



Android Penetration Testing

MOBSF

FRAMEWORK

Contenido

Abstracto.....	3
Instalación	4
Explorando MobSF	6
Página de destino	6
Análisis del certificado de firmante.....	8
Permisos de aplicación	8
Actividades navegables y análisis de seguridad de red.....	9
Análisis de manifiesto.....	11
Análisis de código.....	12
Relacionado con el análisis de malware	13
Secretos codificados.....	17
Componentes de la actividad presentes.....	17
Analizador dinámico.....	20
Secuencia Logcat	24
Monitoreo API.....	25
Descarga de informes.....	26

Abstracto

MobSF es una herramienta de código abierto desarrollada por Ajin Abraham que se utiliza para el análisis automatizado de un APK. Esta es una colección de herramientas que se ejecutan en una interfaz, realizan sus propias tareas individuales (como Jadx, apktool, etc.) y muestran sus resultados en una interfaz común. Estos informes también se pueden descargar en formato PDF y también brindan análisis detallados con las capturas de pantalla necesarias. Puedes descargar MobSF [aquí](#). En esta publicación, recorreremos la fase de instalación en el sistema operativo Ubuntu y lo guiaremos a través de varias opciones que esta herramienta tiene para ofrecer.

Instalación

Para instalar MobSF, cree un directorio y siga los comandos:

```
clon de git https://github.com/MobSF/Mobile-Security-Framework-MobSF.git
cd Mobile-Security-Framework-MobSF
```

```
root@hex:/home/hex/android-toolkit# git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git
Cloning into 'Mobile-Security-Framework-MobSF'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 16859 (delta 0), reused 1 (delta 0), pack-reused 16856
Receiving objects: 100% (16859/16859), 1.09 GiB | 7.64 MiB/s, done.
Resolving deltas: 100% (8022/8022), done.
root@hex:/home/hex/android-toolkit# cd Mobile-Security-Framework-MobSF/
root@hex:/home/hex/android-toolkit/Mobile-Security-Framework-MobSF#
```

Necesitamos instalar dependencias antes de poder ejecutar:

```
apt-get instala python3-venv
instalación de pip3 -r requisitos.txt
```

```
root@hex:/home/hex/android-toolkit/Mobile-Security-Framework-MobSF# apt-get install python3-venv
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3-venv is already the newest version (3.8.2-0ubuntu2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@hex:/home/hex/android-toolkit/Mobile-Security-Framework-MobSF# pip3 install -r requirements.txt
Ignoring waitress: markers 'platform_system == "Windows"' don't match your environment
Requirement already satisfied: Django>=3.1.5 in /usr/local/lib/python3.8/dist-packages (from -r requirements.txt (line 1)) (3.1.7)
Requirement already satisfied: lxml>=4.6.2 in /usr/local/lib/python3.8/dist-packages (from -r requirements.txt (line 2)) (4.6.2)
```

Una vez hecho esto, podemos ejecutar el archivo de instalación para instalar MobSF y todos los componentes automáticamente.

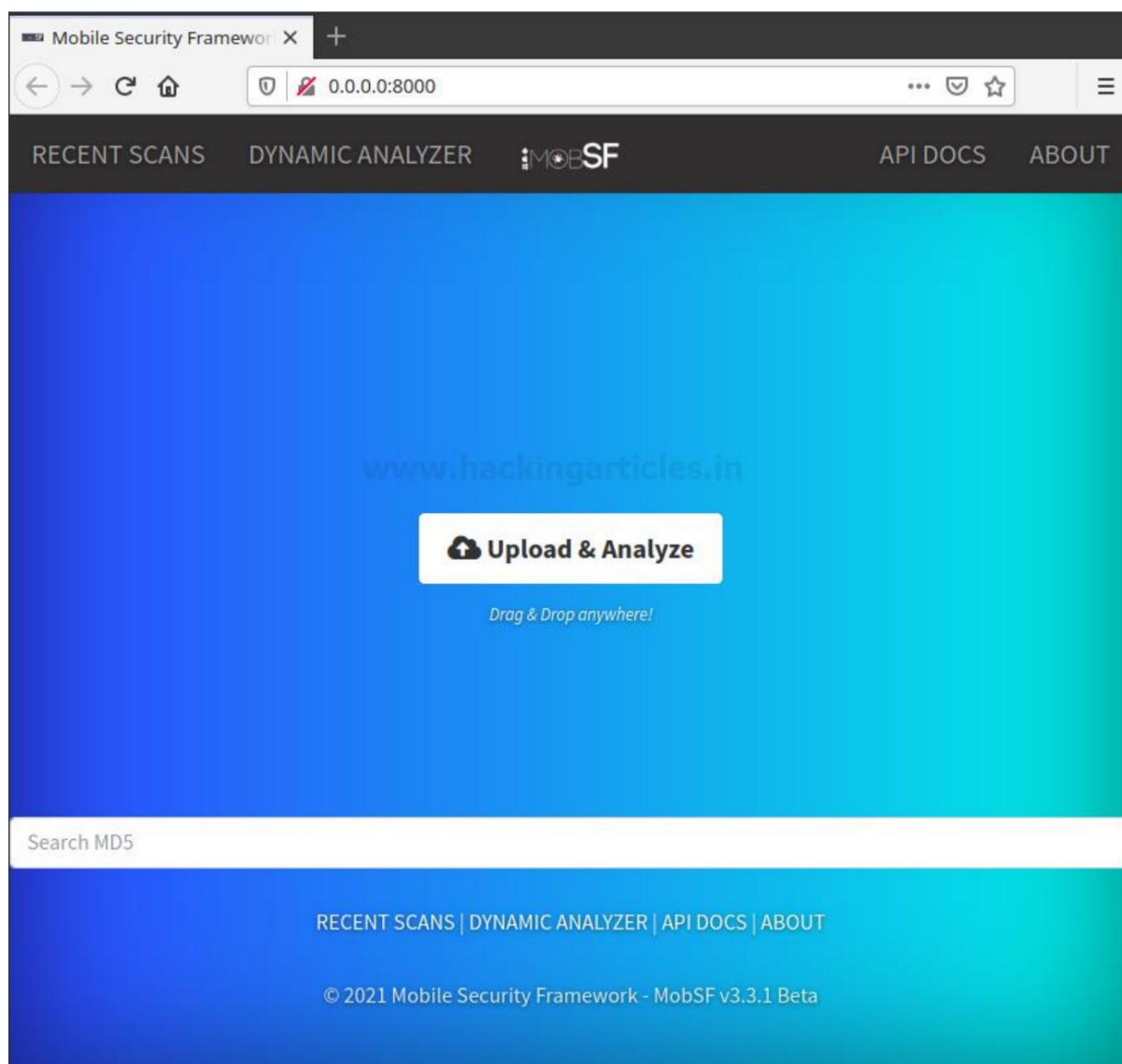
```
./setup.sh
```

```
root@hex:/home/hex/android-toolkit/Mobile-Security-Framework-MobSF# ./setup.sh
[INSTALL] Found Python 3.8.5
pip 21.0.1 from /root/.local/lib/python3.8/site-packages/pip (python 3.8)
[INSTALL] Found pip
Requirement already satisfied: pip in /root/.local/lib/python3.8/site-packages (21.0.1)
[INSTALL] Using python virtualenv
```


Ahora, para ejecutar MobSF ejecutamos el archivo run.sh. Como se puede interpretar en la captura de pantalla siguiente, MobSF se ejecutaría en un servidor local en el puerto 8000.

```
root@hex:/home/hex/android-toolkit/Mobile-Security-Framework-MobSF# ./run.sh
[2021-02-22 21:15:01 +0530] [15422] [INFO] Starting gunicorn 20.0.4
[2021-02-22 21:15:01 +0530] [15422] [INFO] Listening at: http://0.0.0.0:8000 (15422)
[2021-02-22 21:15:01 +0530] [15422] [INFO] Using worker: threads
[2021-02-22 21:15:01 +0530] [15424] [INFO] Booting worker with pid: 15424
```

Ahora abramos el enlace en el navegador y veamos si MobSF se instaló correctamente o no.



Explorando MobSF

Página de destino

Ahora que MobSF está en funcionamiento, podemos arrastrar un APK ficticio (en este caso, llevaré InjuredAndroid de Kyle Benac (aquí) a la interfaz de MobSF y veré qué sucede. Después de esperar un par de minutos, pudimos ver "Se realiza el análisis estático del APK. Ahora, aquí en la página de inicio, podemos ver que se proporciona una puntuación de gravedad. Cuanto más alta sea esta puntuación, más segura será la aplicación. A continuación, también se proporcionan los hashes, el nombre de archivo y el tamaño del APK. En la tercera columna de la primera fila, también podemos ver el nombre del paquete, la actividad principal, la versión mínima del SDK y la versión de la aplicación. También se proporciona la descripción de la aplicación.

The screenshot displays the MobSF Static Analysis web interface. The browser address bar shows the URL: `0.0.0.0:8000/static_analyzer/?name=InjuredAndroid-1.0.10-release.apk&ch`. The interface is divided into several sections:

- RECENT SCANS**, **STATIC ANALYZER**, **DYNAMIC ANALYZER**, **API DOCS**, and **ABOUT** are visible in the top navigation bar.
- APP SCORES**: Shows an Android icon, Average CVSS 7.0, Security Score 70/100, and Trackers Detection 0/343.
- FILE INFORMATION**:
 - File Name: InjuredAndroid-1.0.10-release.apk
 - Size: 17.49MB
 - MD5: c7c91424ab41179ead7186cf586c8a3d
 - SHA1: 47d86e733821cec0cb5dae8eab68edde0de10
 - SHA256: 5e111b2bdcd9c9550737ec468bd30ddfb6e4af40bf61a573c775a27d61eae02
- APP INFORMATION**:
 - App Name: InjuredAndroid
 - Package Name: b3nac.injuredandroid
 - Main Activity: b3nac.injuredandroid.MainActivity
 - Target SDK: 29, Min SDK: 21, Max SDK: 29
 - Android Version Name: 1.0.9
 - Android Version Code: 17
- PLAYSTORE INFORMATION**:
 - Title: InjuredAndroid
 - Score: 0.0, Installs: 100+, Price: 0
 - Android Version Support: 5.0 and up
 - Category: Education
 - Play Store URL: [b3nac.injuredandroid](https://play.google.com/store/apps/details?id=b3nac.injuredandroid)
 - Developer: B3nac, Developer ID: B3nac
 - Developer Address: None
 - Developer Website: <https://b3nac.com/>
 - Developer Email: b3nac.sec@gmail.com
 - Release Date: Jul 20, 2020
 - Privacy Policy: [Privacy link](#)
 - Description:
 - Setup for a physical device
 - 1. Download the latest release from Google Play.
 - Setup for an Android Emulator using Android Studio
 - 1. Pull the apk from a physical device after installing from Google Play with `adb pull`.

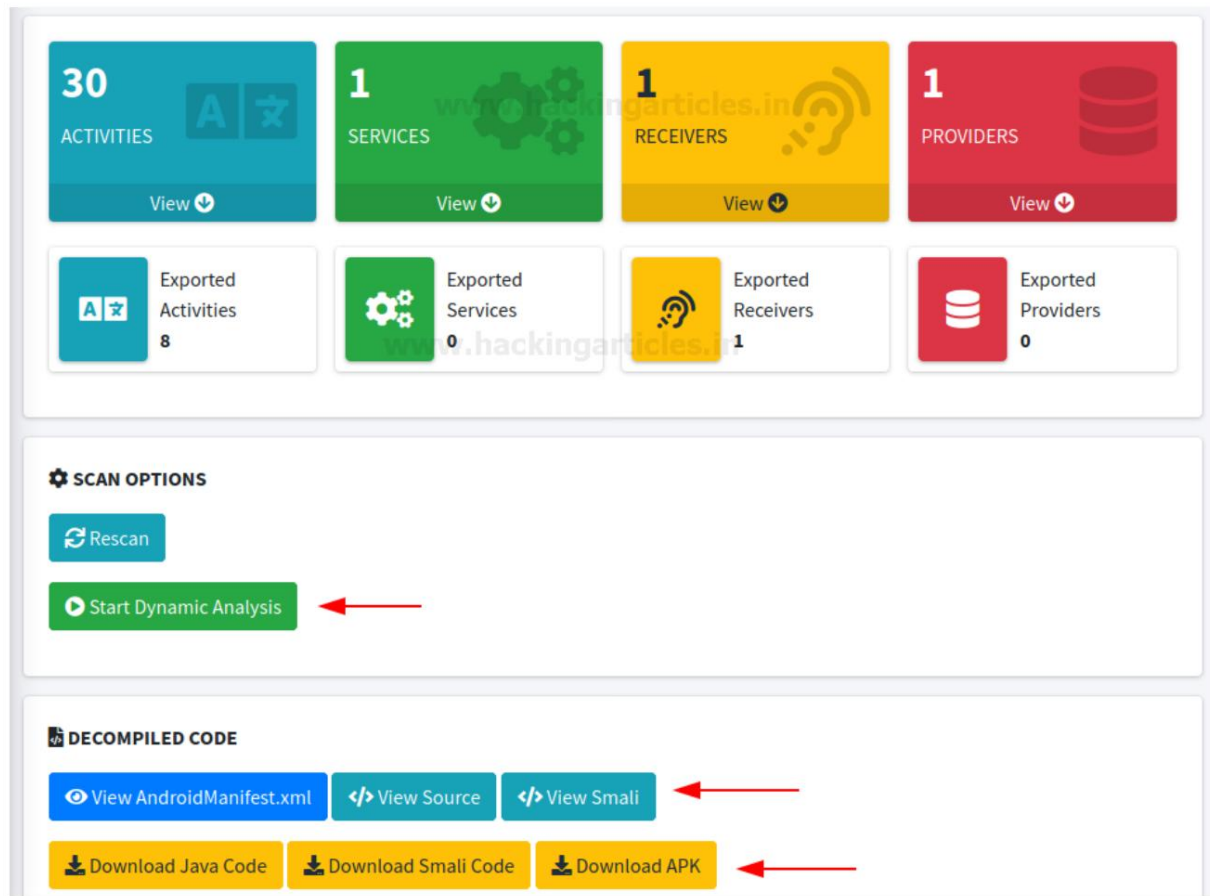
Después de desplazarnos un poco hacia abajo, esto es todo lo que podemos ver:

En tarjetas pequeñas vemos diferentes componentes de la aplicación.

Opción de análisis dinámico que ayudará a MobSF a realizar análisis de tiempo de ejecución. Opción para ver el código descompilado.

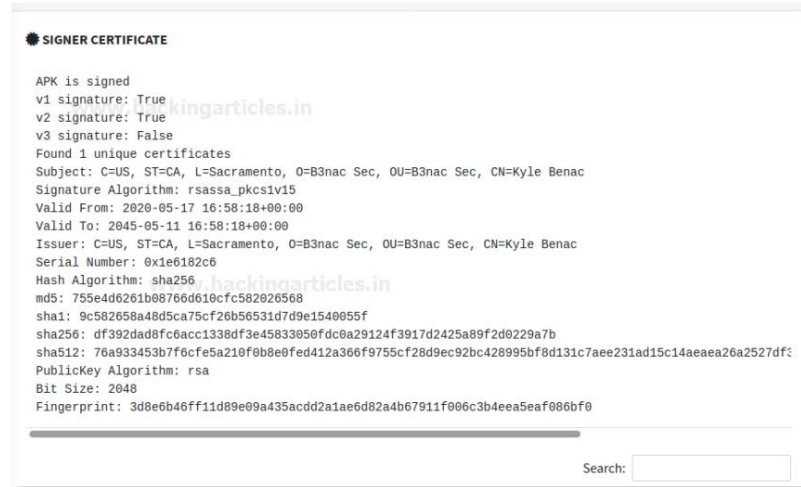
Este es el código generado por apktool. Generalmente, el archivo de recursos también se decodificaría.

También es posible ver código pequeño. Hace que sea más fácil segregar y ver el código fuente en clases Java separadas usando esto.



Análisis del certificado de firmante

En la columna del certificado podemos ver el certificado del firmante donde se puede encontrar información importante sobre el desarrollador, país, estado, tipo de algo, tamaño de bits, etc.



Permisos de aplicación

Además, podemos ver todos los permisos que tiene una aplicación. Existen varios permisos que se clasifican como peligrosos o normales. Desde el punto de vista de un analista de seguridad, es importante comprender qué permisos pueden provocar daños mayores. Por ejemplo, si una aplicación tiene acceso a medios externos y almacena información crítica en los medios externos, podría resultar peligrosa ya que los archivos almacenados en medios externos se pueden leer y escribir globalmente.

APPLICATION PERMISSIONS

Search:

PERMISSION	STATUS	INFO	DESCRIPTION
android.permission.ACCESS_NETWORK_STATE	normal	view network status	Allows an application to view the status of all networks.
android.permission.INTERNET	normal	full Internet access	Allows an application to create network sockets.
android.permission.READ_EXTERNAL_STORAGE	dangerous	read external storage contents	Allows an application to read from external storage.
android.permission.READ_PHONE_STATE	dangerous	read phone state and identity	Allows the application to access the phone features of the device. An application with this permission can determine the phone number and serial number of this phone, whether a call is active, the number that call is connected to and so on.
android.permission.WRITE_EXTERNAL_STORAGE	dangerous	read/modify/delete external storage contents	Allows an application to write to external storage.

Showing 1 to 5 of 5 entries

Previous 1 Next

Actividades navegables y análisis de seguridad de red

A continuación, en la sección de actividades navegables, podemos ver todas las actividades que han implementado un esquema de enlace profundo. Consulte el artículo [aquí](#) para comprender todo sobre los enlaces profundos, su implementación y explotación.

En la sección de seguridad de la red, se pueden encontrar algunos detalles sobre problemas de seguridad de la red relacionados con la aplicación. Estos problemas pueden provocar a veces ataques críticos como MiTM. Por ejemplo, en la captura de pantalla siguiente, se puede encontrar que la aplicación no utiliza el mecanismo de fijación SSL implementado.

BROWSABLE ACTIVITIES

Search:

ACTIVITY ↑↓	INTENT ↑↓
b3nac.injuredandroid.CSPBypassActivity	Schemes: http://, https://, Hosts: b3nac.com, Path Patterns: /*,
b3nac.injuredandroid.DeepLinkActivity	Schemes: flag11://, https://,
b3nac.injuredandroid.RCEActivity	Schemes: flag13://, Hosts: rce,

Showing 1 to 3 of 3 entries

Previous **1** Next

NETWORK SECURITY

Search:

NO ↑↓	SCOPE ↑↓	SEVERITY ↑↓	DESCRIPTION ↑↓
1	*	high	Base config is insecurely configured to permit clear text traffic to all domains.

Showing 1 to 1 of 1 entries

Previous **1** Next

Análisis manifiesto

En la siguiente sección, MobSF analizó el archivo de manifiesto. Se pueden encontrar muchos pliegues de información en el archivo de manifiesto de Android, como qué actividades se exportan, si la aplicación se puede depurar o no, esquemas de datos, etc. Como referencia, mire la captura de pantalla a continuación.

MANIFEST ANALYSIS			
Search: <input type="text"/>			
NO ↑↓	ISSUE ↑↓	SEVERITY ↑↓	DESCRIPTION ↑↓
1	App has a Network Security Configuration [android:networkSecurityConfig=@xml/network_security_config]	Info	The Network Security Configuration feature lets apps customize their network security settings in a safe, declarative configuration file without modifying app code. These settings can be configured for specific domains and for a specific app.
2	Application Data can be Backed up [android:allowBackup] flag is missing.	medium	The flag [android:allowBackup] should be set to false. By default it is set to true and allows anyone to backup your application data via adb. It allows users who have enabled USB debugging to copy application data off of the device.
3	Activity (b3nac.injuredandroid.CSPBypassActivity) is not Protected. An intent-filter exists.	high	An Activity is found to be shared with other apps on the device

Análisis de código

Una de las características más interesantes de la herramienta MobSF es la sección de análisis de código. En esta sección, podemos ver que MobSF analizó y comparó algunos comportamientos de la aplicación basándose en prácticas estándar de seguridad de la industria como OWASP MSTG y mapeó las vulnerabilidades con OWASP Top 10. Es interesante ver que se menciona CWE y se asigna aquí una puntuación CVSS que podría ayudar en varios escenarios de analistas y facilitar la creación de informes.

CODE ANALYSIS				
Search: <input type="text"/>				
NO	ISSUE	SEVERITY	STANDARDS	FILES
1	App uses SQLite Database and execute raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection. Also sensitive information should be encrypted and written to the database.	high	CVSS V2: 5.9 (medium) CWE: CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') OWASP Top 10: M7: Client Code Quality	b3nac/injuredandroid/f.java
2	The App logs information. Sensitive information should never be logged.	Info	CVSS V2: 7.5 (high) CWE: CWE-532 Insertion of Sensitive Information into Log File OWASP MASVS: MSTG-STORAGE-3	b/c/c/c.java b3nac/injuredandroid/FlagFiveReceiver.java b3nac/injuredandroid/k.java a/i/b/c.java b/c/a/a/b/d.java b3nac/injuredandroid/AssemblyActivity.java b/c/a/b/y/b.java b/c/a/a/d/c/q1.java a/g/e/g.java io/flutter/view/c.java a/g/j/b.java b/c/a/b/n/a.java a/g/d/d/b.java c/a/a.java a/g/l/b.java a/a/n/g.java

Relacionado con el análisis de malware

MobSF también alberga una sección donde se ofrece un análisis de APKiD. APKiD es una herramienta de código abierto que es muy útil para identificar varios empaquetadores, compiladores, ofuscadores, etc. en archivos de Android. Es análogo a PEiD en APK. Aquí se puede ver que ha detectado un código anti-vm en el APK.

FILE ANALYSIS

Search:

NO	ISSUE	FILES
No data available in table		

Showing 0 to 0 of 0 entries

Previous Next

APKiD ANALYSIS

Search:

DEX	DETECTIONS						
classes.dex	<div>Search: <input type="text"/></div> <table> <thead> <tr> <th>FINDINGS</th> <th>DETAILS</th> </tr> </thead> <tbody> <tr> <td>Anti-VM Code</td> <td>Build.MANUFACTURER check</td> </tr> <tr> <td>Compiler</td> <td>unknown (please file detection issue!)</td> </tr> </tbody> </table> <div>Showing 1 to 2 of 2 entries</div> <div>Previous 1 Next</div>	FINDINGS	DETAILS	Anti-VM Code	Build.MANUFACTURER check	Compiler	unknown (please file detection issue!)
FINDINGS	DETAILS						
Anti-VM Code	Build.MANUFACTURER check						
Compiler	unknown (please file detection issue!)						

Showing 1 to 1 of 1 entries

Previous 1 Next

Algo relacionado con el análisis de malware es la función de verificación de malware de dominio. Aquí, MobSF extrae todas las URL/direcciones IP que están codificadas o que se utilizan en la aplicación y muestra su estado de malware y también utiliza ip2location para proporcionar su geolocalización.

DOMAIN MALWARE CHECK

Search:

DOMAIN	STATUS	GEOLOCATION
github.com	good	IP: 13.234.176.102 Country: India Region: Maharashtra City: Mumbai Latitude: 19.01441 Longitude: 72.847939 View: Google Map
injuredandroid.firebaseio.com	good	IP: 35.201.97.85 Country: United States of America Region: Missouri City: Kansas City Latitude: 39.099731 Longitude: -94.578568 View: Google Map
m.do.co	good	IP: 104.21.61.61 Country: United States of America Region: California City: San Francisco Latitude: 37.7757 Longitude: -122.395203 View: Google Map

También está disponible un análisis completo de cadenas. Las personas que conocen el análisis de malware conocen las cadenas en profundidad, pero para aquellos que no, las cadenas son secuencias de caracteres imprimibles en ASCII y Unicode incrustadas en un archivo. La extracción de cadenas puede dar pistas sobre la funcionalidad del programa y los indicadores asociados con un binario sospechoso. Por ejemplo, si un APK muestra algo como salida, se llamará a esa secuencia y, por lo tanto, se mostrará en las cadenas. Esto no es lo mismo que el archivo strings.xml. Muchas veces, aquí se hace visible una dirección IP de un tercero con la que se comunica APK. Esto es esencial desde el punto de vista del análisis de malware.

A STRINGS

```

id-aes256-wrap
lcircumflexsmall
half4 color = half4(%s, %s, %s, %s);
"abc_searchview_description_query" : "ಪ್ರಶ್ನೆಯನ್ನು ಹುಡುಕಿ"
_input
"common_google_play_services_install_button" : "Инсталиране"
blend_dst_in
experimental
Private_Use_Area
float2 ab = mix(P[0], P[1], T);
Math_Alphanum
Dart_NewStringFromCString
Khojki
"abc_searchview_description_search" : "शोध"
Separator
UNABLE_TO_CREATE_NEW_SECTION
"abc_prepend_shortcut_label" : "Menú +"
"abc_capital_on" : "ಆನ್"
UNKNOWN_EXTENSION_NAME
../third_party/libcxxabi/src/abort_message.cpp
AHZzsm
[%-8s : sp(%#x) fp(%#x) pc(%#x) %s%s ]
"abc_activitychooserview_choose_application" : "Escolher uma aplicação"
GrRenderTargetContext::drawTextureSet
glBindAttribLocation
"common_google_play_services_install_text" : "%1$s нема да се извршува без услугите на Google Play што ги нем
"abc_searchview_description_clear" : "क्वेरी साफ करा"
Show invisible frames in stack traces.
noexcept
"abc_menu_sym_shortcut_label" : "Sym+"
sInt64List.
ENTITIES
"status_bar_notification_info_overflow" : "+999"
null-safety
Surrogate

```

Al igual que los correos electrónicos, las URL a menudo también se encuentran codificadas. Se pueden encontrar URL jugosas que se utilizan a veces. A menudo, los analistas encuentran que también se accede a URL maliciosas o incluso a un servidor C&C.

URLS

Search:

URL	FILE
http://localhost	b/c/a/a/d/c/a2.java
http://schemas.android.com/apk/res/android	a/g/d/d/g.java
http://www.w3.org/XML/1998/namespace data:application/dart data:application/dart; http://www.w3.org/2000/xmlns/ https://www.w3.org/Style/CSS/Test/Fonts/Ahem/).	lib/armeabi-v7a/libflutter.so
http://www.w3.org/XML/1998/namespace data:application/dart data:application/dart; http://www.w3.org/2000/xmlns/ https://www.w3.org/Style/CSS/Test/Fonts/Ahem/).	lib/arm64-v8a/libflutter.so
https://github.com/flutter/flutter/issues/2897).it	io/flutter/plugin/platform/i.java
https://injuredandroid.firebaseio.com	Android String Resource
https://m.do.co/c/9348bb7410b4	b3nac/injuredandroid/ContactActivity.java

Showing 1 to 7 of 7 entries

Previous

1

Next

Secretos codificados

A menudo, los desarrolladores tienen la costumbre de almacenar claves críticas como el ID de AWS y las credenciales en strings.xml y utilizar un objeto como referencia en la actividad de Java. Pero hacer esto no ayuda de ninguna manera ya que strings.xml se puede decodificar fácilmente.

POSSIBLE HARDCODED SECRETS

```
"AWS_ID" : "AKIAZ36DGKTUIOLDOBN6"  
"AWS_SECRET" : "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"  
"enter_password" : "Enter password"  
"firebase_database_url" : "https://injuredandroid.firebaseio.com"  
"flag_eight_aws" : "flag eight - aws"  
"flag_nine_firebase" : "flag nine - Firebase"  
"google_api_key" : "AlzaSyCUImEIOSvqAswLqFak75xhskkB6illd7A"  
"google_crash_reporting_api_key" : "AlzaSyCUImEIOSvqAswLqFak75xhskkB6illd7A"
```

Componentes de la actividad presentes

También se puede desplazar una lista de todas las actividades presentes usando MobSF. Esto da una idea del esqueleto del APK de Android. Además, a veces jadx reemplaza los nombres reales de la clase con alguna letra aleatoria si el desarrollador ha aplicado ofuscación, MobSF también puede asociar su nombre real (no sucede todo el tiempo o en casos de ofuscación fuerte).

ACTIVITIES

b3nac.injuredandroid.FlagSeventeenActivity
b3nac.injuredandroid.CSPBypassActivity
b3nac.injuredandroid.AssemblyActivity
io.flutter.embedding.android.FlutterActivity
b3nac.injuredandroid.RCEActivity
b3nac.injuredandroid.SettingsActivity
b3nac.injuredandroid.ExportedProtectedIntent
b3nac.injuredandroid.QXV0aA
b3nac.injuredandroid.FlagTwelveProtectedActivity
b3nac.injuredandroid.DeepLinkActivity
b3nac.injuredandroid.FlagTenUnicodeActivity
b3nac.injuredandroid.FlagOneLoginActivity
b3nac.injuredandroid.FlagNineFirebaseActivity
b3nac.injuredandroid.FlagEightLoginActivity
b3nac.injuredandroid.FlagSevenSqliteActivity
b3nac.injuredandroid.FlagsOverview
b3nac.injuredandroid.FlagSixLoginActivity
b3nac.injuredandroid.MainActivity
b3nac.injuredandroid.XSSTextActivity
b3nac.injuredandroid.DisplayPostXSS
b3nac.injuredandroid.FlagOneSuccess
b3nac.injuredandroid.b25IAActivity
b3nac.injuredandroid.FlagTwoActivity
b3nac.injuredandroid.FlagThreeActivity
b3nac.injuredandroid.FlagFourActivity
b3nac.injuredandroid.FlagFiveActivity
b3nac.injuredandroid.TestBroadcastReceiver
b3nac.injuredandroid.ContactActivity
com.google.firebase.auth.internal.FederatedSignInActivity
com.google.android.gms.common.api.GoogleApiActivity

De manera similar, un analista también puede recorrer servicios, transmisiones, proveedores y receptores de contenido junto con todos los archivos presentes en el archivo APK para crear un mapa de todos los recursos presentes en la aplicación.

SERVICES

com.google.firebase.components.ComponentDiscoveryService

RECEIVERS

b3nac.injuredandroid.FlagFiveReceiver

PROVIDERS

com.google.firebase.provider.FirebaseInitProvider

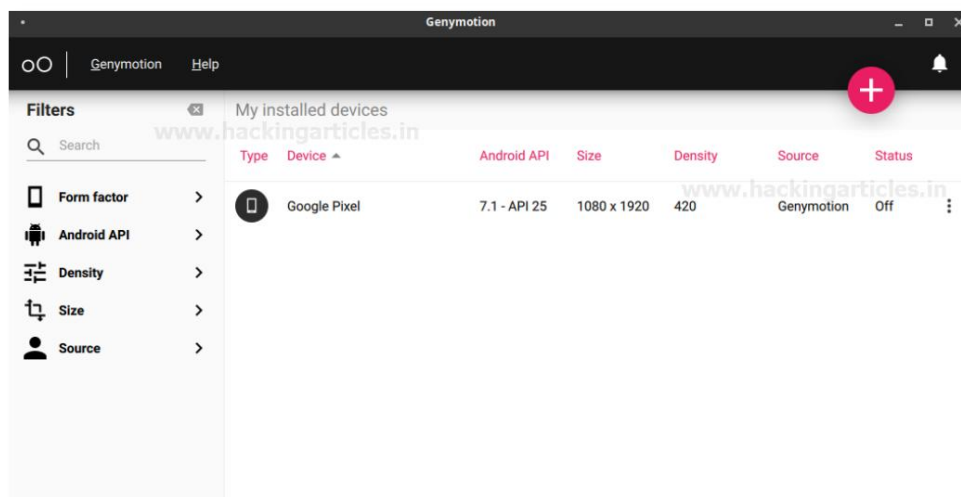
LIBRARIES

FILES

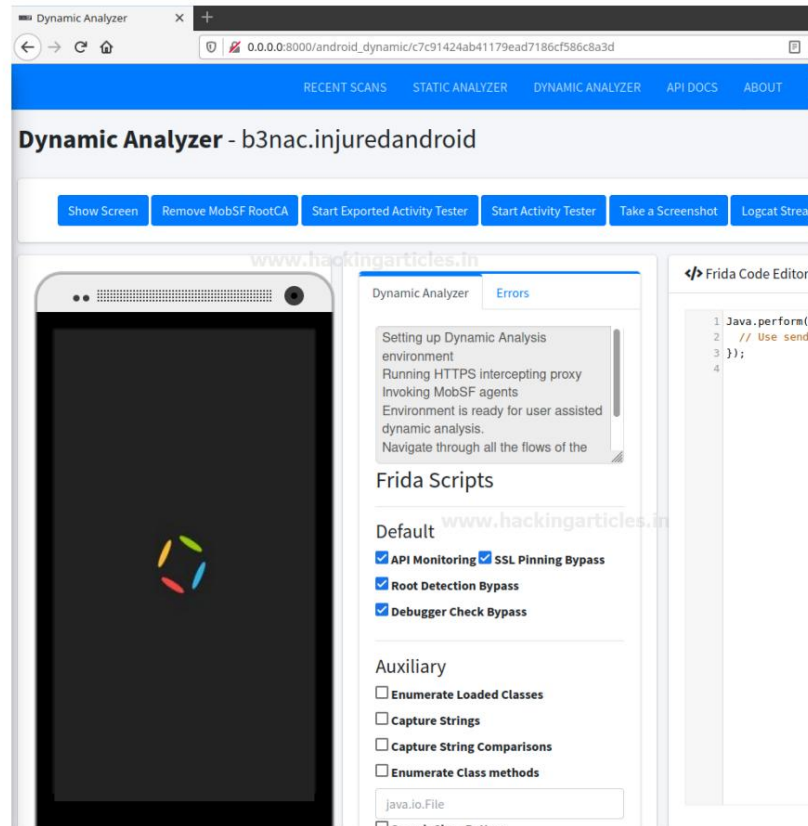
AndroidManifest.xml
META-INF/CERT.RSA
META-INF/CERT.SF
META-INF/MANIFEST.MF
META-INF/androidx.activity_activity.version
META-INF/androidx.appcompat_appcompat-resources.version
META-INF/androidx.appcompat_appcompat.version
META-INF/androidx.arch.core_core-runtime.version
META-INF/androidx.cardview_cardview.version
META-INF/androidx.coordinatorlayout_coordinatorlayout.version
META-INF/androidx.core_core-ktx.version
META-INF/androidx.core_core.version

Analizador dinámico

Para el análisis dinámico, primero necesitaríamos iniciar la VM de Android en genymotion. Aquí he creado una máquina virtual de Android en la versión 7.1.



Cuando presiona la opción de analizador dinámico presente en el panel de navegación superior, MobSF se conectará automáticamente a la VM en ejecución si MobSF y genymotion se ejecutan en la misma máquina base. Sin embargo, si MobSF está en otra máquina virtual, es posible que deba conectar el agente MobSF a la IP y al puerto remotos de la VM de genymotion. Una vez adjunto, vemos la siguiente pantalla.



Debajo de la barra de estado del analizador podemos ver varios scripts de Frida predeterminados disponibles que verificarían varias vulnerabilidades básicas, como la omisión de fijación de SSL y las comprobaciones de detección de raíz. Si no has leído sobre Frida, hazlo [ingresando aquí](#). También existen otros scripts auxiliares que permiten a un analista enumerar varias clases y también capturar comparaciones de cadenas en tiempo real (nuevamente útil desde el punto de vista de los analistas de malware). Luego simplemente haga clic en iniciar instrumentación y los scripts seleccionados se adjuntarán a la aplicación automáticamente. Por lo tanto, si he seleccionado el script de omisión de fijación de SSL y se captura el tráfico (visible en el registro o en el monitor API más adelante), eso significaría que se ha omitido la fijación de SSL.

Setting up Dynamic Analysis environment
Running HTTPS intercepting proxy
Invoking MobSF agents
Environment is ready for user assisted dynamic analysis.
Navigate through all the flows of the app manually.

www.hackingarticles.in

Frida Scripts

Default ←

☒ API Monitoring ☒ SSL Pinning Bypass ☒ Root Detection Bypass
☒ Debugger Check Bypass

Auxiliary ←

☒ Enumerate Loaded Classes ☒ Capture Strings
☒ Capture String Comparisons
☐ Enumerate Class methods
java.io.File
☐ Search Class Pattern
ssl.Trust*
☐ Trace Class Methods
java.net.Socket,java.io.File,java.lang.String

Start Instrumentation Frida Live Logs


↑

Ahora más, para analizar las actividades en busca de vulnerabilidades, se pueden ver dos botones en la parte superior para las actividades exportadas y no exportadas.

Dynamic Analyzer - b3nac.injuredandroid

www.hackingarticles.in

Show Screen Remove MobSF RootCA Start Exported Activity Tester Start Activity Tester Take a Screenshot



Dynamic Analyzer Errors

Setting up Dynamic Analysis environment
Running HTTPS intercepting proxy
Invoking MobSF agents
Environment is ready for user assisted dynamic analysis.
Navigate through all the flows of the app manually.
Instrumenting app with Frida
Successfully attached

Frida Scripts


Default

☒ API Monitoring ☒ SSL Pinning Bypass ☒ Root Detection Bypass
☒ Debugger Check Bypass

Auxiliary

☒ Enumerate Loaded Classes ☒ Capture Strings
☒ Capture String Comparisons
☐ Enumerate Class methods
java.io.File
☐ Search Class Pattern
ssl.Trust*

De manera similar, si uno no tiene que conformarse con guiones de Frida preconfigurados, también es posible pegar el guión de Frida en el cuadro de texto de la derecha. También hay un cuadro desplegable que cargaría esos scripts. También puedes editar lo mismo.


 Frida Code Editor

```

1 // https://codeshare.frida.re/@dzonerzy/aesinfo/
2
3 Java.perform(function () {
4     var complete_bytes = new Array();
5     var index = 0;
6     var secretKeySpecDef =
7     Java.use('javax.crypto.spec.SecretKeySpec');
8     var ivParameterSpecDef =
9     Java.use('javax.crypto.spec.IvParameterSpec');
10    var cipherDef = Java.use('javax.crypto.Cipher');
11    var cipherDoFinal_1 = cipherDef.doFinal.overload();
12    var cipherDoFinal_2 = cipherDef.doFinal.overload('[B');
13    var cipherDoFinal_3 = cipherDef.doFinal.overload('[B',
14    'int');
15    var cipherDoFinal_4 = cipherDef.doFinal.overload('[B',
16    'int', 'int');
17    var cipherDoFinal_5 = cipherDef.doFinal.overload('[B',
18    'int', 'int', '[B');
19    var cipherDoFinal_6 = cipherDef.doFinal.overload('[B',
20    'int', 'int', '[B', 'int');
21    var cipherUpdate_1 = cipherDef.update.overload('[B');
22    var cipherUpdate_2 = cipherDef.update.overload('[B', 'int',
23    'int');
24    var cipherUpdate_3 = cipherDef.update.overload('[B', 'int',
25    'int', '[B');
26    var cipherUpdate_4 = cipherDef.update.overload('[B', 'int',
27    'int', '[B', 'int');
28    var secretKeySpecDef_init_1 =
29    secretKeySpecDef.$init.overload('[B', 'java.lang.String');
30    var secretKeySpecDef_init_2 =
31    secretKeySpecDef.$init.overload('[B', 'int', 'int',

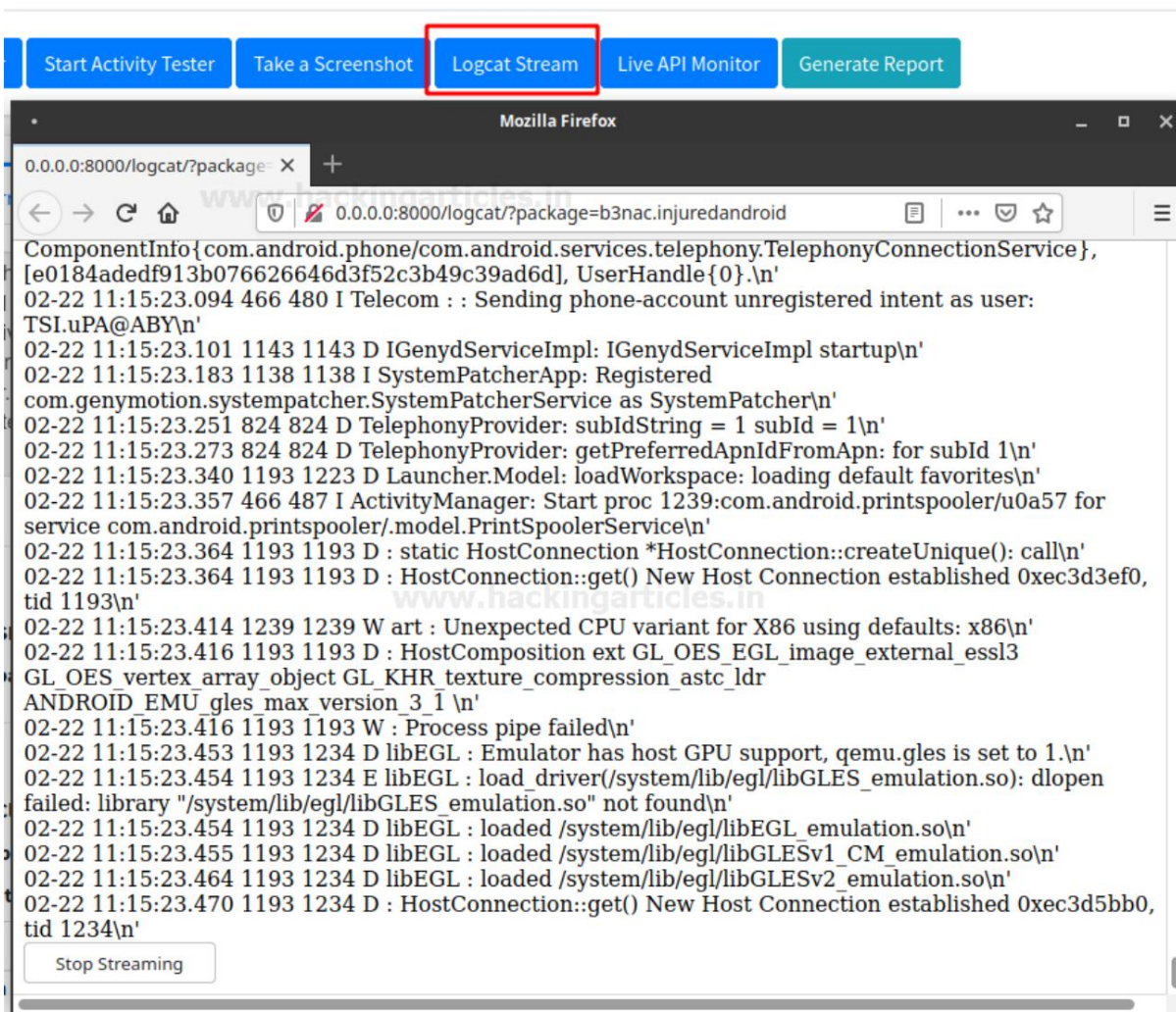
```

Available Scripts (Use CTRL to choose multiple)

- default
- helper
- jni_hook_by_address
- aes_key** 

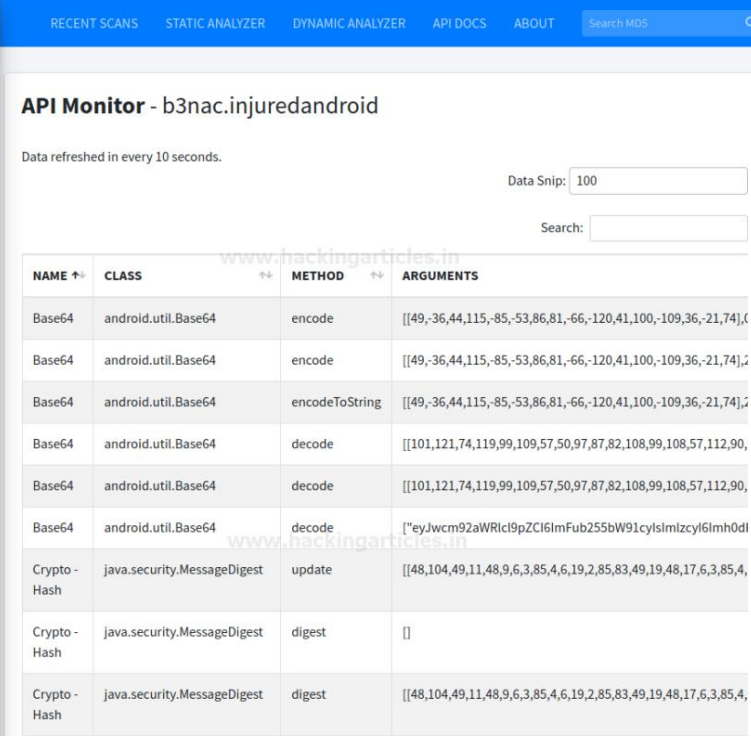
Corriente Logcat

Logcat también se puede ver en el entorno nativo de MobSF. Hay un botón en el menú superior que permite ver esto.



Monitor de API

Al igual que logcat monitorea los registros del dispositivo, las API también se pueden monitorear. Los APK utilizan varias API en tiempo real para realizar diversas funciones, por ejemplo, la biblioteca Base64.



RECENT SCANS STATIC ANALYZER DYNAMIC ANALYZER API DOCS ABOUT Search MD5

API Monitor - b3nac.injuredandroid

Data refreshed in every 10 seconds.

Data Snip:

Search:

NAME ↕	CLASS ↕	METHOD ↕	ARGUMENTS
Base64	android.util.Base64	encode	[[49,-36,44,115,-85,-53,86,81,-66,-120,41,100,-109,36,-21,74],{
Base64	android.util.Base64	encode	[[49,-36,44,115,-85,-53,86,81,-66,-120,41,100,-109,36,-21,74],{
Base64	android.util.Base64	encodeToString	[[49,-36,44,115,-85,-53,86,81,-66,-120,41,100,-109,36,-21,74],{
Base64	android.util.Base64	decode	[[101,121,74,119,99,109,57,50,97,87,82,108,99,108,57,112,90,
Base64	android.util.Base64	decode	[[101,121,74,119,99,109,57,50,97,87,82,108,99,108,57,112,90,
Base64	android.util.Base64	decode	["eyJwcm92aWRic9pZCj6ImFub255bW91cytsImlzcyl6Imh0dl
Crypto - Hash	java.security.MessageDigest	update	[[48,104,49,11,48,9,6,3,85,4,6,19,2,85,83,49,19,48,17,6,3,85,4,
Crypto - Hash	java.security.MessageDigest	digest	[]
Crypto - Hash	java.security.MessageDigest	digest	[[48,104,49,11,48,9,6,3,85,4,6,19,2,85,83,49,19,48,17,6,3,85,4,

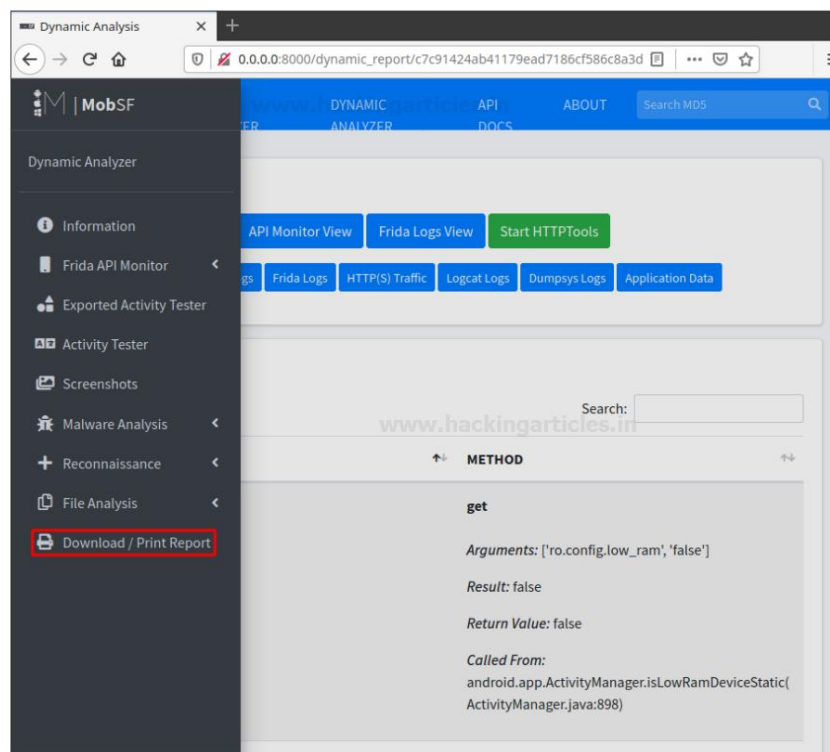
Por lo tanto, si una función utiliza esta API y descifra un valor, podemos ver ese valor aquí y decodificarlo.

Por ejemplo, a continuación puede ver el valor de retorno de una vez dicha función en Base64.

RESULT	RETURN VALUE
"[object Object]"	"77100119115995411876861087143105671081"
"[object Object]"	"77100119115995411876861087143105671081"
"MdWSC6vLVIG+iClkkyTrSg=="	"MdWSC6vLVIG+iClkkyTrSg=="
"[object Object]"	"12334112114111118105100101114951051003"
"[object Object]"	"12334112114111118105100101114951051003"
"[object Object]"	"12334112114111118105100101114951051003"

Descarga de informes

Una vez que haya realizado el análisis, es posible descargar el informe deslizando el control deslizante de la barra de menú en el lado izquierdo y haciendo clic en generar el informe.

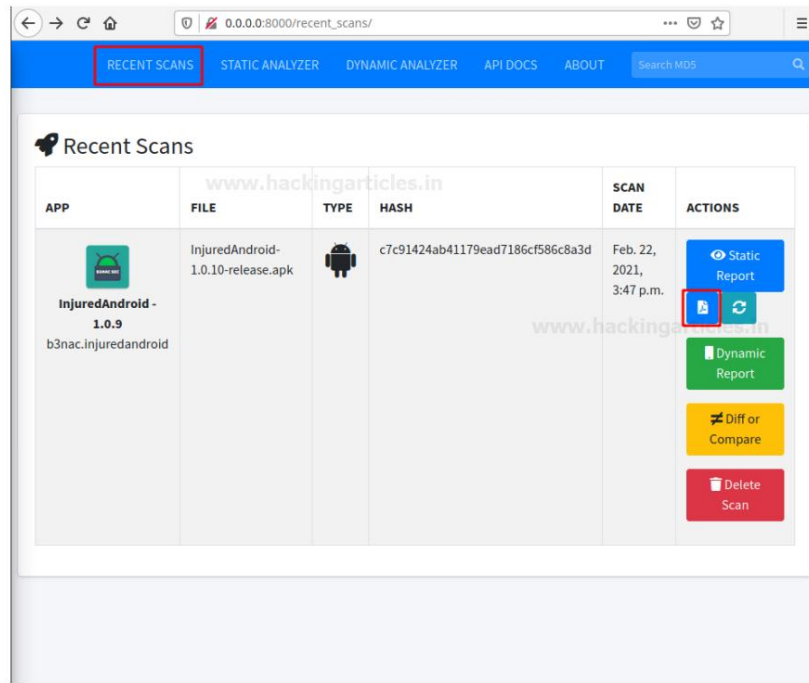


Es posible que observe algunos errores al generar informes. Para resolver esto, puede seguir el siguiente comando e instalar el módulo wkhtmltopdf :

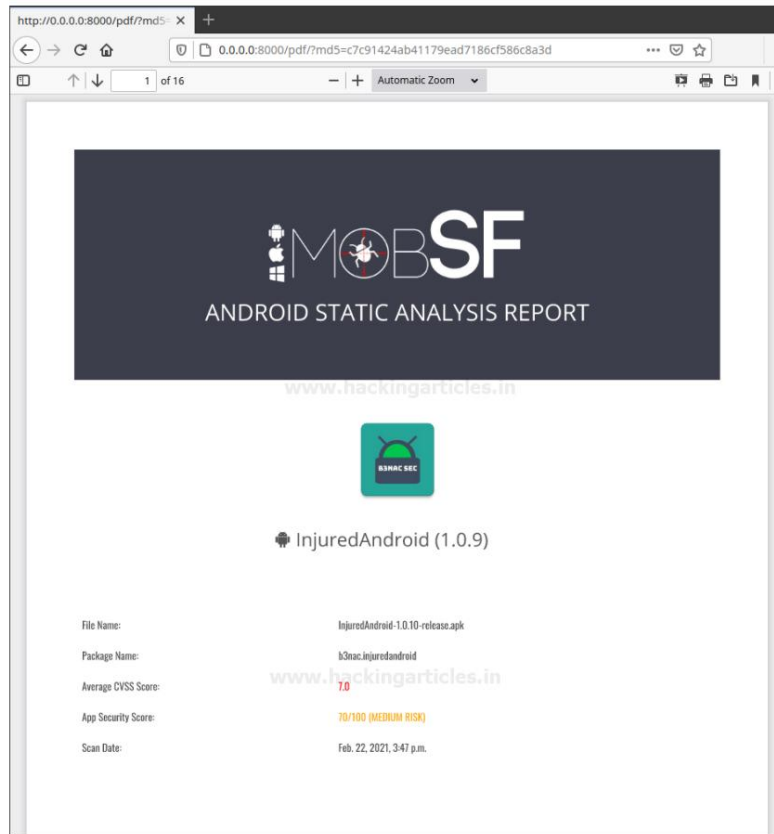
apt-get instalar wkhtmltopdf

```
root@hex:/home/hex# apt-get install wkhtmltopdf
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  wkhtmltopdf
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 203 kB of archives.
After this operation, 1,111 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 wkhtmltopdf amd64 0.12.5-1build1 [203 kB]
Fetched 203 kB in 1s (232 kB/s)
Selecting previously unselected package wkhtmltopdf.
(Reading database ... 462931 files and directories currently installed.)
Preparing to unpack .../wkhtmltopdf_0.12.5-1build1_amd64.deb ...
Unpacking wkhtmltopdf (0.12.5-1build1) ...
Setting up wkhtmltopdf (0.12.5-1build1) ...
Processing triggers for man-db (2.9.1-1) ...
root@hex:/home/hex#
```

Ahora, una vez más, si hace clic en una barra de análisis reciente, verá opciones de generación de informes estáticos y dinámicos.



El informe se parece a esto:



ÚNETE A NUESTRO PROGRAMAS DE ENTRENAMIENTO

