Antivirus Evasion Techniques

Pablo Gallegos, Usama Bin Ashraf

Motivation

Abstract

Antivirus detection and evasion has been a back and forth battle between attacker and defenders. This project outlines the evasion techniques in practice by the malicious actors to bypass antivirus for carrying the attack successfully. Many modern antivirus developers are crafting their defenses using the traditional mechanisms and also attackers on the other hand are developing evasion methodology by learning the defensive systems. The project merely focuses on the effectiveness of evasion tools relative to different defenders. Also we will be taking both offensive and defensive approach in research as well as results from both angles.

Keywords: antiviruses, evasion tools, kali linux/windows, malicious payloads, OS vulnerabilities, http server, Apache,

1. P. Casey, M. Topor, E. Hennessy, S. Alrabaee, M. Aloqaily and A. Boukerche, "Applied Comparative Evaluation of the Metasploit Evasion Module," *2019 IEEE Symposium on Computers and Communications (ISCC)*, Barcelona, Spain, 2019, pp. 1-6, doi: 10.1109/ISCC47284.2019.8969663.

2. F. A. Garba, K. I. Kunya, S. A. Ibrahim, A. B. Isa, K. M. Muhammad and N. N. Wali, "Evaluating the State of the Art Antivirus Evasion Tools on Windows and Android Platform," *2019 2nd International Conference of the IEEE Nigeria Computer Chapter (NigeriaComputConf)*, Zaria, Nigeria, 2019, pp. 1-4, doi: 10.1109/NigeriaComputConf45974.2019.8949637.

Objectives

- create malicious payloads and inject it to victim’s machine
- bypass the anti-virus software in defending machine
- deploy all possible evasion techniques (encryption, encoding, polymorphism, generating backdoor, manual binary editing, shellcode to windows executable conversion etc) [1] [2]
- determine the effectiveness of each tool and defense
- comparative analysis of tools and anti-viruses

Implementation

Virtualbox - on windows 10 host machine, 64 bit processor, 16Gb RAM, intel core i7

Offensive -

- we have set up an attacking machine - Kali Linux 2019, 4Mb memory
- Http server - Apache 2.4.35 Debian (on Kali platform) , to transfer files from one machine to another.
- However we could have used solid media for transfering files more conveniently but for the learning purposes the above method was suitable.
- Installed tools - creating malicious payloads and bypassing anti-viruses
- Hyperion 2.2, PeCloak.py, Veil Framework 3.0, FAT RAT, Phantom-Evasion, Binary-editing technique

Hyperion 2.2 - runtime encrypter for 32/64 bit portable executables by nullsecurity.net (access: https://github.com/nullsecuritynet/tools/raw/master/binary/hyperion/release/Hyperion-1.2.zip.

peCloak.py - avoids signature-based detection, it is been employed encoding with the constraints of the use only add ,sub ,XOR instructions and dynamic constructions of the encoding order. The number of instructions, the order and the modifiers are all chosen pseudo randomly to increase variation.

Veil Evasion 3.0 - generates random and unique payloads for payloads. The ability to make random changes to the payload is similar to polymorphic malwares that changes as it moves from host to host.

The Fat Rat- Easy tool to generate backdoor and easy tool to post exploitation attack like browser attack and etc, compiles a malware with popular payload, then the compiled malware can be execute on windows, android, mac, malware created with this tool have an ability to bypass most Antivirus softwares. (clone `https://github.com/Screetsec/TheFatRat.git`

Manual Binary editing [1] - identify the signature that is causing the binary to be flagged and change that portion without having to recompile the entire executable.

Manual Binary Editing

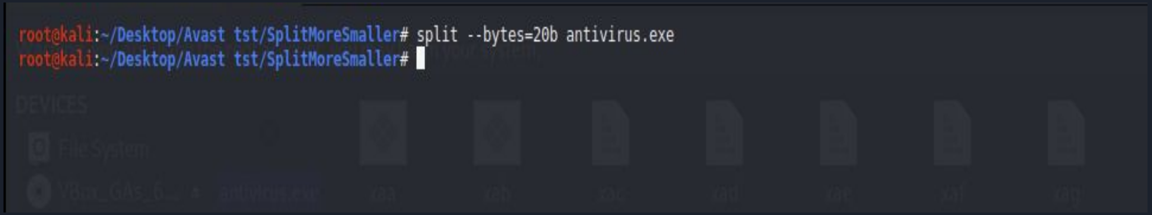
- Can be used to bypass the antivirus
- This can be achieved by finding the antivirus signature on the malware
- Once the signature it’s found a few bytes can be altered
- Can be facilitated by splittin the payload into smaller segments
- Not as effective most antivirus today used more advance scanning methods.

Binary Editing

Static Analysis & Dynamic Analysis

- In Dynamic Analysis we analyze the malware by executing it
- In Static Analysis we analyze the malware without execution
- We use both approaches for our experiment focusing more on Static Analysis
- With this method we are able to examine the binary to make an educated guess on which byte to modify

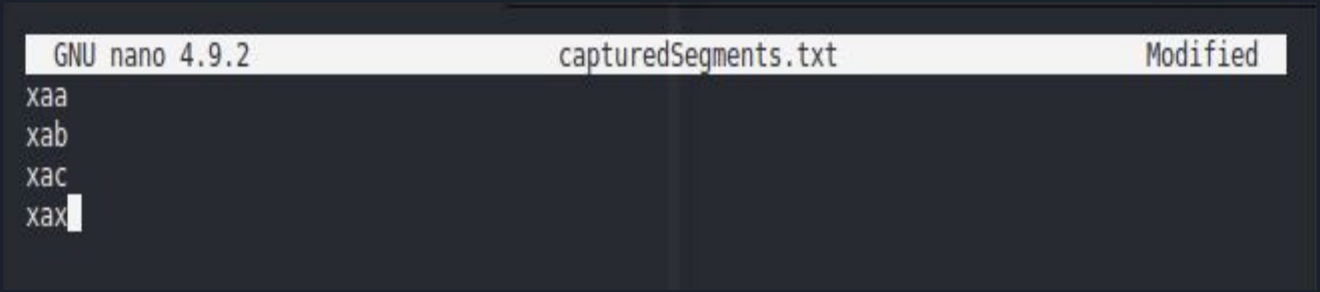
- Next we dissected the payload
- Payload size 505kb
- The malware was split into smaller segments of 20kb



```
root@kali:~/Desktop/Avast tst/SplitMoreSmaller# split --bytes=20b antivirus.exe
root@kali:~/Desktop/Avast tst/SplitMoreSmaller# 
```

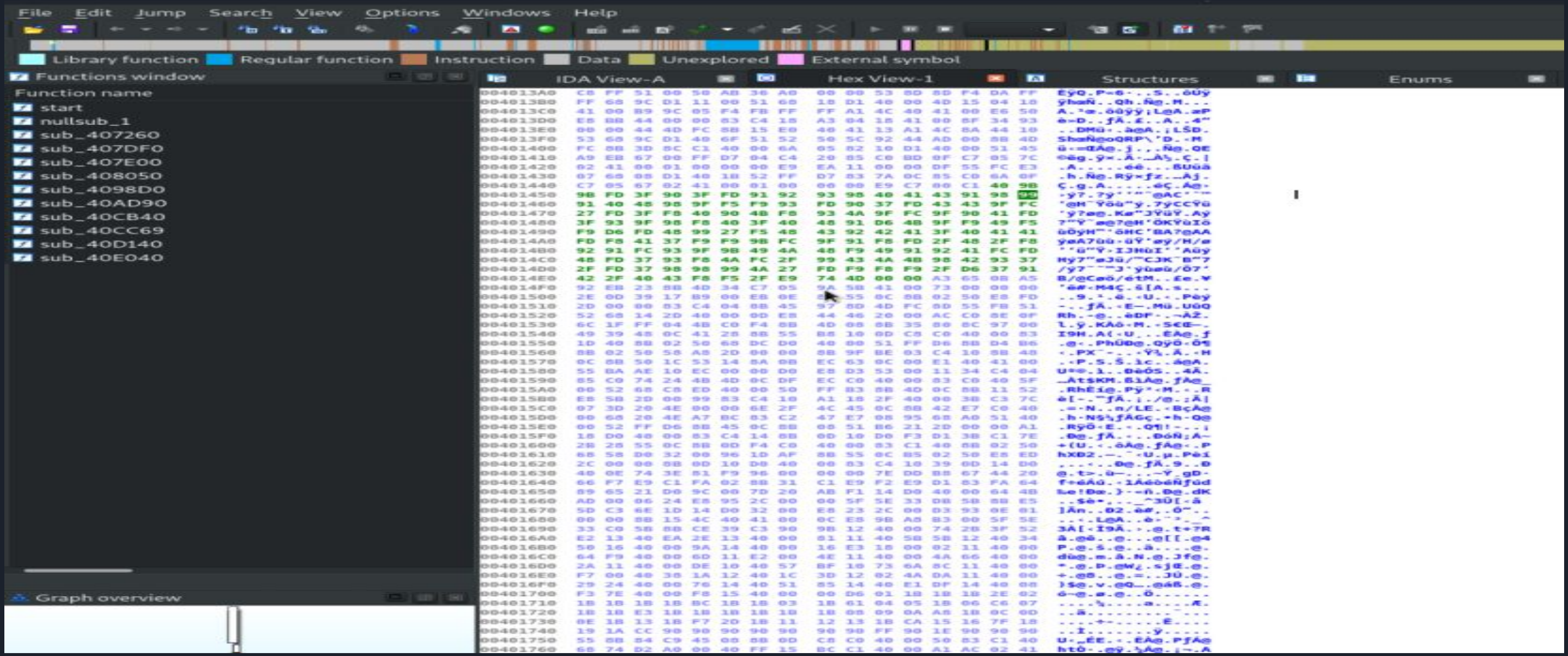
- That gave us 51 files to examined
- These file were again uploaded to windows 7

- We needed to figure out which of the files were matching Bitdefender’s signatures.
- So we scanned the folder containing all the split files with Bitdefender
- This time the antivirus it detected 4 files



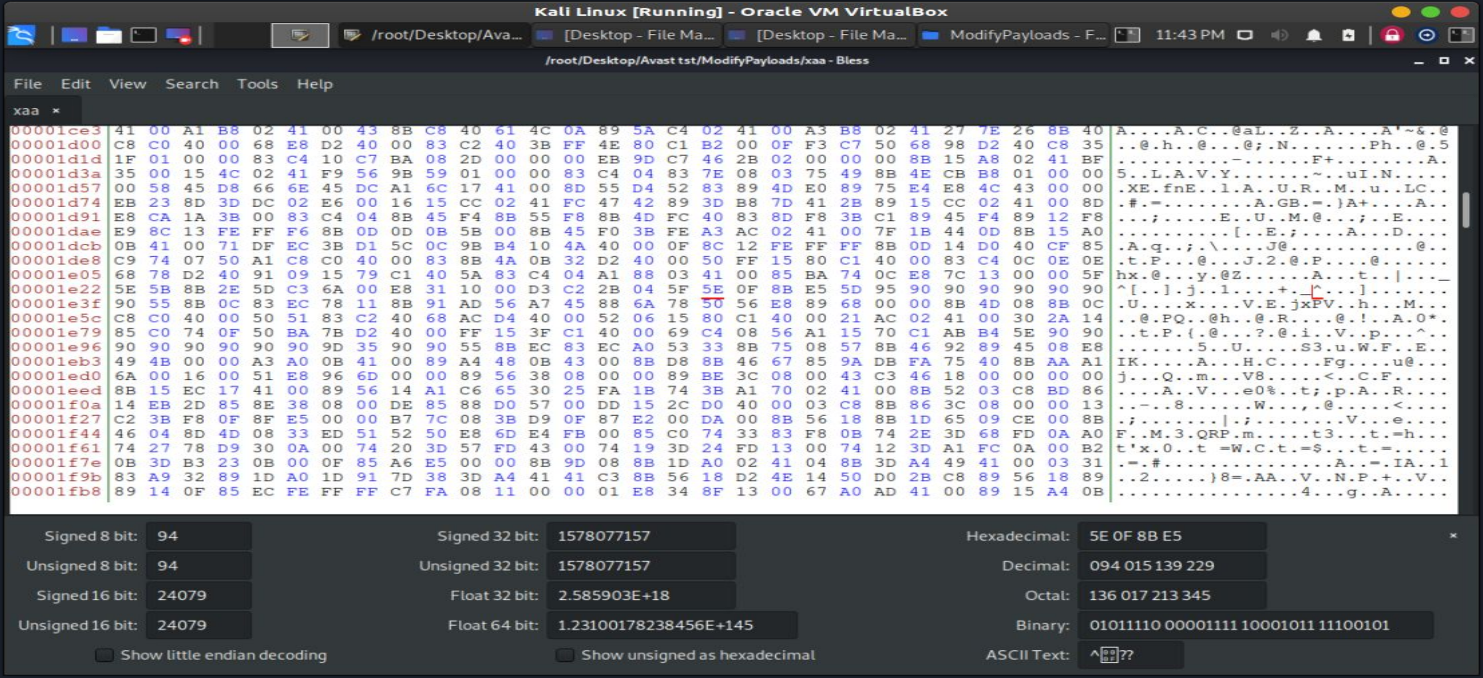
```
  GNU nano 4.9.2                    capturedSegments.txt                    Modified
xaa
xab
xac
xax
```

- With that information we knew then where to look
- Each one of these files was then uploaded to IDA and Bless Hex Editor
- File antivirus.exe on IDA



Function name
start
nullsub_1
sub_407260
sub_407DF0
sub_407E00
sub_408050
sub_4098D0
sub_40AD90
sub_40CB40
sub_40CC69
sub_40D140
sub_40E040

File: xaa
Bless Hex Editor



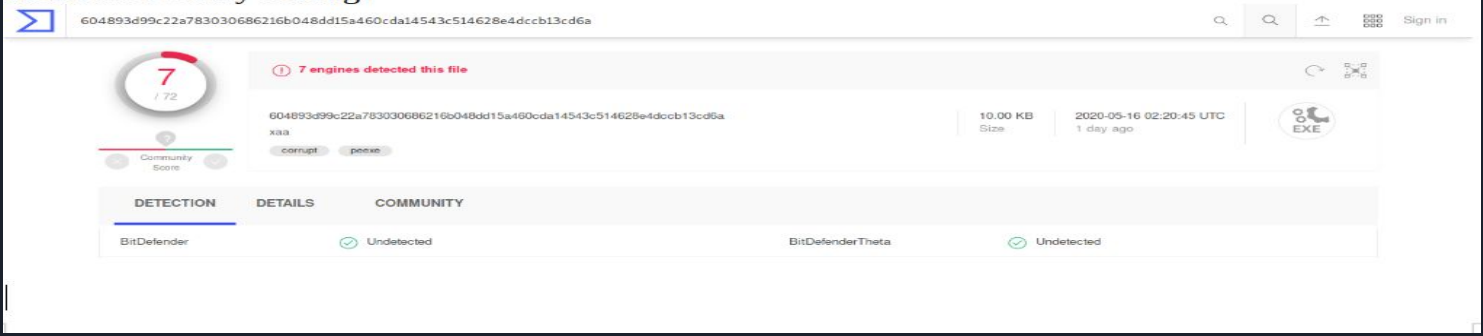
/root/Desktop/Avast tst/ModifyPayloads/xaa - Bless

```
00001ce3 41 00 A1 B8 02 41 00 43 8B C8 40 61 4C 0A 89 5A C4 02 41 00 A3 B8 02 41 27 7E 26 8B 40 A....A.C..@aL..Z..A....A'~&.@
00001d00 C8 C0 40 00 68 E8 D2 40 00 83 C2 40 3B FF 4E 80 C1 B2 00 0F F3 C7 50 68 98 D2 40 C8 35 ..@.h..@...@;.N.......Ph..@.5
00001d1d 1F 01 00 00 83 C4 10 C7 BA 08 2D 00 00 00 EB 9D C7 46 2B 02 00 00 00 8B 15 A8 02 41 BF ..........-......F+........A.
00001d3a 35 00 15 4C 02 41 F9 56 9B 59 01 00 00 83 C4 04 83 7E 08 03 75 49 8B 4E CB B8 01 00 00 5..L.A.V.Y.......~..uI.N.....
00001d57 00 58 45 D8 66 6E 45 DC A1 6C 17 41 00 8D 55 D4 52 83 89 4D E0 89 75 E4 E8 4C 43 00 00 .XE.fnE..l.A..U.R..M..u..LC..
00001d74 EB 23 8D 3D DC 02 E6 00 16 15 CC 02 41 FC 47 42 89 3D B8 7D 41 2B 89 15 CC 02 41 00 8D .#.=........A.GB.=.}A+....A..
00001d91 E8 CA 1A 3B 00 83 C4 04 8B 45 F4 8B 55 F8 8B 4D FC 40 83 8D F8 3B C1 89 45 F4 89 12 F8 ...;.....E..U..M.@...;..E....
00001dae E9 8C 13 FE FF F6 8B 0D 0D 0B 5B 00 8B 45 F0 3B FE A3 AC 02 41 00 7F 1B 44 0D 8B 15 A0 ..........[..E.;....A...D....
00001dcb 0B 41 00 71 DF EC 3B D1 5C 0C 9B B4 10 4A 40 00 0F 8C 12 FE FF FF 8B 0D 14 D0 40 CF 85 .A.q..;.\....J@...........@..
00001de8 C9 74 07 50 A1 C8 C0 40 00 83 8B 4A 0B 32 D2 40 00 50 FF 15 80 C1 40 00 83 C4 0C 0E 0E .t.P...@...J.2.@.P....@......
00001e05 68 78 D2 40 91 09 15 79 C1 40 5A 83 C4 04 A1 88 03 41 00 85 BA 74 0C E8 7C 13 00 00 5F hx.@...y.@Z......A...t..|..._
00001e22 5E 5B 8B 2E 5D C3 6A 00 E8 31 10 00 D3 C2 2B 04 5F 5E 0F 8B E5 5D 95 90 90 90 90 90 90 ^[..].j..1....+._^...].......
00001e3f 90 55 8B 0C 83 EC 78 11 8B 91 AD 56 A7 45 88 6A 78 50 56 E8 89 68 00 00 8B 4D 08 8B 0C .U....x....V.E.jxPV..h...M...
00001e5c C8 C0 40 00 50 51 83 C2 40 68 AC D4 40 00 52 06 15 80 C1 40 00 21 AC 02 41 00 30 2A 14 ..@.PQ..@h..@.R....@.!..A.0*.
00001e79 85 C0 74 0F 50 BA 7B D2 40 00 FF 15 3F C1 40 00 69 C4 08 56 A1 15 70 C1 AB B4 5E 90 90 ..t.P.{.@...?.@.i..V..p...^..
00001e96 90 90 90 90 90 90 9D 35 90 90 55 8B EC 83 EC A0 53 33 8B 75 08 57 8B 46 92 89 45 08 E8 .......5..U.....S3.u.W.F..E..
00001eb3 49 4B 00 00 A3 A0 0B 41 00 89 A4 48 0B 43 00 8B D8 8B 46 67 85 9A DB FA 75 40 8B AA A1 IK.....A...H.C....Fg....u@...
00001ed0 6A 00 16 00 51 E8 96 6D 00 00 89 56 38 08 00 00 89 BE 3C 08 00 43 C3 46 18 00 00 00 00 j...Q..m...V8.....<..C.F.....
00001eed 8B 15 EC 17 41 00 89 56 14 A1 C6 65 30 25 FA 1B 74 3B A1 70 02 41 00 8B 52 03 C8 BD 86 ....A..V...e0%..t;.p.A..R....
00001f0a 14 EB 2D 85 8E 38 08 00 DE 85 88 D0 57 00 DD 15 2C D0 40 00 03 C8 8B 86 3C 08 00 00 13 ..-..8......W...,.@.....<....
00001f27 C2 3B F8 0F 8F E5 00 00 B7 7C 08 3B D9 0F 87 E2 00 DA 00 8B 56 18 8B 1D 65 09 CE 00 8B .;.......|.;........V...e....
00001f44 46 04 8D 4D 08 33 ED 51 52 50 E8 6D E4 FB 00 85 C0 74 33 83 F8 0B 74 2E 3D 68 FD 0A A0 F..M.3.QRP.m.....t3...t.=h...
00001f61 74 27 78 D9 30 0A 00 74 20 3D 57 FD 43 00 74 19 3D 24 FD 13 00 74 12 3D A1 FC 0A 00 B2 t'x.0..t =W.C.t.=$...t.=.....
00001f7e 0B 3D B3 23 0B 00 0F 85 A6 E5 00 00 8B 9D 08 8B 1D A0 02 41 04 8B 3D A4 49 41 00 03 31 .=.#...............A..=.IA..1
00001f9b 83 A9 32 89 1D A0 1D 91 7D 38 3D A4 41 41 C3 8B 56 18 D2 4E 14 50 D0 2B C8 89 56 18 89 ..2.....}8=.AA..V..N.P.+..V..
00001fb8 89 14 0F 85 EC FE FF FF C7 FA 08 11 00 00 01 E8 34 8F 13 00 67 A0 AD 41 00 89 15 A4 0B ................4...g..A.....
```

```
  Signed 8 bit: 94        Signed 32 bit: 1578077157              Hexadecimal: 5E 0F 8B E5
Unsigned 8 bit: 94      Unsigned 32 bit: 1578077157                  Decimal: 094 015 139 229
 Signed 16 bit: 24079      Float 32 bit: 2.585903E+18                  Octal: 136 017 213 345
Unsigned 16 bit: 24079     Float 64 bit: 1.23100178238456E+145        Binary: 01011110 00001111 10001011 11100101
□ Show little endian decoding   □ Show unsigned as hexadecimal        ASCII Text: ^??
```

- The next steps were extremely challenging
- In the editor we got the address at the left, hex at the middle, ascii characters at the right.
- We took it very slow look for patterns that made sense to us
- We then replaced capital letter with lower case
- We then test for detection
- A lot of trial and error
- Until the antivirus gave us a negative detection

XAA after binary editing:



604893d99c22a783030686216b048dd15a460cda14543c514628e4dccb13cd6a

7
/ 72

7 engines detected this file

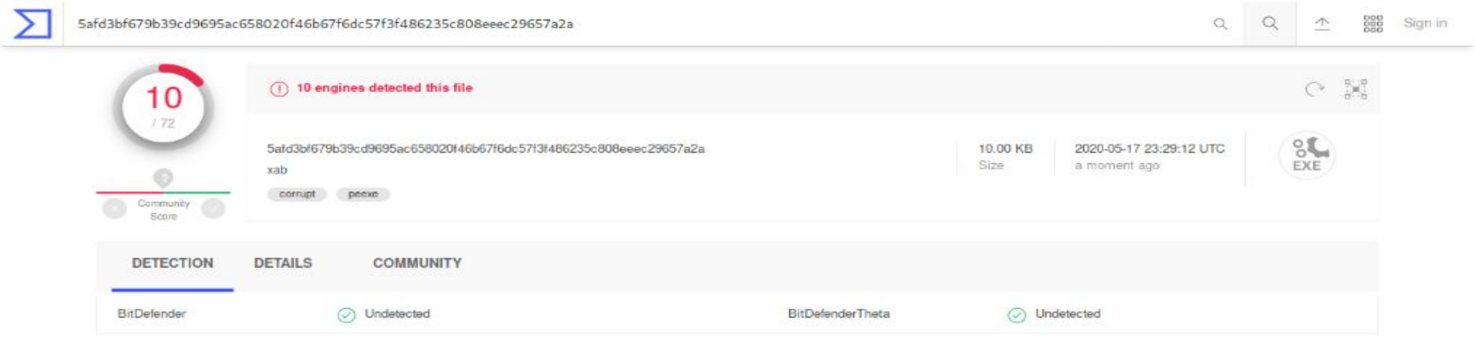
604893d99c22a783030686216b048dd15a460cda14543c514628e4dccb13cd6a
xaa
corrupt peexe

10.00 KB
Size

2020-05-16 02:20:45 UTC
1 day ago

BitDefender ✓ Undetected BitDefenderTheta ✓ Undetected

XAB after binary editing:



5afd3bf679b39cd9695ac658020f46b67f6dc57f3f486235c808eeec29657a2a

10
/ 72

10 engines detected this file

5afd3bf679b39cd9695ac658020f46b67f6dc57f3f486235c808eeec29657a2a
xab
corrupt peexe

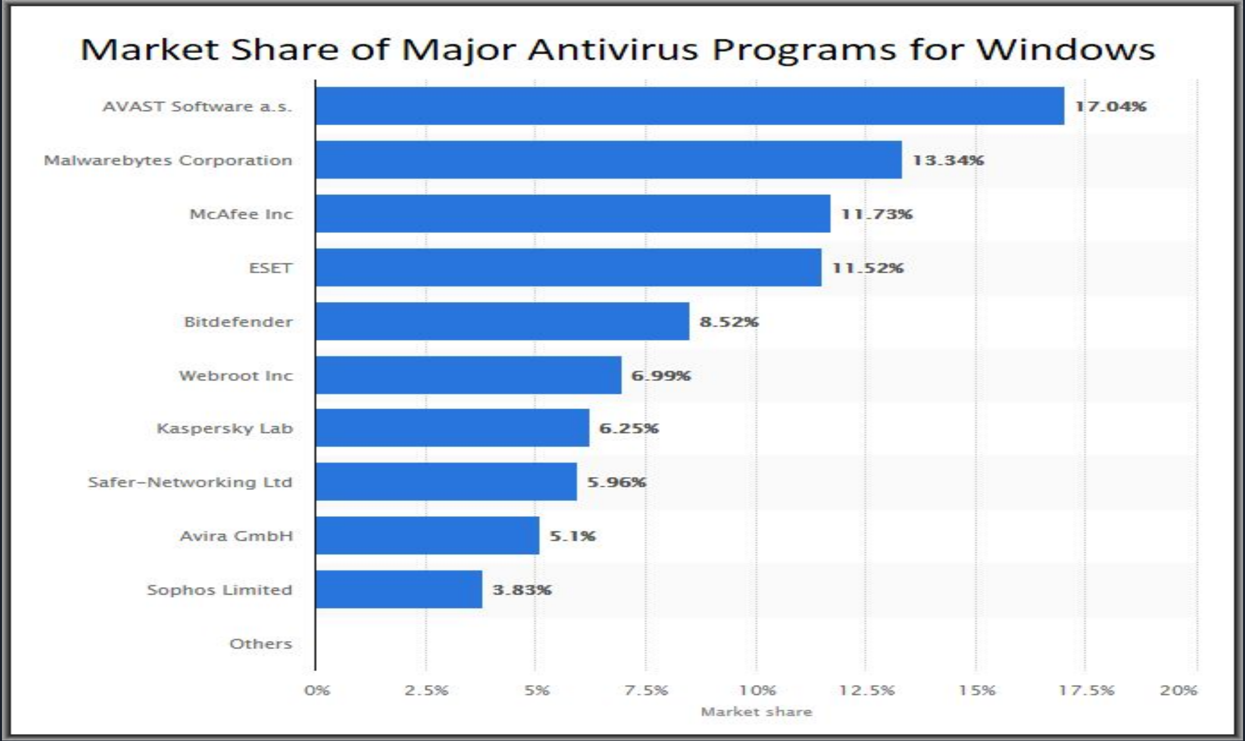
10.00 KB
Size

2020-05-17 23:29:12 UTC
a moment ago

BitDefender ✓ Undetected BitDefenderTheta ✓ Undetected

Defensive-

- Defending machine - Windows 10 64bit,
- Antivirus software from different vendors on virtual machine



➢ Fig 1
Since there are a number of softwares available. Using this statistics,, we chose four Anti-virus products on the basis of market share and secondly the ranking for the protection that provides.

Based on the above information and online sources we used the below anti-viruses on windows machine to test our payloads.

- Avast Free AntiVirus
- McAfee Total Protection
- Avira AntiVirus Pro
- Norton Security
- Windows Defender

For comparative analysis, we have applied a strategy to score the tools on the basis of their functionality. Lets say payload generated through Hyperion 2.2, bypasses the Avast free AntiVirus scores 1 otherwise 0.

We created few samples using the evasion tool from defensive part and tested them against all selected antiviruses.

For the evaluation test of Veil Evasion Framework 3.0 , we selected two custom payloads and two predefined payloads, the two custom payloads were a reverse_tcp meterpreter and a reverse_tcp shell both encoded with Shikata.

veil_cp1 - windows reverse_tcp meterpreter with Shikata encoding

veil_cp2 - windows reverse_tcp shell with Shikata encoding

veil_p1 - windows reverse_tcp (predefined payload1)

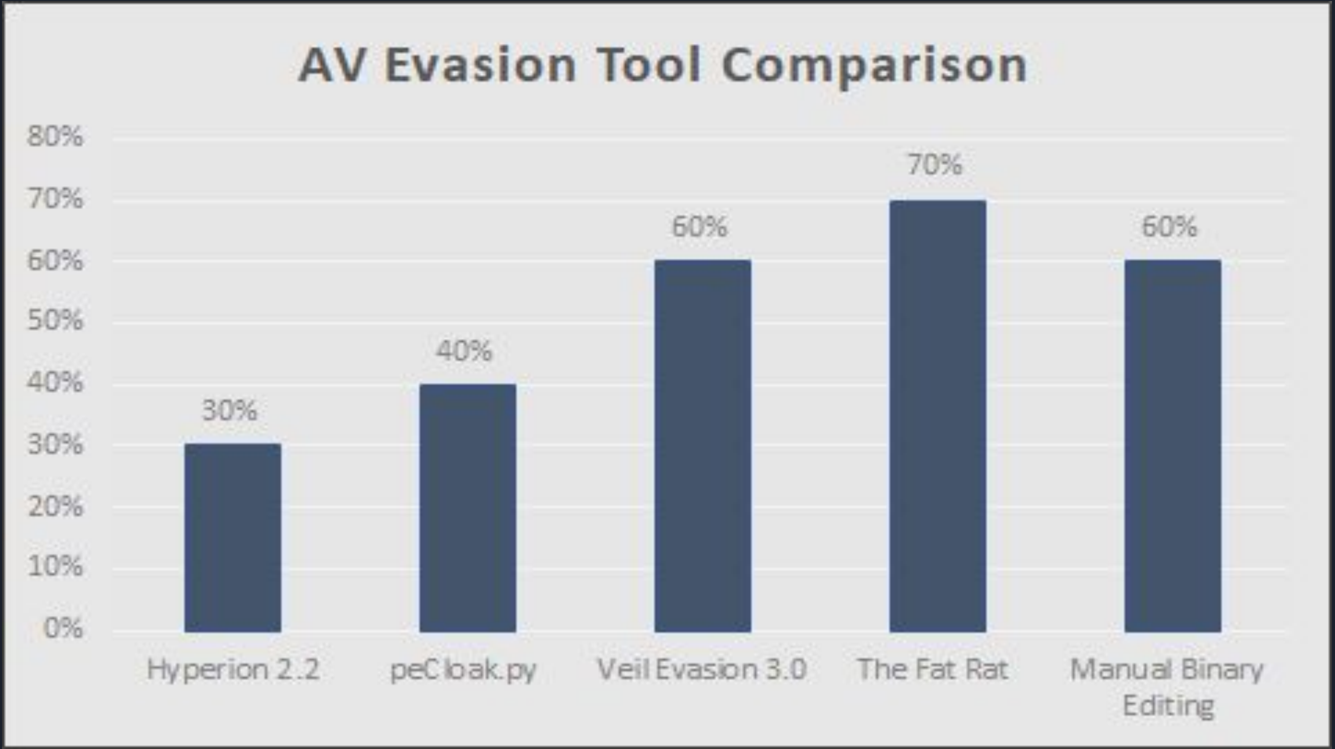
veil_p2 - windows reverse_tcp (predefined payload2)

Veil EV sample	veil_cp1	veil_cp2	veil_p1	Veil_p2
Avast Free	1	1	0	0
McAfee	1	1	0	0
Avira Antivirus	1	1	0	0
Norton Security	0	0	0	0
Windows Defender	1	1	0	0

Table 1 - As shown in the table, the best effort for the veil evasion is with custom payloads which bypassed 4 out of 5 Anti-viruses.

Outcomes

The graph is constructed using the evasion ratio
(AVs evaded/total number of selected evasion tools)



Conclusion

The evaluation revealed that the AV evasion task is not impossible mission. Actually is easy enough to bypass enough of the most known AV software products of the market. The are enough evasion tools that efficiently are completing their task.

As concern the technical aspect of this, most of the tools employing code caving and encryption techniques, in order to bypass signature based detection. In addition, the use of iterations is the easiest way to evade sandboxing and heuristic detection.

AVRanking with respect to the tools we used

1. Norton Security
2. Avast Free software
3. McAfee Total Protection
4. Avira Antivirus Pro

REFERENCES

1. P. Casey, M. Topor, E. Hennessy, S. Alrabaee, M. Aloqaily and A. Boukerche, "Applied Comparative Evaluation of the Metasploit Evasion Module," *2019 IEEE Symposium on Computers and Communications (ISCC)*, Barcelona, Spain, 2019, pp. 1-6, doi: 10.1109/ISCC47284.2019.8969663.

2. F. A. Garba, K. I. Kunya, S. A. Ibrahim, A. B. Isa, K. M. Muhammad and N. N. Wali, "Evaluating the State of the Art Antivirus Evasion Tools on Windows and Android Platform," *2019 2nd International Conference of the IEEE Nigeria Computer Chapter (NigeriaComputConf)*, Zaria, Nigeria, 2019, pp. 1-4, doi: 10.1109/NigeriaComputConf45974.2019.8949637.