesgo de sobrefacturación, considere implementar API key pinning o utilizar APIs autenticadas en servicios con costos de uso potencialmente elevados. El API key pinning permite restringir el uso de una clave a una aplicación específica exigiendo una firma criptográfica, y puede habilitarse desde el proveedor del servicio (p. ej., Google Maps).

Para las claves que podrían permitir el acceso no autorizado, es importante limitar los permisos y roles a funciones no críticas, o bien exponer el servicio mediante una API autenticada.

Para reforzar aún más la seguridad, la API también debe contar con controles de acceso adecuados y rate limiting, y las claves deben rotarse periódicamente para evitar usos no autorizados.

Referencias

- [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](#)

POTENCIALMENTE

Modo de respaldo activado

CVSS v3: 3,9

La aplicación está habilitando el modo de respaldo.

El modo de copia de seguridad es una función de Android que permite a los usuarios hacer copias de seguridad y restaurar datos y configuraciones de un dispositivo a otro. De forma predeterminada, Android realiza una copia de seguridad completa de las aplicaciones, incluidos los archivos privados almacenados en la partición `/data`. Cuando el modo de copia de seguridad está habilitado en un dispositivo Android, el **Backup Manager Service** sube periódicamente datos y configuraciones a la cuenta de Google Drive del usuario, como datos de aplicaciones, contraseñas de Wi-Fi y otras configuraciones. Estos datos pueden restaurarse posteriormente en el mismo dispositivo o en uno diferente si el usuario inicia sesión con la misma cuenta de Google.

Estas son algunas de las características principales del **Backup Manager Service** en Android:

1. *Copia de seguridad automática:* El **Backup Manager Service** realiza copias de seguridad automáticas de los datos de las aplicaciones a intervalos regulares, normalmente una vez al día. También puede iniciar copias de seguridad ante eventos específicos, como cuando el dispositivo se conecta a una fuente de alimentación o cuando el usuario inicia una copia manualmente.
2. *Copias de seguridad incrementales:* El **Backup Manager Service** realiza copias de seguridad incrementales, es decir, solo respalda los datos que han cambiado desde la última copia. Esto ayuda a reducir el tamaño de la copia de seguridad y a acelerar el proceso de copia de seguridad.
3. *Copias de seguridad cifradas:* Las copias de seguridad creadas por el **Backup Manager Service** se cifran con una clave única del dispositivo de cada usuario, garantizando que solo el usuario o alguien con las credenciales de su cuenta de Google pueda acceder a los datos de la copia de seguridad.
4. *Copias de seguridad específicas por aplicación:* El **Backup Manager Service** permite que las aplicaciones especifiquen qué datos deben incluirse en la copia de seguridad y cuáles deben excluirse de las copias. Esto ayuda a reducir el tamaño de las copias y a garantizar que no se incluyan datos sensibles.
5. *Funcionalidad de restauración:* El **Backup Manager Service** también proporciona una funcionalidad de restauración que permite a los usuarios restaurar los datos de sus aplicaciones en un dispositivo nuevo o restablecido de fábrica, o restaurarlos en un dispositivo existente después de que una aplicación haya sido desinstalada y reinstalada.

Detalles técnicos

Hallazgo #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">
```

Recomendación

En el archivo de manifiesto de la aplicación, configure el atributo `android:allowBackup` para habilitar o deshabilitar la copia de seguridad. El valor predeterminado es true, pero para dejar claras sus intenciones, se recomienda definir explícitamente el atributo en el manifiesto de la aplicación, como se muestra a continuación:

</ XML

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

Si la aplicación contiene datos sensibles que no desea que se respalden o restauren, puede deshabilitar el modo de copia de seguridad estableciendo el atributo

`android:allowBackup` en false, como se muestra a continuación:

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

Detalles del vector CVSS

Vector CVSS V3:

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

Vector de ataque:

☐ Red (N)
 ☐ Adyacente (A)
 ☐ Local (L)
 ☒ Físico (P)

Complejidad de ataque:

☒ Bajo (L)
 ☐ Alto (H)

Privilegios requeridos:

☐ Ninguno (N)
 ☐ Bajo (L)
 ☒ Alto (H)

Interacción del usuario:

☒ Ninguno (N)
 ☐ Requerido (R)

Alcance:

☐ Cambiado (C)
 ☒ Sin cambios (U)

Confidencialidad:

☒ Alto (H)
 ☐ Bajo (L)
 ☐ Ninguno (N)

Integridad:

☐ Alto (H)
 ☐ Bajo (L)
 ☒ Ninguno (N)

Disponibilidad:

☐ Alto (H)
 ☐ Bajo (L)
 ☒ Ninguno (N)

Referencias

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

POTENCIALMENTE

Travesía de rutas de archivo

La presencia de un path traversal en la aplicación indica una vulnerabilidad mediante la cual un atacante puede manipular parte de la ruta del archivo. Esta ruta manipulada se utiliza posteriormente en las APIs del sistema de archivos sin una validación o revisión adecuada.

Una aplicación puede permitir que un atacante navegue por el sistema de archivos más allá de los límites previstos. Esto puede conducir a un acceso no autorizado a archivos o directorios sensibles. Al manipular rutas de archivos, un atacante puede eludir los controles de acceso y recuperar o modificar datos críticos, como archivos de configuración, credenciales de usuario o documentos confidenciales. Esta vulnerabilidad representa una amenaza significativa, ya que permite al atacante escalar privilegios, ejecutar código arbitrario o lanzar ataques adicionales contra el sistema.

DART

```
import 'package:file/local.dart';

void main() {
  var file = new LocalFileSystem();
  var f = file.file("../..passwords.txt");
  f.copy("pass.txt");
}

// O cambiando la raíz del proceso actual en ejecución:

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}")
    }
}
```

Detalles técnicos

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

Hallazgo #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$0.getApplicationInfo().dataDir));
        v2_0.setReadable(1);
        v2_0.setWritable(1);
        CharSequence v3_1 = new java.io.FileWriter(v2_0);
        CharSequence v4_1 = new StringBuilder();
        v4_1.append(username is );
        String v5_1 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_1, username);
        v4_1.append(v5_1.getText().toString());
        v4_1.append(

    );

        v3_1.write(v4_1.toString());
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(password is );
        String v5_5 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v5_5, password);
        v4_4.append(v5_5.getText().toString());
        v4_4.append(

    );

        v3_1.write(v4_4.toString());
        v3_1.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (Exception v0_5) {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
        v0_5.printStackTrace();
    }
    return;
}
```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p8)
{
    try {
        android.content.Context v2_0 = java.io.File.createTempFile(users, tmp, new java.io.File(this.this$0.getApplicationInfo().dataDir));
        v2_0.setReadable(1);
        v2_0.setWritable(1);
        CharSequence v3_1 = new java.io.FileWriter(v2_0);
        CharSequence v4_1 = new StringBuilder();
        v4_1.append(username is );
        String v5_1 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v5_1, username);
        v4_1.append(v5_1.getText().toString());
        v4_1.append(

    );

        v3_1.write(v4_1.toString());
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(password is );
        String v5_5 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v5_5, password);
        v4_4.append(v5_5.getText().toString());
        v4_4.append(

    );

        v3_1.write(v4_4.toString());
        v3_1.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (Exception v0_5) {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
        v0_5.printStackTrace();
    }
    return;
}
```

Method `owasp.sat.agoat.InsecureStorageTempActivity$onCreate$1.onClick()` :

```
public final void onClick(android.view.View p8)
{
    try {
        android.content.Context v2_0 = java.io.File.createTempFile(users, tmp, new java.io.File(this.this$0.getApplicationInfo().dataDir));
        v2_0.setReadable(1);
        v2_0.setWritable(1);
        CharSequence v3_1 = new java.io.FileWriter(v2_0);
        CharSequence v4_1 = new StringBuilder();
        v4_1.append(username is );
        String v5_1 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v5_1, username);
        v4_1.append(v5_1.getText().toString());
        v4_1.append(

    );

        v3_1.write(v4_1.toString());
        CharSequence v4_4 = new StringBuilder();
        v4_4.append(password is );
        String v5_5 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkNotNull(v5_5, password);
        v4_4.append(v5_5.getText().toString());
        v4_4.append(

    );

        v3_1.write(v4_4.toString());
        v3_1.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (Exception v0_5) {
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data not saved), 1).show();
        v0_5.printStackTrace();
    }
    return;
}
```

Method `owasp.sat.agoat.InsecureStorageTempActivity.onCreate()` :

```
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;
}
```

Hallazgo #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.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.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;
    }
}

```

Hallazgo #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.$username;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_6, username);
    v0_1.append(v1_6.getText().toString());
    v0_1.append(' ');
    String v0_2 = v0_1.toString();
    try {
        owasp.sat.agoat.SQLInjectionActivity.access$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;
}

```

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.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.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;
}

```

Recomendación

Contención de rutas: Normalice la ruta y compruebe si se encuentra dentro del directorio de destino.

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}")
    }
}
```

Uso de rutas absolutas: Prefiera utilizar rutas absolutas en lugar de relativas siempre que sea posible. Al utilizar rutas absolutas, la aplicación especifica de forma explícita la ubicación exacta del archivo o directorio, sin margen de interpretación.

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")
    }
}

```

Referencias

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

POTENCIALMENTE

Almacenamiento inseguro de datos de la aplicación

Almacenamiento inseguro de datos de la aplicación que permite a otras aplicaciones leerlos y/o alterarlos.

La vulnerabilidad de almacenamiento inseguro en aplicaciones móviles se refiere a un fallo de seguridad en el que datos sensibles, como credenciales de usuario, información personal u otros datos confidenciales, se almacenan en el dispositivo en ubicaciones world-readable. Esto puede dejar los datos expuestos a accesos no autorizados por parte de otras aplicaciones del dispositivo o de un actor malicioso con acceso físico al mismo.

El almacenamiento inseguro no solo ocurre cuando la aplicación escribe en ubicaciones world-readable, sino también cuando carga datos, como archivos de configuración, desde ubicaciones world-writable, donde un actor malicioso puede modificar dichos archivos y, en consecuencia, manipular la funcionalidad de la aplicación.

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 {
            // Proceso de autenticación simulado
            val token = authenticate("username", "password") // Lógica de autenticación

            // Almacenar token en almacenamiento público
            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()
        // Realizar cualquier configuración adicional después de cargar la vista.
    }
}

```

```

    let loginButton = UIButton(type: .system)
    loginButton.setTitle("Iniciar sesión", 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() {
    // Proceso de autenticación simulado
    let token = authenticate(username: "username", password: "password")

    // Almacenar token en almacenamiento público
    storeTokenInPublicStorage(token: token)
}

private func storeTokenInPublicStorage(token: String) {
    let filePath = "/var/mobile/Media/insecure_app/jwt_config.txt"
    let fileURL = URL(fileURLWithPath: filePath)

    do {
        try token.write(to: fileURL, atomically: true, encoding: .utf8)
        print("Token almacenado correctamente.")
    } catch {
        print("Error al escribir el 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('Almacenamiento inseguro en Flutter'),
            ),
            body: Center(
                child: ElevatedButton(
                    onPressed: () {
                        // Proceso de autenticación simulado
                        String token = authenticate("username", "password");

                        // Almacenar token en almacenamiento público
                        storeTokenInPublicStorage(token);
                    },
                    child: Text('Iniciar sesión'),
                ),
            ),
        );
    }
}

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 almacenado correctamente.');
```

Detalles técnicos

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

```

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

```

Hallazgo #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;
}

```

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

```

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;
}

```

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

```

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

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

Hallazgo #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.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;
}

```

```
}
```

Recomendación

- **Usar almacenamiento seguro:** almacenar de forma segura datos sensibles como tokens de autenticación o credenciales. Utilizar los mecanismos de almacenamiento seguro que proporciona la plataforma, como Keychain en iOS y SharedPreferences con cifrado en Android.
- **Cifrado de datos:** cifrar los datos sensibles en reposo mediante algoritmos de cifrado robustos para proteger la información almacenada en el dispositivo del usuario.
- **Evitar la codificación rígida de información sensible:** en lugar de incluir directamente en el código nombres de usuario, contraseñas o tokens, considerar los mecanismos de almacenamiento seguro sugeridos anteriormente.
- **Evitar permisos de archivo permisivos:** asegurarse de que los permisos de archivo se configuren adecuadamente para restringir el acceso a archivos y directorios sensibles. En Android, por ejemplo, evitar utilizar `MODE_WORLD_READABLE` o `MODE_WORLD_WRITEABLE` al crear archivos, ya que otorgan acceso amplio de lectura o escritura a todas las aplicaciones. Aplicar siempre el principio de mínimo privilegio al establecer permisos de archivo y limitar el acceso únicamente a lo necesario para la funcionalidad de la aplicación.
- **Validación de entradas:** al cargar datos desde ubicaciones de almacenamiento públicas, asegurarse siempre de validarlos y sanitizarlos.

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.');
```

Referencias

- [CWE-359: Exposure of Private Information\("Privacy Violation"\)](#)
- [OWASP Mobile Top10](#)
- [CWE-200: Information Exposure](#)

POTENCIALMENTE

Source to Sink

Se invoca un método Sink peligroso utilizando parámetros potencialmente controlados por el usuario o un método Source no fiable.

Se utiliza un método Source o un parámetro controlado por el usuario para invocar a un método sink.

Source hace referencia a una entrada de datos no confiable que puede provenir de un usuario no confiable. Sink hace referencia a un método peligroso que, de ser accesible para un atacante, podría ser aprovechado para ejecutar un ataque.

Source y Sinks deben revisarse en busca de vulnerabilidades, como inyección, referencia indirecta a objetos o acceso no autorizado a datos.

Detalles técnicos

Hallazgo #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;
}
```

Hallazgo #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.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()` :

```
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;
}

```

Hallazgo #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.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;
}

```

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.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;
}

```

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.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;
}

```

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

```

Hallazgo #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.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;
}

```

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

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this,this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(
Username - );
    }
}

```

```

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

);

        java.io.IOException v0_2 = v0_1.toString();
        android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
        kotlin.jvm.internal.Intrinsics.checkNotNull(v4_0, Environment.getExternalStorageDirectory());
        java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
        android.widget.Toast v2_4 = new StringBuilder();
        v2_4.append(userinfo );
        v2_4.append(v1_10);
        System.out.println(v2_4.toString());
        v1_10.setReadable(1);
        v1_10.setWritable(1);
        java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
        v3_5.write(v0_2);
        v3_5.close();
        android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
    } catch (java.io.IOException v0_3) {
        v0_3.printStackTrace();
    }
    return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this.setContentView(2131296296);
    ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}

```

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

```

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

```

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

```

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

    );

    java.io.IOException v0_2 = v0_1.toString();
    android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
    java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
    android.widget.Toast v2_4 = new StringBuilder();
    v2_4.append(userinfo );
    v2_4.append(v1_10);
    System.out.println(v2_4.toString());
    v1_10.setReadable(1);
    v1_10.setWritable(1);
    java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
    v3_5.write(v0_2);
    v3_5.close();
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
} catch (java.io.IOException v0_3) {
    v0_3.printStackTrace();
}
return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p7)
{
    try {
        java.io.IOException v0_1 = new StringBuilder();
        v0_1.append( This data is stored in SdCard on );
        v0_1.append(new java.util.Date());
        v0_1.append(:
Username - );
        java.io.File v1_0 = this.$username;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_0, username);
        v0_1.append(v1_0.getText().toString());
        v0_1.append( Password - );
        java.io.File v1_4 = this.$password;
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, password);
        v0_1.append(v1_4.getText().toString());
        v0_1.append(

    );

    java.io.IOException v0_2 = v0_1.toString();
    android.content.Context v4_0 = android.os.Environment.getExternalStorageDirectory();
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_0, Environment.getExternalStorageDirectory());
    java.io.File v1_10 = java.io.File.createTempFile(users, tmp, new java.io.File(v4_0.getAbsolutePath()));
    android.widget.Toast v2_4 = new StringBuilder();
    v2_4.append(userinfo );
    v2_4.append(v1_10);
    System.out.println(v2_4.toString());
    v1_10.setReadable(1);
    v1_10.setWritable(1);
    java.io.FileWriter v3_5 = new java.io.FileWriter(v1_10);
    v3_5.write(v0_2);
    v3_5.close();
    android.widget.Toast.makeText(this.this$0.getApplicationContext(), ((CharSequence) Data saved), 1).show();
} catch (java.io.IOException v0_3) {
    v0_3.printStackTrace();
}
return;
}

```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this setContentView(2131296296);
    ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1.<init>()` :

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

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this setContentView(2131296296);
    ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}
```

Method `owasp.sat.agoat.InsecureStorageSDCardActivity.onCreate()` :

```
protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
    this setContentView(2131296296);
    ((android.widget.Button) this.findViewById(2131165201)).setOnClickListener(((android.view.View$OnClickListener) new
owasp.sat.agoat.InsecureStorageSDCardActivity$onCreate$1(this, ((android.widget.EditText) this.findViewById(2131165369)),
((android.widget.EditText) this.findViewById(2131165303)))));
    return;
}
```

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

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

```

Hallazgo #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.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;
}

```

Hallazgo #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.$ip;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v1_4, ip);
    v0_1.append(v1_4.getText().toString());
    String v0_2 = v0_1.toString();
    StringBuffer v1_2 = new StringBuffer();
    try {
        android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);
        kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v2_1, Runtime.getRuntime().exec(ip1,null,null));
        v2_1.waitFor();
        String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();
        kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v4_3, reader.readLine());
    } catch (android.widget.TextView v2_2) {
        v2_2.printStackTrace();
        android.widget.TextView v2_3 = this.$result;
        kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v2_3, result);
        v2_3.setText(((CharSequence) v1_2.toString()));
        return;
    }
    while (v4_3 != null) {
        String v5_3 = new StringBuilder();
        v5_3.append(v4_3);
        v5_3.append(n);
        v1_2.append(v5_3.toString());
    }
    v2_3 = this.$result;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v2_3, result);
    v2_3.setText(((CharSequence) v1_2.toString()));
    return;
}

```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p8)
{
    String v0_1 = new StringBuilder();
    v0_1.append(ping );
    StringBuffer v1_4 = this.$ip;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v1_4, ip);
    v0_1.append(v1_4.getText().toString());
    String v0_2 = v0_1.toString();
    StringBuffer v1_2 = new StringBuffer();
    try {
        android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);
        kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v2_1, Runtime.getRuntime().exec(ip1,null,null));
        v2_1.waitFor();
        String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();
        kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v4_3, reader.readLine());
    } catch (android.widget.TextView v2_2) {
        v2_2.printStackTrace();
        android.widget.TextView v2_3 = this.$result;
        kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v2_3, result);
        v2_3.setText(((CharSequence) v1_2.toString()));
        return;
    }
    while (v4_3 != null) {
        String v5_3 = new StringBuilder();
        v5_3.append(v4_3);
        v5_3.append(n);
        v1_2.append(v5_3.toString());
    }
    v2_3 = this.$result;
    kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(v2_3, result);
    v2_3.setText(((CharSequence) v1_2.toString()));
}

```

```

    return;
}

```

Method `owasp.sat.agoat.InputValidationsOSCMDInjectionMain2Activity$onCreate$1.onClick()` :

```

public final void onClick(android.view.View p8)
{
    String v0_1 = new StringBuilder();
    v0_1.append(ping );
    StringBuffer v1_4 = this.$ip;
    kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v1_4, ip);
    v0_1.append(v1_4.getText().toString());
    String v0_2 = v0_1.toString();
    StringBuffer v1_2 = new StringBuffer();
    try {
        android.widget.TextView v2_1 = Runtime.getRuntime().exec(v0_2, 0, 0);
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v2_1, Runtime.getRuntime().exec(ip1,null,null));
        v2_1.waitFor();
        String v4_3 = new java.io.BufferedReader(((java.io.Reader) new java.io.InputStreamReader(v2_1.getInputStream()))).readLine();
        kotlin.jvm.internal.Intrinsics.checkExpressionValueIsNotNull(v4_3, reader.readLine());
    } catch (android.widget.TextView v2_2) {
        v2_2.printStackTrace();
        android.widget.TextView v2_3 = this.$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()` :

```

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;
}

```

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

Recomendación

La recomendación varía según la clase de vulnerabilidad identificada.

POTENCIALMENTE Manipulación de URL

Un atacante puede controlar la URL que una aplicación utiliza para obtener contenido; esto supone un riesgo de seguridad, ya que puede manipular dicho contenido y dar lugar a la ejecución de código malicioso o al robo de información sensible.

Cuando un atacante toma el control de la URL que una aplicación utiliza para obtener contenido, esto supone un riesgo de seguridad crítico. Al manipular la URL, el atacante puede redirigir la aplicación hacia un servidor malicioso o inyectar su propio contenido en la respuesta. Esto puede derivar en diversas vulnerabilidades de

seguridad y consecuencias potenciales. Por ejemplo:

- Ejecución de código: el atacante puede modificar la URL para que apunte a un servidor bajo su control, lo que le permite entregar código malicioso a la aplicación. Este código podría explotar vulnerabilidades en la aplicación, ejecutar comandos arbitrarios o instalar malware en el sistema del usuario.
- Robo de datos: al redirigir la aplicación a un servidor fraudulento, el atacante puede engañar a los usuarios para que introduzcan información sensible, como credenciales de acceso, datos de tarjetas de crédito o datos personales, que pueden ser capturados y utilizados de forma indebida.
- Manipulación de contenido: el atacante puede modificar el contenido recuperado por la aplicación, alterando la información mostrada o inyectando scripts maliciosos. Esto puede derivar en diversas consecuencias, como mostrar información engañosa, realizar defacing de páginas web o ejecutar ataques de phishing.
- Ataques a la cadena de suministro: si la aplicación obtiene contenido de fuentes externas, como bibliotecas o plugins, el control de la URL puede permitir al atacante reemplazar recursos legítimos por versiones comprometidas o maliciosas. Esto puede comprometer la seguridad de todo el sistema.

A continuación se muestra un código de ejemplo de una aplicación que obtiene una entrada controlada por el atacante:

DART

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

class MainActivity extends StatefulWidget {
  final String url;

  MainActivity({required this.url});

  @override
  _MainActivityState createState() => _MainActivityState();
}

class _MainActivityState extends State<MainActivity> {
  String? content;

  @override
  void initState() {
    super.initState();
    fetchContent(widget.url);
  }

  Future<void> fetchContent(String url) async {
    try {
      final response = await http.get(Uri.parse(url));

      if (response.statusCode == 200) {
        setState(() {
          content = response.body;
        });
        // TODO: Process the fetched content here
      } else {
        // Handle error response
      }
    } catch (e) {
      // Handle network or API call error
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Fetching URL Content'),
      ),
      body: Center(
        child: content != null
          ? Text(content!)
          : CircularProgressIndicator(),
      ),
    );
  }
}
```

➤ SWIFT

```
import UIKit

class AppDelegate: UIResponder, UIApplicationDelegate {
```

```

var window: UIWindow?

func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool
{
    // Handle deep link URL if the app was launched from a URL
    if let url = launchOptions?[.url] as? URL {
        handleDeepLink(url)
    }
    return true
}

func application(_ application: UIApplication, open url: URL, options: [UIApplication.OpenURLOptionsKey : Any] = [:]) -> Bool {
    // Handle deep link URL when the app is already running
    handleDeepLink(url)
    return true
}

private func handleDeepLink(_ url: URL) {
    // Check if the URL scheme is "ostorlab"
    if url.scheme == "ostorlab" {
        // Extract the URL from the deep link
        if let deepLinkURL = URL(string: url.absoluteString.replacingOccurrences(of: "ostorlab://", with: "")) {
            // Fetch the content from the provided URL
            fetchContent(deepLinkURL)
        }
    }
}

private func fetchContent(_ url: URL) {
    URLSession.shared.dataTask(with: url) { (data, response, error) in
        if let error = error {
            print("Error fetching content: \(error.localizedDescription)")
            return
        }
        if let data = data, let content = String(data: data, encoding: .utf8) {
            // Process the fetched content here
            print("Fetched content: \(content)")
        }
    }.resume()
}
}

```

◀ KOTLIN

```

import android.content.Intent
import android.net.Uri
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch
import java.io.BufferedReader
import java.io.InputStreamReader
import java.net.HttpURLConnection
import java.net.URL

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

        // Get the URL from the intent
        val intent: Intent = intent
        val url: String? = intent.getStringExtra("url")

        if (url != null) {
            // Fetch the content from the provided URL
            GlobalScope.launch(Dispatchers.IO) {
                val content: String? = fetchContent(url)
                // Process the fetched content as needed
                if (content != null) {
                    // TODO: Process the fetched content here
                }
            }
        }
    }

    private fun fetchContent(urlString: String): String? {

```

```

var connection: HttpURLConnection? = null
var reader: BufferedReader? = null

try {
    val url = URL(urlString)
    connection = url.openConnection() as HttpURLConnection
    connection.requestMethod = "GET"

    // Check if the response code is successful
    if (connection.responseCode == HttpURLConnection.HTTP_OK) {
        reader = BufferedReader(InputStreamReader(connection.inputStream))
        val response = StringBuilder()
        var line: String?

        // Read the response line by line
        while (reader.readLine().also { line = it } != null) {
            response.append(line)
        }

        return response.toString()
    }
} catch (e: Exception) {
    e.printStackTrace()
} finally {
    // Close the connection and reader
    connection?.disconnect()
    reader?.close()
}

return null
}
}

```

Detalles técnicos

Hallazgo #1

The parameter at position: **0** of the method **loadUrl** in the class **WebView** is controlled by an attacker and can be used to retrieve malicious content. Taint traced from non-reachable component: **owasp.sat.agoat.InputValidationsWebViewURLActivity**. The TaintElement is coming from: Class **InputValidationsWebViewURLActivity** Method **onCreate** Position **0** Signature **(Landroid/os/Bundle;)V**. Call trace:

```

at android.webkit.WebView.loadUrl
at owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.onClick
at owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate
at owasp.sat.agoat.InputValidationsWebViewURLActivity$onCreate$1.<init>
at owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate
at owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate

```

at **android.webkit.WebView.loadUrl**

Unable to extract source code.

at **owasp.sat.agoat.InputValidationsWebViewURLActivity\$onCreate\$1.onClick**

```

public final void onClick(android.view.View p4)
{
    android.webkit.WebSettings v0_1 = this.$webview11.getSettings();
    kotlin.jvm.internal.Intrinsics.checkNotNull(v0_1, webSettings1);
    v0_1.setJavaScriptEnabled(1);
    String v1_4 = this.$url;
    kotlin.jvm.internal.Intrinsics.checkNotNull(v1_4, url);
    this.$webview11.loadUrl(v1_4.getText().toString());
    return;
}

```

at **owasp.sat.agoat.InputValidationsWebViewURLActivity.onCreate**

```

protected void onCreate(android.os.Bundle p5)
{
    super.onCreate(p5);
}

```

```

        this.setContentView(2131296293);
        android.widget.Button v0_3 = ((android.widget.Button) this.findViewById(2131165288));
        android.webkit.WebView v1_2 = this.findViewById(2131165374);
        kotlin.jvm.internal.Intrinsics.checkNotNullExpressionValue(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.checkNotNullExpressionValue(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.checkNotNullExpressionValue(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;
}

```

Hallazgo #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;  
}
```

Recomendación

Es fundamental adoptar medidas proactivas para proteger sus aplicaciones contra ataques de manipulación de URL. Considere las siguientes recomendaciones:

- Validación de entradas: Implemente mecanismos estrictos de validación de entradas para garantizar que las URL empleadas para obtener contenido estén correctamente formateadas y se ajusten a los patrones esperados. Esto puede incluir comprobar esquemas de URL válidos, exigir nombres de dominio esperados y validar parámetros de consulta.
- Enfoque de whitelist: Mantenga una whitelist de dominios o fuentes de confianza desde los que su aplicación obtenga contenido. Permita únicamente solicitudes a estas fuentes de confianza para minimizar el riesgo de acceder a contenido malicioso o no autorizado.
- Comprobaciones de integridad del contenido: Implemente mecanismos para verificar la integridad del contenido obtenido. Calcule y compare hashes criptográficos o firmas digitales del contenido recibido con los valores esperados para detectar cualquier modificación o manipulación.

Al implementar estas recomendaciones, podrá reforzar la seguridad de sus aplicaciones, proteger los datos de los usuarios y mitigar los riesgos asociados a los ataques de manipulación de URL.

A continuación se muestra un ejemplo de código para implementar la validación por whitelist:

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 {  
            // Manejar dominio no autorizado  
            // TODO: Implementar la acción adecuada para dominios no autorizados  
        }  
    }  
  
    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: Procesar aquí el contenido obtenido
        } else {
            // Manejar respuesta de error
        }
    } catch (e) {
        // Manejar error de red o de llamada a API
    }
}

@override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: Text('Fetching URL Content'),
        ),
        body: Center(
            child: content != null
                ? Text(content!)
                : CircularProgressIndicator(),
        ),
    );
}

```

SWIFT

```

import UIKit

class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    let allowedDomains = ["example.com", "trusteddomain.com"] // Agregue aquí los nombres de dominio de confianza

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Manejar la URL de deep link si la aplicación se inició desde una URL
        if let url = launchOptions?[.url] as? URL {
            handleDeepLink(url)
        }
        return true
    }

    func application(_ application: UIApplication, open url: URL, options: [UIApplication.OpenURLOptionsKey : Any] = [:]) -> Bool {
        // Manejar la URL de deep link cuando la aplicación ya se está ejecutando
        handleDeepLink(url)
        return true
    }

    private func handleDeepLink(_ url: URL) {
        // Comprobar si el esquema de URL es "ostorlab"
        if url.scheme == "ostorlab" {
            // Extraer la URL del deep link
            if let deepLinkURL = URL(string: url.absoluteString.replacingOccurrences(of: "ostorlab://", with: "")) {
                // Verificar el nombre de dominio contra la whitelist
                if isDomainAllowed(deepLinkURL) {
                    // Obtener el contenido de la URL proporcionada
                    fetchContent(deepLinkURL)
                } else {
                    // Manejar dominio no autorizado
                    // TODO: Implementar la acción adecuada para dominios no autorizados
                }
            }
        }
    }

    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) {
                // Procesar aquí el contenido obtenido
                print("Fetched content: \(content)")
            }
        }.resume()
    }
}

```

KOTLIN

```

import android.content.Intent
import android.net.Uri
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch
import java.io.BufferedReader
import java.io.InputStreamReader
import java.net.HttpURLConnection
import java.net.URL

class MainActivity : AppCompatActivity() {
    private val allowedDomains = listOf("example.com", "trusteddomain.com") // Agregue aquí los nombres de dominio de confianza

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Obtener la URL del intent
        val intent: Intent = intent
        val url: String? = intent.getStringExtra("url")

        if (url != null) {
            // Verificar el nombre de dominio contra la whitelist
            if (isDomainAllowed(url)) {
                // Obtener el contenido de la URL proporcionada
                GlobalScope.launch(Dispatchers.IO) {
                    val content: String? = fetchContent(url)
                    // Procesar el contenido obtenido según sea necesario
                    if (content != null) {
                        // TODO: Procesar aquí el contenido obtenido
                    }
                }
            } else {
                // Manejar dominio no autorizado
                // TODO: Implementar la acción adecuada para dominios no autorizados
            }
        }
    }

    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"

            // Comprobar si el código de respuesta es exitoso
            if (connection.responseCode == HttpURLConnection.HTTP_OK) {
                reader = BufferedReader(InputStreamReader(connection.inputStream))
                val response = StringBuilder()
                var line: String?

                // Leer la respuesta línea por línea
            }
        } catch (e: Exception) {
            // Manejar excepciones
        }
    }
}

```

```

        while (reader.readLine().also { line = it } != null) {
            response.append(line)
        }

        return response.toString()
    }
} catch (e: Exception) {
    e.printStackTrace()
} finally {
    // Cerrar la conexión y el lector
    connection?.disconnect()
    reader?.close()
}

return null
}
}

```

POTENCIALMENTE

Servicios declarados sin permisos

CVSS v3: 3,4

Los servicios declarados no están protegidos con permisos globales.

`service` es un componente de aplicación que gestiona acciones en segundo plano sin interacción del usuario. También puede utilizarse para exponer funcionalidades a otras aplicaciones. Esto corresponde a las llamadas a `Context.bindService()` para establecer una conexión con el servicio e interactuar con él.

Los servicios no protegidos pueden ser invocados por otras aplicaciones y acceder potencialmente a información sensible o realizar acciones privilegiadas.

Detalles técnicos

Hallazgo #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"/>

```

Recomendación

`service` puede exponer varios métodos a componentes externos. Es posible definir permisos arbitrarios para cada método mediante el método `checkPermission`.

También es posible separar servicios y restringir el acceso aplicando permisos en la etiqueta `<service>` del manifiesto.

<> 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>

```

El servicio puede aplicar permisos en llamadas IPC individuales invocando el método `checkCallingPermission` antes de ejecutar la implementación de dicha llamada.

Detalles del vector CVSS

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

Vector de ataque:

Red (N)Adyacente (A)Local (L)Físico (P)

Complejidad de ataque:

Bajo (L)Alto (H)

Privilegios requeridos:

Ninguno (N)Bajo (L)Alto (H)

Interacción del usuario:

Ninguno (N)Requerido (R)

Alcance:

Cambiado (C)Sin cambios (U)

Confidencialidad:

Alto (H)Bajo (L)Ninguno (N)

Integridad:

Alto (H)Bajo (L)Ninguno (N)

Disponibilidad:

Alto (H)Bajo (L)Ninguno (N)

Referencias

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

REFUERZO

Configuración de red insegura

CVSS v3: 9,1

La aplicación no especifica una configuración de seguridad de red o establece ajustes inseguros.

Android Network Security Configuration permite configurar de forma declarativa la seguridad de red de la aplicación.

Si Android Network Security Configuration no está presente o se configura de forma insegura, la aplicación puede quedar vulnerable a MITM.

Detalles técnicos

Hallazgo #1

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

Recomendación

Android Network Security Configuration es un archivo XML que permite configurar de forma declarativa la seguridad de red de la aplicación.

Añadir la configuración de seguridad de red a la aplicación:

- 1– Crea un nuevo archivo XML en el directorio res/xml de la aplicación. Nómbralo network_security_config.xml , o con cualquier otro nombre adecuado. 2– Define las configuraciones de seguridad. Consulta los ejemplos a continuación 3– Aplica la configuración en el Manifiesto:

</ XML

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

Ejemplos de configuración de seguridad de red

- Autoridad de certificación personalizada con soporte para configuraciones exclusivas de depuración

</ 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>
```

```
</trust-anchors>
</debug-overrides>
</network-security-config>
```

- Desactivación declarativa del tráfico en texto plano

</ 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>
```

- Configuración declarativa de claves de certificate pinning

</ XML

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
  <domain-config>
    <domain includeSubdomains="true">example.com</domain>
    <pin-set expiration="2018-01-01">
      <pin digest="SHA-256">7HIpactkIAq2Y49orF00QKurwxmmSFZhBCoQYcRhJ3Y=</pin>
      <!-- backup pin -->
      <pin digest="SHA-256">fwza0LRMXouZHRc8Ei+4PyuldPDcf3UKg0/04cDM1oE=</pin>
    </pin-set>
  </domain-config>
</network-security-config>
```

Detalles del vector CVSS

Vector CVSS V3:

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

Vector de ataque:

Red (N) Adyacente (A) Local (L) Físico (P)

Complejidad de ataque:

Bajo (L) Alto (H)

Privilegios requeridos:

Ninguno (N) Bajo (L) Alto (H)

Interacción del usuario:

Ninguno (N) Requerido (R)

Alcance:

Cambiado (C) Sin cambios (U)

Confidencialidad:

Alto (H) Bajo (L) Ninguno (N)

Integridad:

Alto (H) Bajo (L) Ninguno (N)

Disponibilidad:

Alto (H) Bajo (L) Ninguno (N)

Referencias

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

REFUERZO

Modo de depuración activado

CVSS v3: 5,4

La aplicación se compiló con el modo de depuración habilitado.

El atributo `android:debuggable` en `Application`, definido en el Android Manifest, determina si la aplicación puede depurarse.

El modo de depuración permite a los atacantes acceder al sistema de archivos de la aplicación, adjuntar un depurador, acceder a datos sensibles o realizar acciones maliciosas.

Los siguientes pasos pueden utilizarse para iniciar una sesión de depuración mediante `jdb`:

- Utilizar `adb jdwp` para identificar el `PID` de la aplicación objetivo:

```
$adb jdwp
3466
15446
```

- Crear un canal de comunicación mediante `adb` y adjuntarse a él mediante `jdb` :

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

- Acceder al sistema de archivos de la aplicación:

```
$adb shell
$run-as package-id
$..insert malicious action...
```

Un atacante puede depurar la aplicación sin acceso al código fuente y aprovecharlo para realizar acciones maliciosas en nombre del usuario, modificar el comportamiento de la aplicación o acceder a datos sensibles como credenciales y cookies de sesión.

Detalles técnicos

Hallazgo #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">
```

Recomendación

Deshabilite el modo de depuración estableciendo el atributo `android:debuggable` en false en la etiqueta `Application` del Android Manifest.

</> XML

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

Detalles del vector CVSS

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

Vector de ataque:

Red (N) Adyacente (A) Local (L) Físico (P)

Complejidad de ataque:

Bajo (L) Alto (H)

Privilegios requeridos:

Ninguno (N) Bajo (L) Alto (H)

Interacción del usuario:

Ninguno (N) Requerido (R)

Alcance:

Cambiado (C) Sin cambios (U)

Confidencialidad:

Alto (H) Bajo (L) Ninguno (N)

Integridad:

Alto (H) Bajo (L) Ninguno (N)

Disponibilidad:

Alto (H) Bajo (L) Ninguno (N)

Referencias

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

Superficie de ataque

