Documentación de Vulnerabilidades encontradas

Integrantes: Daniela Díaz – Abner Michea

Introducción

En este documento de detalla la identificación, documentación y corrección de vulnerabilidades críticas en la aplicación GPSMapApp. Se han realizado pruebas de seguridad para mejorar la robustez de la aplicación. Se incluye una documentación detallada en cada cambio y sus impactos en la seguridad.

1. Identificación de Vulnerabilidades Críticas

Pruebas de vulnerabilidad

Se ejecutaron pruebas utilizando las herramientas MobSF para identificar vulnerabilidades de la aplicación y se documentan tres de las vulnerabilidades más críticas encontradas y sus detalles técnicos:

Vulnerabilidad	Gravedad	Descripción	Impacto
Depuración Habilitada	**Alto**	La aplicación tenía habilitada la opción android:debuggable=true, permitiendo a los atacantes enganchar un depurador.	Facilita a los atacantes realizar ingeniería inversa, identificar estructuras de código y explotar fallas de seguridad presentes en la aplicación.
Permisos de Localización Excesivos	**Alto**	La aplicación solicita permisos para ACCESS_FINE_LOCATIONy ACCESS_COARSE_LOCATION, lo que puede ser excesivo.	Los atacantes podrían rastrear al usuario en tiempo real, comprometiendo su privacidad y exponiendo datos sensibles de ubicación.
Tráfico de Red sin Cifrado (HTTP)	**Alto**	Se identificó que la aplicación se conecta a servidores a través de HTTP en vez de HTTPS.	Los datos pueden ser interceptados, alterados o robados por un atacante en redes públicas o no seguras, facilitando ataques de intermediario (MITM).

```
<application
    android:networkSecurityConfig="@xml/network_security_config"
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:debuggable="false"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.GPSMapApp"
    tools:targetApi="31"
    tools:ignore="HardcodedDebugMode">
    <activity
        android:name=".MainActivity"
        android:exported="true" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="false">
        <domain includeSubdomains="true">yourdomain.com</domain>
    </domain-config>
</network-security-config>
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" >

    <!--<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />-->
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

2. Corrección de Vulnerabilidades

En respuesta a las vulnerabilidades encontradas, se han realizado correcciones aplicando buenas prácticas y consejos de seguridad específicos.

Mejores prácticas	**Descripción de Implementación**	**Impacto en la Seguridad**
Deshabilitar Modo Depuración en Producción	Se inició la opción android:debuggable=falseen el archivo AndroidManifest.xml.	Previene la ingeniería inversa y evita que los atacantes puedan engancharse al proceso de la aplicación.
Uso restringido de permisos	Se eliminó el permiso de ubicación precisa ACCESS_FINE_LOCATION, limitando la aplicación a solicitar ACCESS_COARSE_LOCATIONcuando sea estrictamente necesario.	Minimiza el acceso a la información de ubicación del usuario, reduciendo el riesgo de exposición a terceros.
Forzar el uso de HTTPS	Se desarrolló una política de red network_security_config.xmlpara forzar el uso de HTTPS en todas las conexiones de red.	Protege los datos de usuario y previene interceptaciones o modificaciones de los datos en tránsito.

3. Consejos de seguridad Implementados y su impacto en la seguridad

Esta configuración indica que la aplicación solo confiará en certificados con el pin SHA-256 especificado, rechazando conexiones a servidores que no coincidan con el certificado configurado.

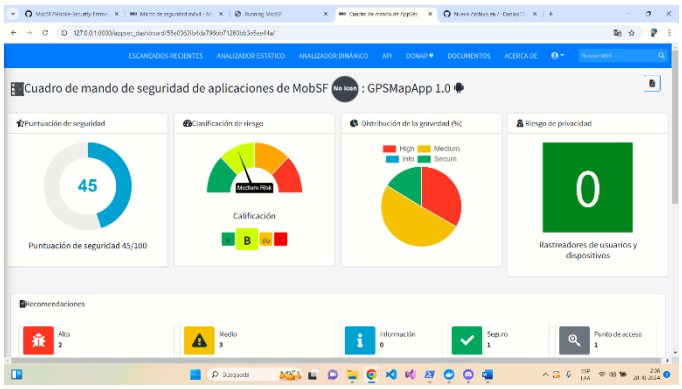
Consejo de Seguridad	Descripción de implementación	Impacto en la seguridad
Aplicar Certificado SSL Fijación	Se implementó SSL Pinning para verificar la autenticidad del certificado en cada conexión de red.	Protege contra ataques MITM, asegurando que solo se conecta a servidores con certificados legítimos.
Deshabilitar Copias de Seguridad de Aplicación	Se inició android:allowBackup=falseen el AndroidManifest.xmlpara evitar la creación de copias de seguridad de los datos de la aplicación a través de ADB.	Impida que usuarios o atacantes con acceso físico al dispositivo puedan extraer datos de la aplicación.
Habilitar Configuración de Seguridad en Componentes	Se revisó y configuró todos los componentes exportados en el manifiesto para evitar accesos no autorizados, limitando accesos externos.	Previene que otras aplicaciones accedan a componentes privados, fortaleciendo el aislamiento de datos.

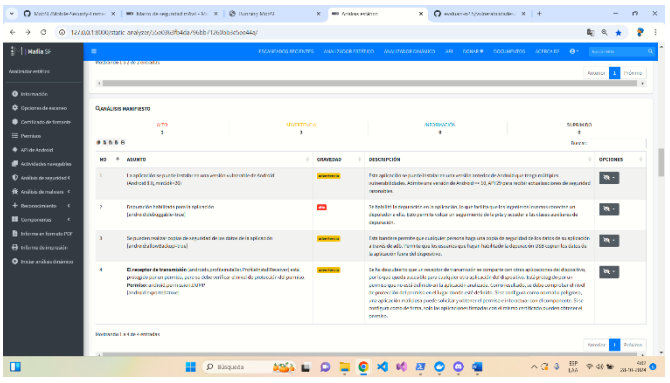
```xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="false">
        <domain includeSubdomains="true">yourdomain.com</domain>
        <pin-set expiration="2025-01-01">
            <pin digest="SHA-256">AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=</pin>
        </pin-set>
    </domain-config>
</network-security-config>
```

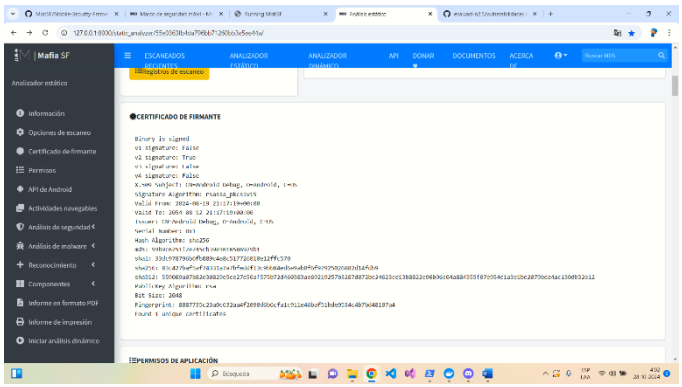
```java
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;
import okhttp3.CertificatePinner;

public class NetworkClient {
    private static OkHttpClient client;

    public static OkHttpClient getClient() {
        if (client == null) {
            CertificatePinner certificatePinner = new CertificatePinner.Builder()
                    .add("yourdomain.com", "sha256/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=")
                    .build();

            client = new OkHttpClient.Builder()
                    .certificatePinner(certificatePinner)
                    .build();
        }
        return client;
    }

    public static Response makeRequest(String url) throws IOException {
        Request request = new Request.Builder()
                .url(url)
                .build();

        return getClient().newCall(request).execute();
    }
}
```

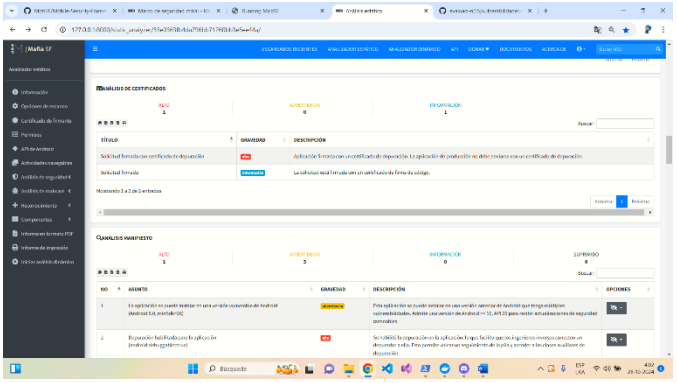
```xml
<application
    android:networkSecurityConfig="@xml/network_security_config"
    android:allowBackup="false"
    android:dataExtractionRules="@xml/data_extraction_rules"
```

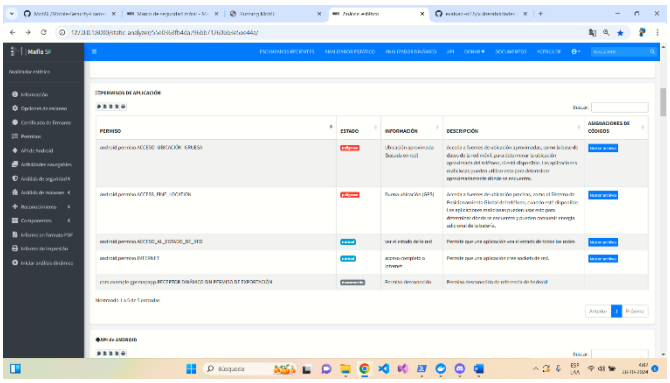
Evidencia del informe

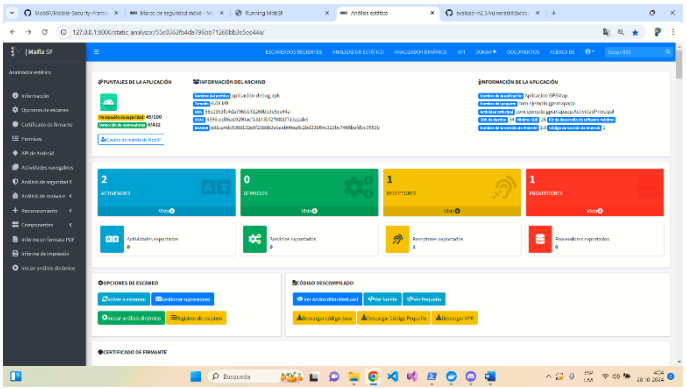


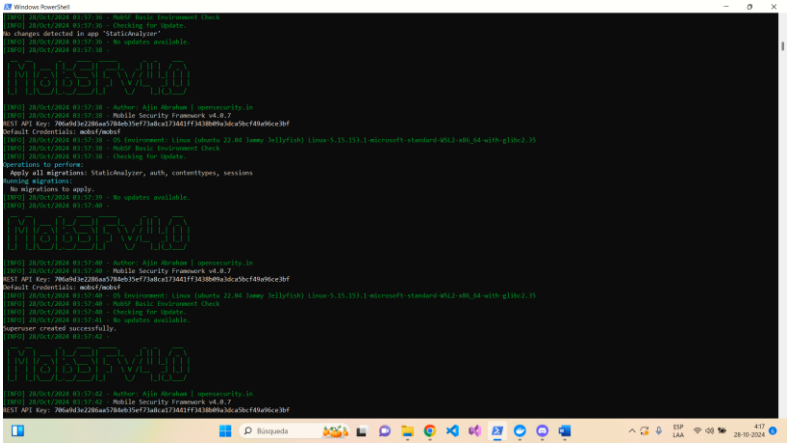












Preguntas de cierre

1. **¿Qué nuevos conocimientos y habilidades has adquirido en la protección de aplicaciones Android contra amenazas de seguridad?**
   Aprendimos a mejorar la seguridad en apps Android aplicando cosas como SSL Pinning para asegurar que las conexiones de red sean confiables, y cómo usar network_security_config.xmlpara obligar a la app a conectarse solo por HTTPS. También nos dimos cuenta de la importancia de controlar permisos sensibles (como los de ubicación) y de desactivar la depuración y copias de seguridad en la versión final para proteger los datos de los usuarios.

2. **¿Cómo podrías utilizar los conocimientos adquiridos para mejorar la seguridad en aplicaciones móviles en diferentes entornos, como aplicaciones financieras, de salud, o de comercio electrónico?**
   Este aprendizaje se puede aplicar en apps de áreas sensibles como finanzas, salud o comercio. Por ejemplo, en una aplicación financiera, podríamos usar SSL Pinning y HTTPS para proteger datos de pago. En una de salud, limite los permisos protege la privacidad del usuario. Estos pasos ayudan a crear una aplicación segura y confiable en distintos entornos.

3. **¿Lograste implementar las mejores prácticas y los consejos de seguridad de manera efectiva? ¿Qué desafíos encontraste en el proceso y cómo los superaste?**
   Pudimos aplicar las prácticas de seguridad correctamente, aunque algunos pasos, como configurar SSL Pinning y limitar permisos, fueron complicados. Lo resolvimos consultando documentación y probando ajustes hasta hacerlo bien. Este proceso nos enseñó cómo manejar la seguridad en aplicaciones Android de forma práctica y efectiva.