SANTO TOMÁS

Vulnerabilidades en desarrollo movil

Nombre: Andres Pasten Rodriguez.

Carrera: Analista Programador.

Profesor: Oscar Monardez.

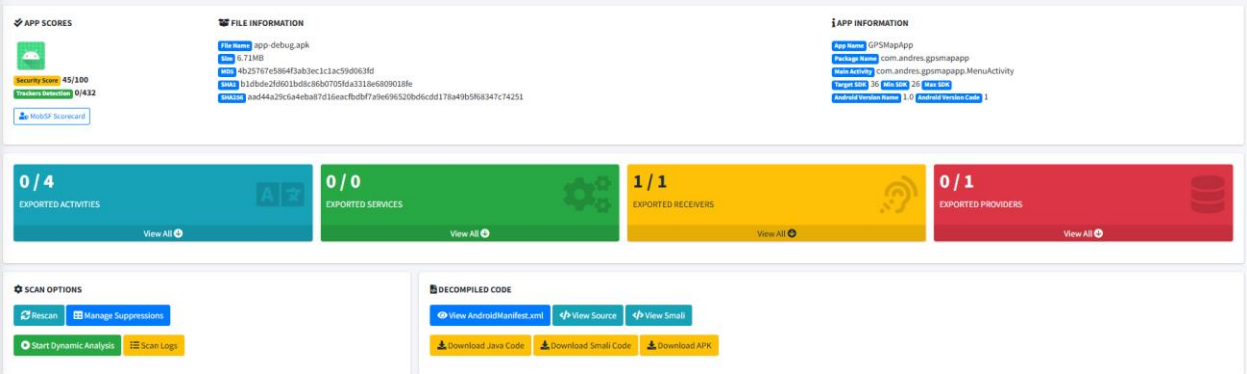
Link GitHub: https://github.com/andre19fs/EvGPSMapApp.git

Analisis de vulnerabilidades

1. Identificacion de vulnerabilidades

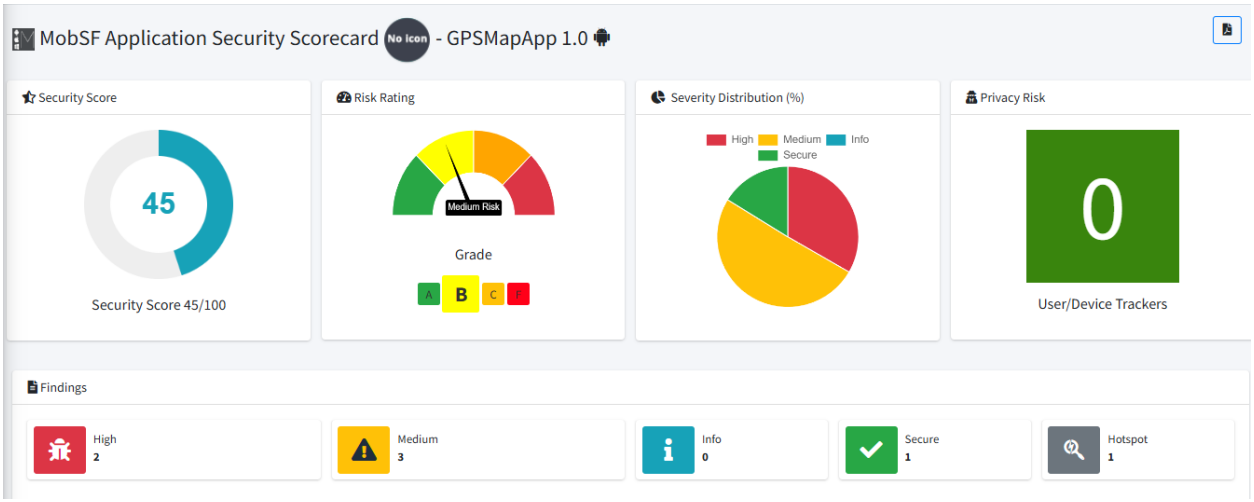
Herramienta de análisis estático utilizada: MobSF.

Aquí observamos de manera general el resultado del análisis estático que se le realizo a nuestra app a través de MobSF. Observamos versión de sdk, nombre de la app, tamaño, puntaje de seguridad y las categorías de alertas.



Scorecard de Nuestra app:

Aquí observamos la escala de valoración de muestra app, 45/100 y no se encuentra muy alto en el ranking de riesgo, esta parte es útil ya que nos permite ver gráficamente el estado de seguridad en nuestra app, como dije anteriormente, no se encuentra en un estado crítico, como “c” o “d”, pero en el estado actual representa un gran riesgo para la privacidad y los datos de los usuarios, por lo que es importante manejar las vulnerabilidades que veremos abajo.



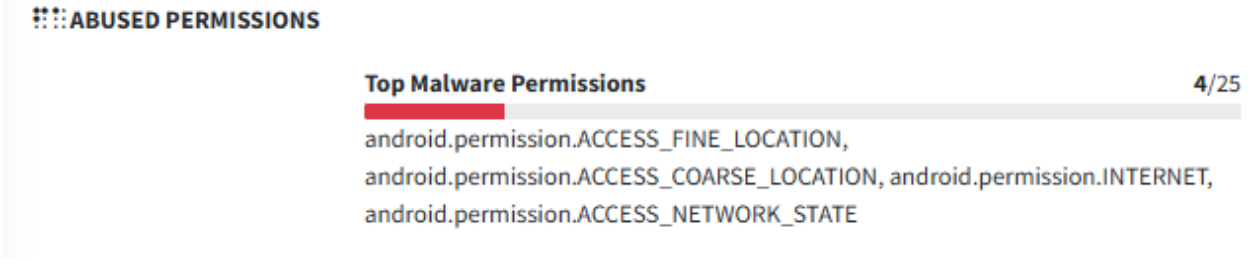
Permisos

-Primero, nuestra aplicación utiliza diversos permisos para poder llevar a cabo funcionalidades como el uso de google maps, seguimiento de la ubicación y descarga de imágenes:

PERMISSION	STATUS
android.permission.ACCESS_COARSE_LOCATION	dangerous
android.permission.ACCESS_FINE_LOCATION	dangerous
android.permission.ACCESS_NETWORK_STATE	normal
android.permission.INTERNET	normal

-ACCESS_COARSE_LOCATION: Este permiso puede llevar a que aplicaciones de terceros, puedan determinar nuestra ubicación aproximada, lo que implica un riesgo para la seguridad de cada usuario.

-ACCESS_FINE_LOCATION: Este permiso, al igual que el anterior puede permitir que terceros accedan a nuestra ubicación, en este caso, de forma precisa y no aproximada, por lo que es más peligroso.



Aquí observamos que cuatro de nuestros permisos están dentro de los más vulnerados por malware, por lo que tenemos que poner especial atención a como los tratamos.

SANTO TOMÁS

Vulnerabilidades

ISSUE	SEVERITY
App can be installed on a vulnerable Android version Android 8.0, minSdk=26]	warning
Debug Enabled For App [android:debuggable=true]	high
Application Data can be Backed up [android:allowBackup=true]	warning
Broadcast Receiver (androidx.profileinstaller.ProfileInstallReceiver) is Protected by a permission, but the protection level of the permission should be checked. **Permission:** android.permission.DUMP [android:exported=true]	warning

-Versión del SDK: la aplicación esta creada para la versión 26 del sdk (Android 8), que, al ser mas antigua tiene múltiples tipos de vulnerabilidades, ya no cuenta con soporte, por lo que no cuenta con actualizaciones de seguridad.

-Debug activado: La aplicación tiene la depuración activada, por lo que terceros pueden acceder a logs (registros), corriendo el riesgo de exponer datos privados o información confidencial.

-Backup: Esto significa que cualquier persona puede realizar copias de los datos de la aplicación fuera del dispositivo a través de la depuración USB

-Broadcast Reciver: Esto significa que nuestra aplicación permite el acceso a aplicaciones que se encuentran en nuestro dispositivo, estos accesos se manejan con un permiso, el cual nuestra app no tiene, por lo que cualquier app del dispositivo podrá acceder, ya sea para alguna utilidad, o para extraer datos o romper la app.

2.Test de vulnerabilidad

Análisis dinámico y estático realizado con la aplicación MobSF:

1.- Análisis Estático:

Para esta prueba descargamos el apk de nuestra app, y lo arrastramos a la aplicación MobSF que previamente descargue:

Pruebas realizadas:

-Análisis de permisos:

-Análisis de exportación de actividades:

-Comprobación de seguridad de componentes:

-Revisión de URLs:

-Revisión API keys:

-Revisión del código para buscar debilidades:

2.- Análisis dinámico:

Para esta prueba utilice un emulador de Android llamado “NoxPlayer” ya que se requería que el dispositivo tuviera permisos de escritura para que MobSF pueda realizar las pruebas:

Pruebas realizadas:

-Intercepción de tráfico HTTP/HTTPS

-Monitoreo de llamadas de red

-Monitoreo de almacenamiento de datos

-Fugas de datos

Resultados

Las vulnerabilidades detectadas las separare en rangos Altas, Medias y bajas.

Vulnerabilidades Altas:

-Componentes exportados sin protección: Las activities de la app (Actividad de mapa, imagen y menú) Son accesibles para terceros, lo que compromete realizar acciones sensibles desde fuera.

Permisos de la app: Si bien los permisos de la app están justificados, ya que las funcionalidades de esta dependen de ellos, son peligrosos, ya que permiten el acceso a datos sensibles y hardware por parte de terceros.

Claves expuestas: Mi aplicación deja expuesta el api key de Google cloud, esto es considerablemente peligroso, ya que, si alguien puede extraerla y utilizarla, va a generar costos para el dueño de la key, en este caso, yo. También puede romper la continuidad del servicio, además de filtración de datos.

Vulnerabilidades medias:

Trafico HTTP inseguro: Mi aplicación utiliza el protocolo de comunicación HTTP y no HTTPS por lo que el intercambio de datos entra la app e internet no es seguro, dejando la posibilidad de que cualquiera pueda acceder a ellos.

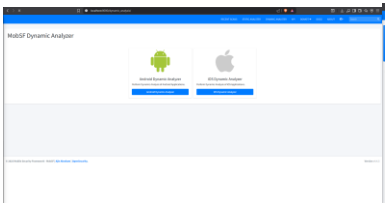
Vulnerabilidades bajas:

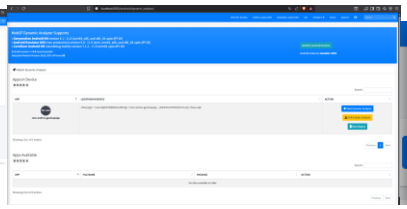
Modo debbug: La aplicación tiene la depuración activada, por lo que terceros pueden acceder a logs (registros), corriendo el riesgo de exponer datos privados o información confidencial.

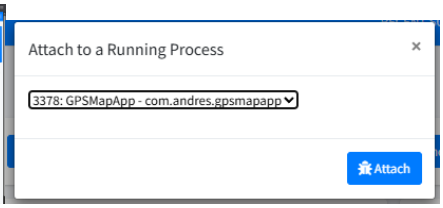
Evaluaciones de los riesgos

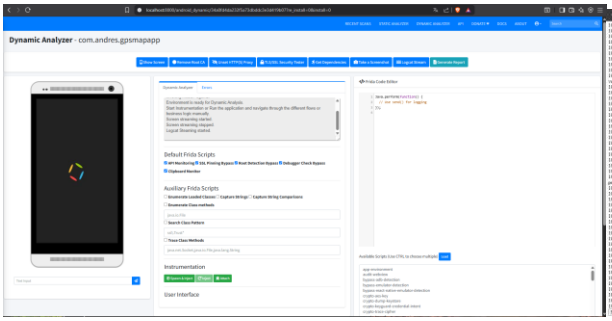
Nivel	Cantidad	Impacto
Alto	3	Altamente peligroso
Medio	1	Peligroso
Bajo	1	Poco riesgo

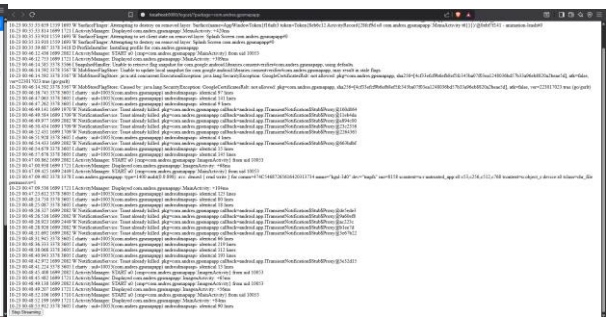
Evidencias











Aplicacion de best practice

1.- Asegurar el Código e infraestructura:

Intención: Evitar que terceros manipulen, extraigan informacion sensible y modifiquen el comportamiento de la aplicación.

Desactivar el modo debug y así evitar depurar la aplicación y leer registros desde un celular.

Dentro de Android Manifest declaramos expresamente el modo debug como false:

```
<application
    android:debuggable="false"
```

2.- Asegurar la comunicacion de red(HTTPS)

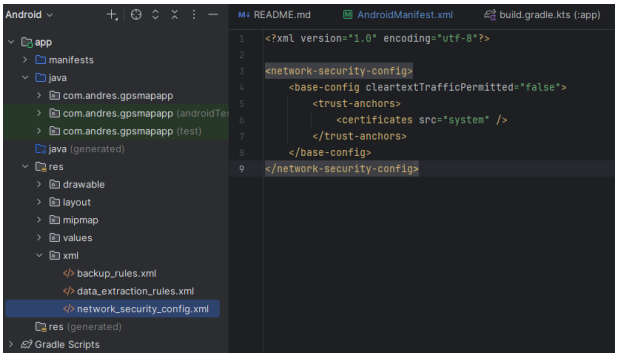
Intención: Evitar que los datos viajen sin seguridad y sean interceptados por terceros.

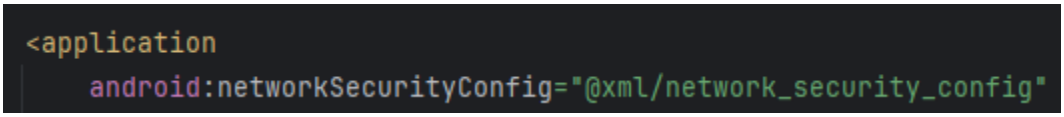
-Asegurarse que siempre que se hagan solicitudes a algun sitio que cuente con HTTPS:

```
public void cargarImagenEnHiloSecundario() throws MalformedURLException {
    URL url = new URL( spec: "https://upload.wikimedia.org/wikipedia/commons/thumb/
```

-Bloquear el uso de HTTP en toda la app:

Cree un archivo en res.xml donde se realiza la configuración de seguridad de red, y se llama desde el Android manifest, asi nos aseguramos que todas las conexiones de la app sean seguras

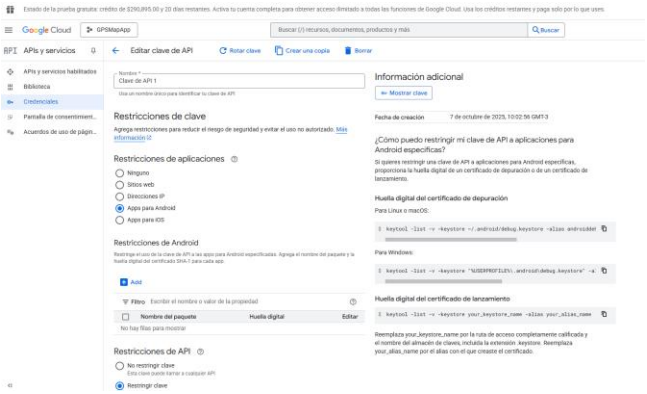




En la actualidad el alcance de mi app no es muy amplio, ya que no cuenta con base de datos, sistemas de autenticación y autorización o manipulación de datos compleja, por lo que algunos tips de seguridad no son adecuados en la etapa actual, pero son vitales si pretendo que mi app escale.

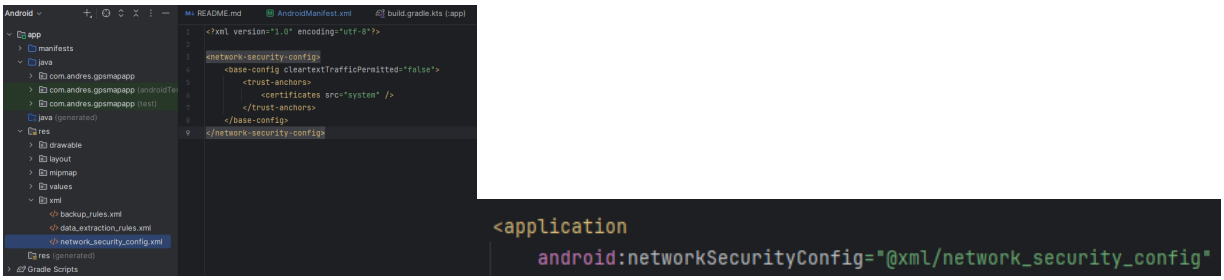
Tip 1: Autenticación y autorización: si bien mi app no cuenta con usuarios ni roles, si utiliza el servicio de maps de Google, que utiliza autenticación con un api key que debemos proteger.

Para esto utilizamos la plataforma de Google cloud para restringir nuestra api key, tanto a tipo de dispositivos como para aplicaciones específicas, además de solicitar identificación al usuario para poder utilizarla.

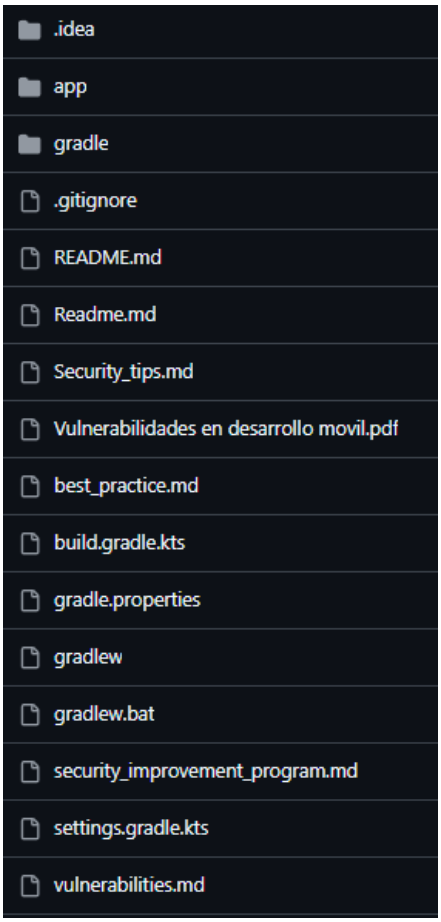


Tip 2: Proteger contra ataques de red MITM: Esto sirve para proteger los datos que son enviados desde nuestra aplicación hacia internet y viceversa.

Nuestra aplicación descarga una imagen de internet y utiliza los servicios de google maps. La medida que debemos tomar es utilizar el protocolo HTTPS y no HTTP, esto ya ha sido realizado por lo que nuestra comunicación con internet es mas segura y evitar asi ataques del tipo “Man in the middle”.



Readmes



Conclusion

Preguntas de cierre

1. ¿Qué nuevos conocimientos y habilidades has adquirido en la protección de aplicaciones Android contra amenazas de seguridad?

R: Primero, es saber que existen algunos tipos de amenazar, herramientas que pueden detectarlas y formas de afrontarlas, ya que de esto depende la seguridad de 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?

R: Bajo ese contexto, es fundamental utilizar todas las herramientas que se tengan a mano para asegurar de la mejor forma la seguridad de la app y todo el que la use, para ello , la herramienta MosSF es fundamental, ya que detecta vulnerabilidades que pueden ser aprovechadas por terceros, pero no basta solo con esta app, en casos como bancos, comercio y salud, es fundamental buscar mas formas de combatir la inseguridad, como auditorias con profesionales especialisados.

3. ¿Lograste implementar las mejores prácticas y los consejos de seguridad de manera efectiva? ¿Qué desafíos encontraste en el proceso y cómo los superaste?

R: Si, logra implementar las mejores practicas y consejos de seguridad entendiendo la limitación de funcionalidades que tenia la app, y el mayor desafío, fue sin duda la instalación y configuración de la herramienta MosSF, ya que dependía de configuraciones y programas externos, además de contar con un dispositivo o emulador rooteado