SANTO TOMÁS

INSTITUTO PROFESIONAL
CENTRO DE FORMACIÓN TÉCNICA

Vulnerabilidades en desarrollo movil

Nombre: Andres Pasten Rodriguez.

Carrera: Analista Programador.

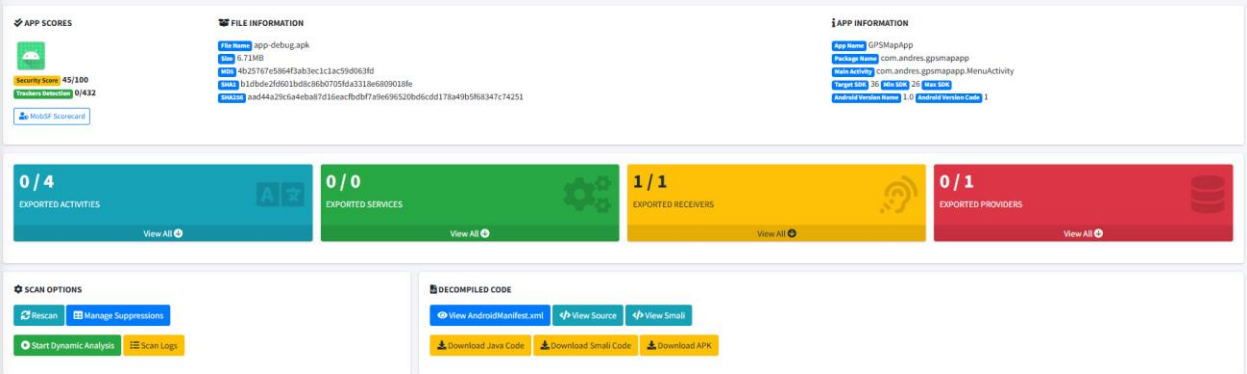
Profesor: Oscar Monardez.

Analisis de vulnerabilidades

1. Identificacion de vulnerabilidades

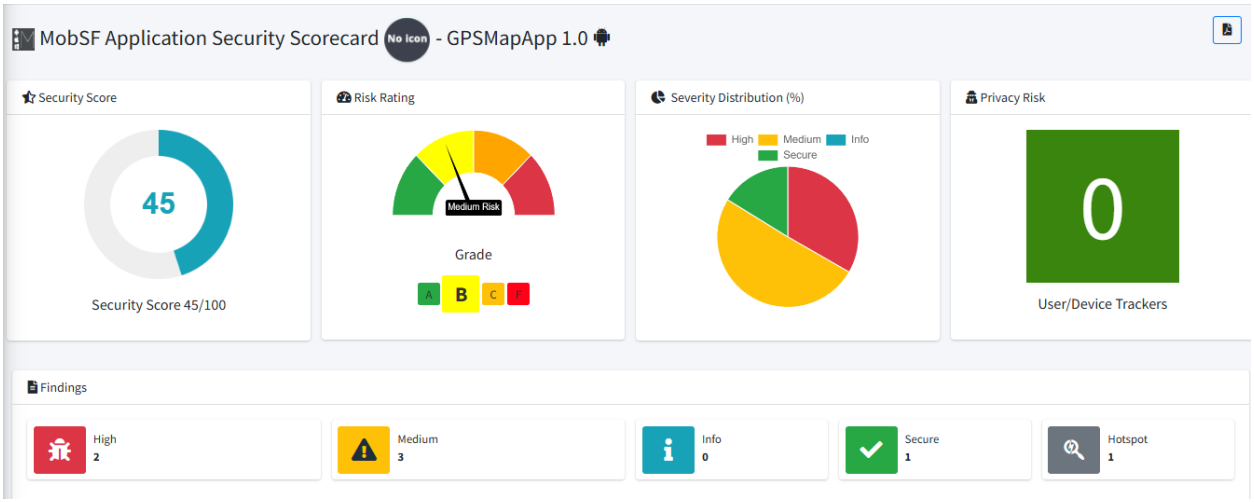
Herramienta de análisis estático utilizada: MobSF.

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.



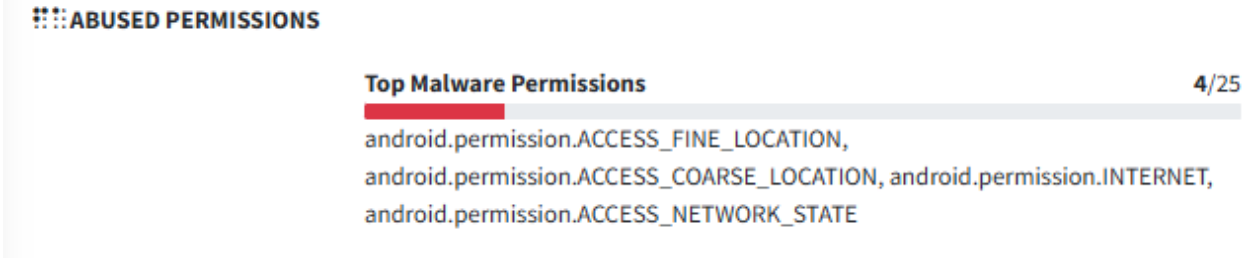
Permisos

-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 aplicacion 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.

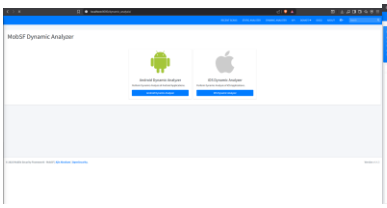
Vulnerabilidades bajas:

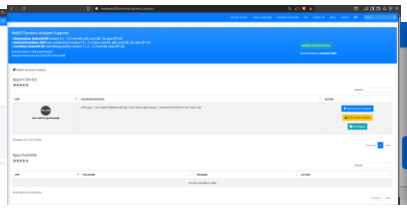
Modo debbug: La aplicación tiene la depuración activada, por lo que terceros pueden acceder a logs (registros), corriendo el riesgo de exponer datos privados o información confidencial.

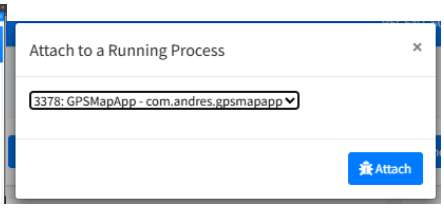
Evaluaciones de los riesgos

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

Evidencias







SANTO TOMÁS

INSTITUTO PROFESIONAL
CENTRO DE FORMACIÓN TÉCNICA