



HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
Posts and Telecommunications Institute of Technology

PHÂN TÍCH MÃ ĐỘC

KHOA AN TOÀN THÔNG TIN
TS. ĐÌNH TRƯỜNG DUY



HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
Posts and Telecommunications Institute of Technology

PHÂN TÍCH MÃ ĐỘC DỰA TRÊN KỸ THUẬT PHÂN TÍCH TĨNH

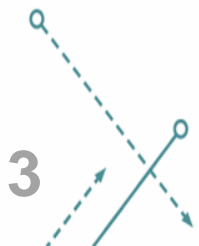
CHƯƠNG 2

KHOA AN TOÀN THÔNG TIN
TS. ĐÌNH TRƯỜNG DUY

Giới thiệu

Phân tích mã độc dựa trên kỹ thuật phân tích tĩnh

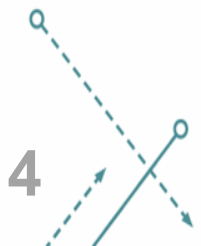
1. Tổng quan về phân tích tĩnh
2. Một số công cụ phân tích tĩnh phổ biến
3. Quy trình phân tích tĩnh
4. Đánh giá về phân tích tĩnh
5. Thực hành phân tích tĩnh sử dụng công cụ
 - 5.1 chuẩn bị môi trường thực nghiệm
 - 5.2 chuẩn bị mã độc, chạy



1. Tổng quan về phân tích tĩnh

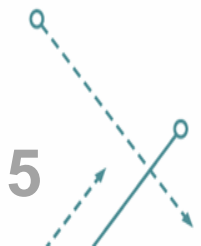
1.1. Một số khái niệm cơ bản

1.2. Vai trò phân tích tĩnh



1.1. Một số khái niệm cơ bản

- Khi bắt đầu phân tích một file (tệp), thì tệp này được gọi là sample (mẫu). Việc phân tích một tệp để xác nhận xem tệp này là malware (mã độc) hay là benign (an toàn)



1.1. Một số khái niệm cơ bản

- Phân tích tĩnh là kỹ thuật phân tích tệp nghi ngờ (mẫu) mà không thực thi nó.
- Là phương pháp phân tích ban đầu liên quan đến việc trích xuất thông tin hữu ích từ tệp nhị phân nghi ngờ để đưa ra quyết định có căn cứ về cách phân loại hoặc phân tích mã độc và chỉ ra hướng phân tích tiếp theo.



1.1. Một số khái niệm cơ bản

Phân tích tĩnh được chia làm 2 loại:

- Phân tích tĩnh cơ bản. Phân tích tĩnh cơ bản bao gồm các kỹ thuật như:
 - Giải mã: Giải mã mã nhị phân thành mã nguồn lập trình.
 - Phân tích chuỗi: Tìm kiếm các chuỗi, hằng số trong mã.
 - Phân tích cấu trúc: Phân tích cấu trúc gói, lớp, hàm trong mã.
- Phân tích tĩnh nâng cao. Phân tích tĩnh nâng cao bao gồm các kỹ thuật phức tạp hơn như:
 - Phân tích luồng điều khiển: Tìm hiểu luồng điều khiển của chương trình.
 - Phân tích dòng dữ liệu: Theo dõi dòng dữ liệu qua chương trình.
 - Phân tích hàm: Phân tích các hàm trong chương trình để hiểu chức năng.



1.2 Vai trò của phân tích tĩnh

- Cung cấp thông tin ban đầu mà không cần thực thi tệp: Phân tích tĩnh có thể được thực hiện mà không cần thực thi tệp. Điều này cho phép thu thập thông tin ban đầu về tệp mà không kích hoạt mã độc tiềm ẩn.
- Xác định hành vi tiềm ẩn: Phân tích tĩnh có thể tiết lộ những gợi ý về hành vi tiềm ẩn của tệp như kết nối mạng, truy cập hệ thống tập tin, v.v. Điều này có thể giúp xác định liệu tệp có đáng nghi ngờ hay không.
- Xác định cách thức hoạt động: Phân tích tĩnh có thể tiết lộ cách thức hoạt động của tệp thông qua việc khám phá cấu trúc, luồng điều khiển và các hàm.
- Hỗ trợ phân tích động sau đó: Thông tin thu thập được từ phân tích tĩnh có thể hỗ trợ phân tích động sau đó khi thực thi tệp trong môi trường cô lập.
- Phân loại tệp: Phân tích tĩnh có thể cung cấp đủ thông tin để phân loại nhanh tệp là benign hoặc đáng ngờ.

2.2 Một số công cụ phân tích tĩnh

IDA

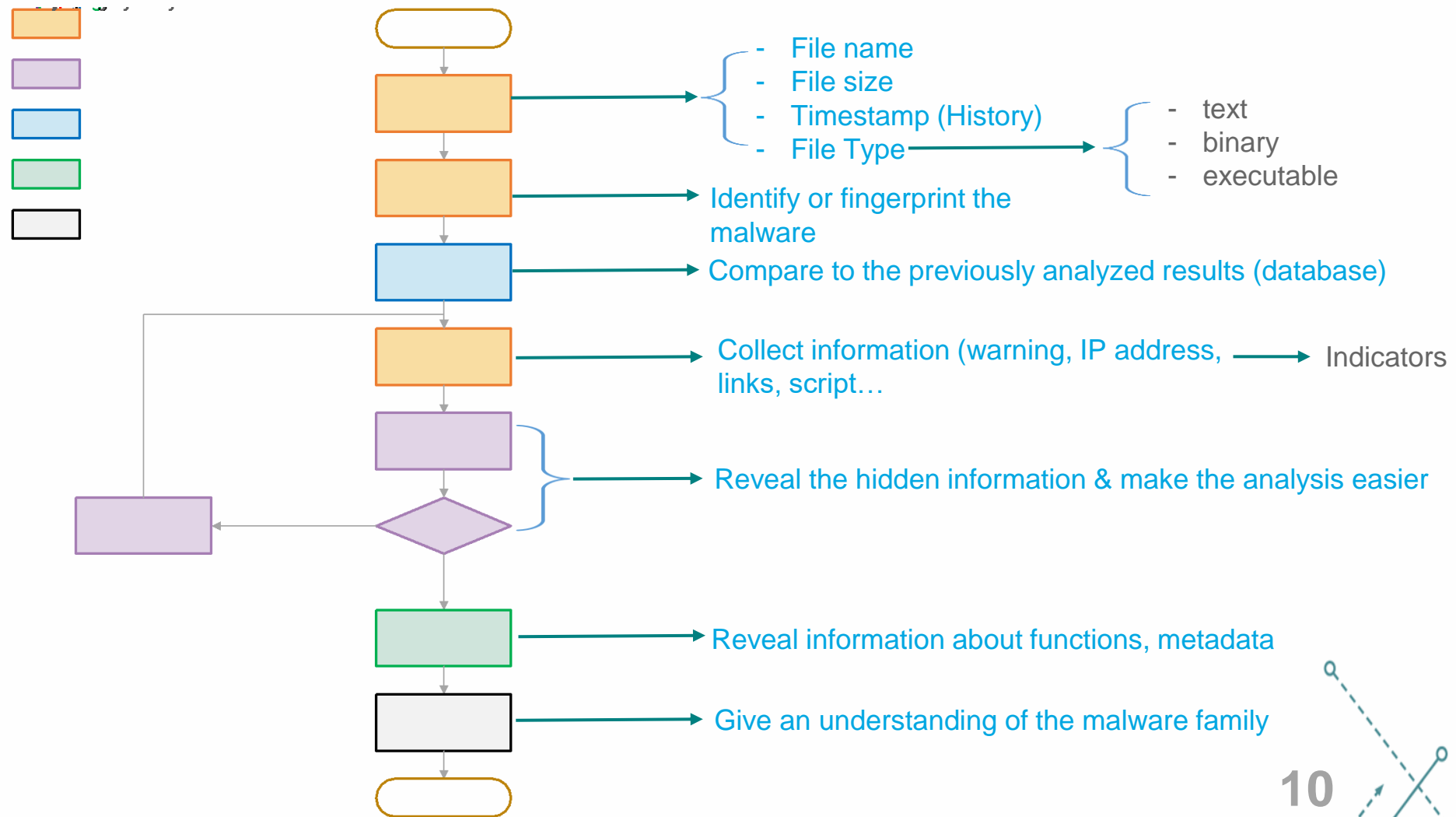


“FLOSS”

NTCore



3. Quy trình phân tích tĩnh



3.1 Xác định loại của tệp

- Việc xác định loại tệp sẽ tiết lộ thông tin quan trọng như:
 - Hệ điều hành mục tiêu. Ví dụ .exe thường dành cho Windows.
 - Kiến trúc nền tảng 32 bit hay 64 bit. Phần mềm độc hại phải tương thích với kiến trúc đó.
- Phần mở rộng tệp (File extension)
 - là một phần của tên tệp được gắn liền với tên tệp để chỉ định định dạng hoặc loại tệp, vd: .exe, .dll, .sys, docx, xlsx...
 - Phần mở rộng tệp giúp hệ điều hành và các ứng dụng phần mềm nhận biết loại tệp và sử dụng chương trình mở tương ứng để xem hoặc xử lý tệp.
 - Tuy nhiên nó có thể bị thay đổi để lừa người dùng thực thi nó.

3.1 Xác định loại của tệp

- **Chữ ký tệp (File Signature):**

- có thể được sử dụng để xác định loại tệp, thay cho phần mở rộng tệp.
 - Chữ ký tệp là trình tự byte duy nhất được viết vào phần đầu tệp.
 - Các tệp khác nhau có chữ ký khác nhau có thể được sử dụng để xác định loại tệp.
 - VD: Các tệp thực thi Windows, còn gọi là tệp PE (như các tệp kết thúc bằng .exe, .dll, .com, .drv, .sys ...) có chữ ký tệp là MZ hoặc ký tự thập lục phân 4D 5A trong hai byte đầu tiên của tệp.
- <https://www.filesignatures.net/>

<u>Extension</u>	<u>Signature</u>	<u>Description</u>
☆ <u>EXE</u>	<u>4D 5A</u> ASCII MZ	Windows DOS executable file Size: 2 Bytes Offset: 0 Bytes
<u>Extension</u>	<u>Signature</u>	<u>Description</u>
☆ <u>DMG</u>	<u>78</u> ASCII x	MacOS X image file Size: 1 Bytes Offset: 0 Bytes

3.1 Xác định loại của tệp

Popular Executable File Formats and Their Magic Bytes

OS	File Type/Format	Magic Bytes HEX	Magic Bytes ASCII
Windows	Windows Executable	4D 5A	MZ
Linux	Linux Executable	7F 45 4C 46	.ELF
Mach-O	Mach-O Executable	FE ED FA CE

Table 3-4. *Popular Nonexecutable File Formats and Their Magic Bytes*

File Format/Type	File Extension	Magic Bytes HEX	Magic Bytes ASCII
PDF Document	.pdf	25 50 44 46	%PDF
Adobe Flash	.swf	46 57 53	FWS
Flash Video	.flv	46 4C 56	FLV
Video AVI files	.avi	52 49 46 46	RIFF
Zip compressed files	.zip	50 4B	PK
Rar compressed files	.rar	52 61 72 21	rar!
Microsoft document	.doc	D0 CF	

3.1 Xác định loại của tệp

- Dữ liệu chứa các tệp từ kẻ tấn công hoặc chứa các tệp khác được nhúng vào trong tệp cha bên ngoài.

The screenshot shows a Wireshark interface with a packet list and packet details pane. The packet list shows three TCP segments. The packet details pane shows the raw data of a packet, with the text "ZIPPED FILE RECOGNIAED BY MAGIC BYTES PK" overlaid. An arrow points to the "PK" in the text, which is circled in the raw data pane.

No.	Time	Source	Destination	Protocol	Length	Info
18				TCP	1334	[TCP segment of a reassemb
19				TCP	54	49174 → 80 [ACK] Seq=323 A
20				TCP	1514	[TCP segment of a reassemb

Offset	Raw Data	Decoded Data
0180	73 3a 20 30 0d 0a 43 61 63 68 65 2d 43 6f 6e 74	s: 0..Ca che-Cont
0190	72 6f 6c 3a 20 6d 75 73 74 2d 72 65 76 61 6c 69	rol: mus t-revali
01a0	64 61 74 65 0d 0a 50 72 61 67 6d 61 3a 20 70 75	date..Pr agma: pu
01b0	62 6c 69 63 0d 0a 0d 0a 50 4b 03 04 14 00 00 00	blic... PK... ..
01c0	08 00 4f a6 73 4f 6e bc 72 41 17 bd 01 00 86 94	..0.sOn. tA.....

Sử dụng các "magic bytes" có thể nhanh chóng xác định sự có mặt của các tệp trong dữ liệu khác như là dữ liệu trong gói tin

3.1 Xác định loại của tệp

- Các công cụ sử dụng để xác định loại của tệp:
 - Sử dụng phương pháp thủ công để xác định loại tệp là tìm kiếm chữ ký tệp bằng cách mở nó trong một trình chỉnh sửa hex (vd: HxD hex editor)

The screenshot shows the HxD hex editor interface with the file 'log.exe' open. The hex view displays the first 70 bytes of the file. The first two bytes, '4D 5A', are highlighted with a red box. The ASCII view on the right shows the 'MZ' signature, which is also highlighted with a red box. The ASCII view also displays the text 'is program cannot be run in DOS mode'.

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	E8	00	00	00
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00

3.1 Xác định loại của tệp

- Sử dụng phương pháp xác định bằng công cụ.
 - Trên hệ điều hành Linux có thể sử dụng công cụ *file utility*.

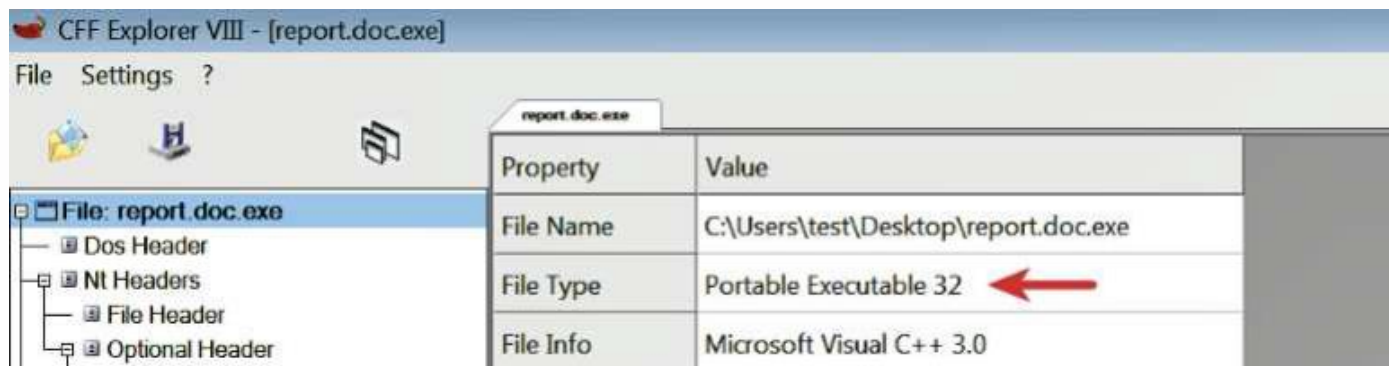
```
$ file mini
```

```
mini: PE32 executable (GUI) Intel 80386, for MS Windows
```

```
$ file notepad.exe
```

```
notepad.exe: PE32+ executable (GUI) x86-64, for MS Windows
```

- Trên hệ điều hành Windows có thể sử dụng công cụ *CFF Explorer*, là 1 phần của công cụ *Explorer Suite*



3.1 Xác định loại của tệp

- Sử dụng phương pháp xác định bằng ngôn ngữ Python.
Sử dụng thư viện *python-magic*

```
$ python
```

```
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
```

```
>>> import magic
```

```
>>> m = magic.open(magic.MAGIC_NONE)
```

```
>>> m.load()
```

```
>>> ftype = m.file(r'log.exe')
```

```
>>> print ftype
```

```
PE32 executable (GUI) Intel 80386, for MS Windows
```



3.2 Phân tích mã Hash

- Quá trình tạo các giá trị băm cho các tệp nghi ngờ dựa trên nội dung của chúng, cũng giống như tạo vân tay (Fingerprinting) cho mã độc. Các thuật toán băm thường được sử dụng như MD5, SHA1 hoặc SHA256. Sử dụng các giá trị băm cho phân tích mã độc mang lại các lợi thế sau:
 - Định danh duy nhất cho mã độc trong quá trình phân tích.
 - Xác định một mẫu malware dựa trên tên tệp là không hiệu quả vì cùng một mẫu malware có thể sử dụng các tên tệp khác nhau, nhưng giá trị băm mã học được tính dựa trên nội dung tệp sẽ giữ nguyên.



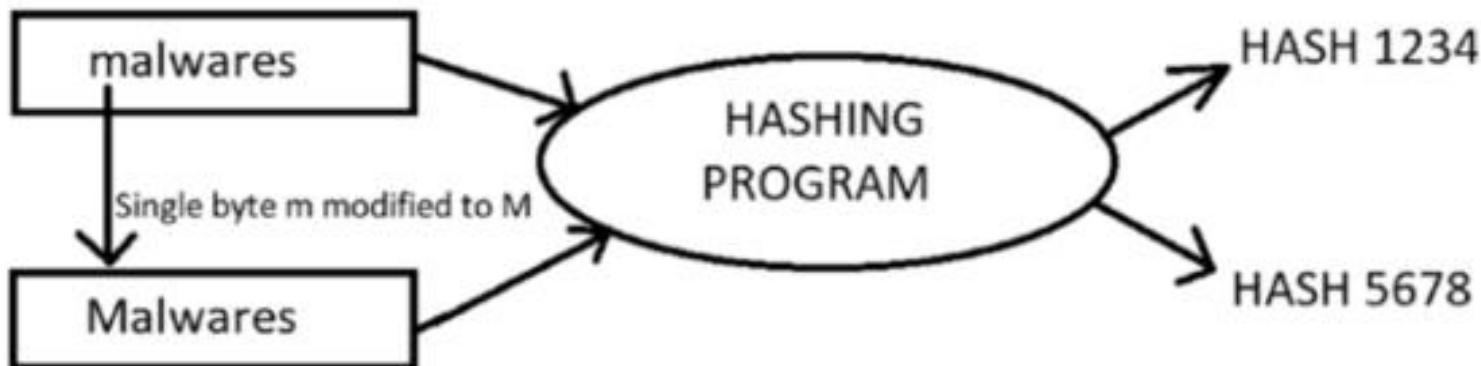
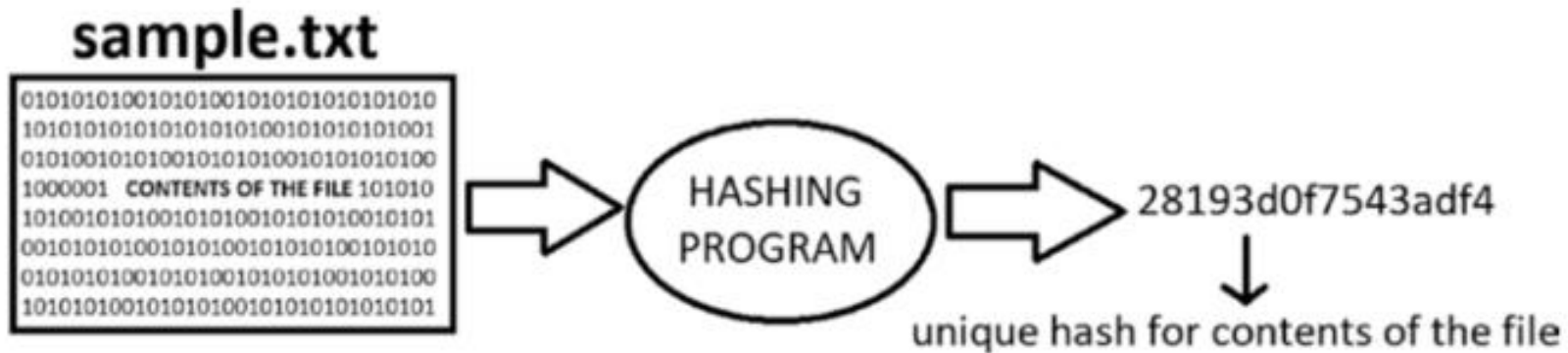
3.2 Phân tích mã Hash

- Quyết định liệu phân tích cần được thực hiện trên một mẫu duy nhất hay nhiều mẫu.
 - Trong quá trình phân tích động, khi malware được thực thi, nó có thể sao chép chính nó đến một vị trí khác hoặc tạo ra một mẫu malware khác. Có giá trị băm mã học của mẫu sẽ giúp xác định xem mẫu mới được sao chép/hành động có giống với mẫu gốc hay không.
- Được sử dụng để chia sẻ cho các nhà nghiên cứu bảo mật khác khi cần xác định mẫu.
- Xác định nhanh xem mã độc đã được phát hiện trước đó bằng cách tìm kiếm trực tuyến hoặc tìm kiếm trong cơ sở dữ liệu của các dịch vụ quét mã độc như VirusTotal.



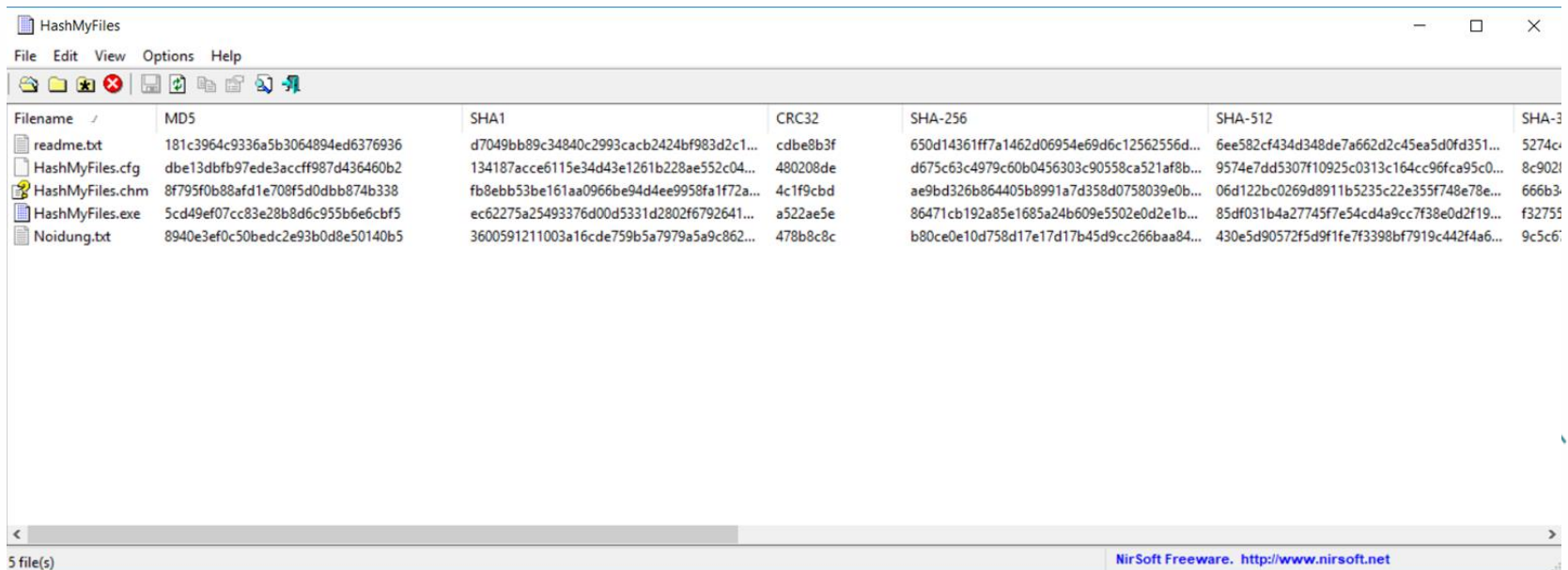


3.2 Phân tích mã Hash



3.2 Phân tích mã Hash

- Các công cụ có thể sử dụng:
 - Linux: Md5sum, Sha256sum, Sha1sum ...
 - Windows: HashMyFiles, FsumFrontEnd, Jacksum ...

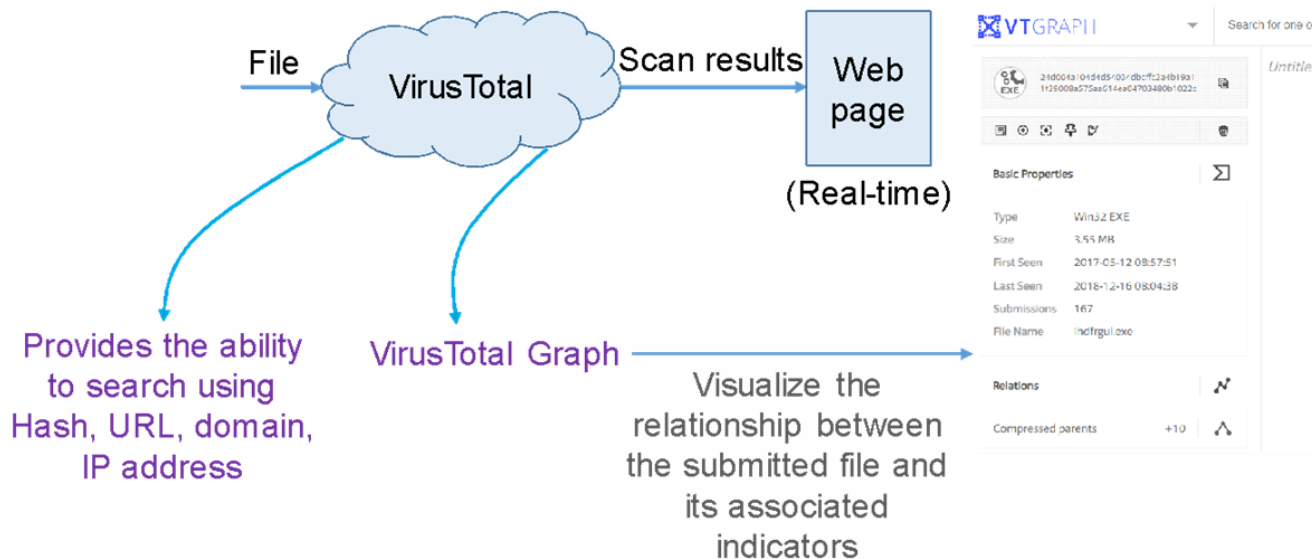


Filename	MD5	SHA1	CRC32	SHA-256	SHA-512	SHA-3
readme.txt	181c3964c9336a5b3064894ed6376936	d7049bb89c34840c2993cacb2424bf983d2c1...	cdbe8b3f	650d14361ff7a1462d06954e69d6c12562556d...	6ee582cf434d348de7a662d2c45ea5d0fd351...	5274c...
HashMyFiles.cfg	dbe13dbfb97ede3acff987d436460b2	134187acce6115e34d43e1261b28ae552c04...	480208de	d675c63c4979c60b0456303c90558ca521af8b...	9574e7dd5307f10925c0313c164cc96ca95c0...	8c902f...
HashMyFiles.chm	8f795f0b88afd1e708f5d0dbb874b338	fb8ebb53be161aa0966be94d4ee9958fa1f72a...	4c1f9cbd	ae9bd326b864405b8991a7d358d0758039e0b...	06d122bc0269d8911b5235c22e355f748e78e...	666b3...
HashMyFiles.exe	5cd49ef07cc83e28b8d6c955b6e6cbf5	ec62275a25493376d00d5331d2802f6792641...	a522ae5e	86471cb192a85e1685a24b609e5502e0d2e1b...	85df031b4a27745f7e54cd4a9cc7f38e0d2f19...	f32755...
Noidung.txt	8940e3ef0c50bedc2e93b0d8e50140b5	3600591211003a16cde759b5a7979a5a9c862...	478b8c8c	b80ce0e10d758d17e17d17b45d9cc266baa84...	430e5d90572f5d9f1fe7f3398bf7919c442f4a6...	9c5c6...

5 file(s) NirSoft Freeware. <http://www.nirsoft.net>

3.3 Phân tích mẫu đang có sử dụng công cụ kết hợp nhiều Antivirus (AV)

- Sử dụng các công cụ tích hợp nhiều AV giúp việc so sánh chữ ký của các mẫu nghi ngờ với cơ sở dữ liệu của các AV giúp cho nhanh chóng phát hiện mã độc cũng như làm giàu thêm cho các cơ sở dữ liệu này.



3.3 Phân tích mẫu đang có sử dụng công cụ kết hợp nhiều Antivirus (AV)



62 engines detected this file

SHA-256 24d004a104d4d54034dbcf2a4b19a11f39008a575aa614ea04703480b1022c
File name lhdfgule.exe
File size 3.55 MB
Last analysis 2019-01-16 04:07:51 UTC
Community score -1570

62 / 71

Detection

Details

Relations

Behavior

Community 10

Acronis	suspicious	Ad-Aware	Trojan.Ransom.WannaCryptor.H
AhnLab-V3	Trojan/Win32.WannaCryptor.R200572	ALYac	Trojan.Ransom.WannaCryptor
Antiy-AVL	Trojan[Ransom]/Win32.Scatter	Arcabit	Trojan.Ransom.WannaCryptor.H
Avast	Sf:WNCryLdr-A [Trj]	AVG	Sf:WNCryLdr-A [Trj]
Avira	TR/Ransom.IZ	Baidu	Win32.Worm.Rbot.a
BitDefender	Trojan.Ransom.WannaCryptor.H	Bkav	W32.WannaCryLdr.Trojan
CAT-QuickHeal	Trojan.Mauvaise.SL1	ClamAV	Win.Ransomware.WannaCry-6313787-0
Comodo	Trojan.Win32.WannaCryjet@714um4	CrowdStrike Falcon	malicious_confidence_100% (W)
Cybereason	malicious.7c37d2	Cylance	Unsafe
Cyren	W32/Trojan.ZTSA-8671	DrWeb	Trojan.Encoder.11432
eGambit	Trojan.Generic	Emsisoft	Trojan-Ransom.WannaCryptor (A)
Endgame	malicious (high confidence)	eScan	Trojan.Ransom.WannaCryptor.H



62 engines detected this file

SHA-256 24d004a104d4d54034dbcf2a4b19a11f39008a575aa614ea04703480b1022c
File name lhdfgule.exe
File size 3.55 MB
Last analysis 2019-01-16 04:07:51 UTC
Community score -1570

Detection

Details

Relations

Behavior

Community 10

Basic Properties

MD5 db349b97c37d22f5ea1d1841e3c89eb4
SHA-1 e889544af85ffaf8b0dda705105dee7c97fe26
Authenthash 1646cad4fe91337460de0d4c2c5451095023e74bdab331642aaca12647b2f46
Imphash 9ecee117164e0b870a53dd187cdd7174
File Type Win32 EXE
Magic PE32 executable for MS Windows (GUI) Intel 80386 32-bit
SSDeep 98304wDgPbH21aRxcSUDK365AEhvxW9P593R8yAVp2g3RwDqPe1Ccxk3ZAEUadzR8y4gB
TRID Win32 Executable MS Visual C++ (generic) (41%)
Win64 Executable (generic) (36.3%)
Win32 Dynamic Link Library (generic) (8.6%)
Win32 Executable (generic) (5.9%)
OS/2 Executable (generic) (2.6%)
File Size 3.55 MB

Tags

peexe cve-2017-0147 exploit

History

Creation Time 2010-11-20 09:03:08
First Seen In The Wild 2010-11-20 02:03:08
First Submission 2017-05-12 08:57:51
Last Submission 2018-12-16 08:04:38
Last Analysis 2019-01-16 04:07:51

File name

File size

File Identifier

Hash

Features

History

Signature info

Packer

PE info

23

23

<https://virusscan.jotti.org/>
<https://metadefender.opswat.com/#/>
<http://www.virscan.org/language/en/>

3.4 Phân tích strings

- Strings là các chuỗi ký tự ASCII và Unicode có thể hiển thị (Unicode-printable), chúng được nhúng trong một tệp.
- Phân tích chuỗi có thể cung cấp gợi ý về chức năng của chương trình và các chỉ số liên quan đến một tệp nhị phân (binary file) đáng nghi.
- Các chuỗi được trích xuất từ tệp nhị phân có chứa các tham chiếu đến tên tệp, URL, tên miền, địa chỉ IP, lệnh tấn công, registry,... thì khả năng cao là có dính mã độc.



3.4 Phân tích strings

- Phân tích string sử dụng công cụ:
 - **Trên linux:** strings utility (lệnh `strings`) trích xuất các chuỗi ASCII có ít nhất bốn ký tự. Với tùy chọn `-a`, có thể trích xuất các chuỗi từ toàn bộ tệp.
 - Các mẫu mã độc cũng sử dụng các chuỗi Unicode (2 byte cho mỗi ký tự). Để trích xuất chuỗi Unicode bằng lệnh "`strings`", sử dụng tùy chọn `-el`, vd: `strings -a -el multi.exe`

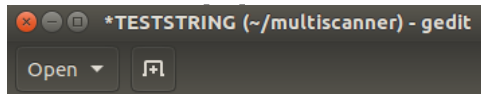
```
$ strings -a log.exe
!This program cannot be run in DOS mode.
Rich
.text
`.rdata
@.data
L$"%
h4z@
128.91.34.188
%04d-%02d-%02d %02d:%02d:%02d %s
```

Phát hiện ra địa chỉ IP
khi phân tích file log.exe



3.4 Phân tích strings

- Phân tích string sử dụng công cụ:
 - Trên Windows: PEstudio, PPEE, strings utility,



```
GetModuleFileNameA
ExpandEnvironmentStringsW
EnumDateFormatsA
lstrlenA
GetTickCount
GetCurrentProcess
OutputDebugStringA
HeapAlloc
GetProcessHeap
SetLastError
VirtualLock
GetModuleFileNameW
GetTempPathW
GetModuleHandleA
GetMailslotInfo
CreateEventA
CancelIoEx
WaitForSingleObject
lstrcpyA
lstrlenW
GetCommandLineW
GlobalFree
GetOverlappedResult
GlobalAlloc
CreateEventW
KERNEL32.dll
SetDlgItemTextA
GetWindowRect
GetDesktopWindow
GetParent
DialogBoxParamA
```

pestudio 8.54 - Malware Initial Assessment - www.winitor.com

	type	size	loc...	blacklisted (61)	item (372)
indicators (3/9)	unicode	7	-	x	AppData
virustotal (n/a)	unicode	45	-	x	Software\Microsoft\Windows\CurrentVersion\Run
dos-stub (64 bytes)	unicode	38	-	x	netsh firewall delete allowedprogram *
file-header (20 bytes)	unicode	4	-	x	.exe
optional-header (224 bytes)	unicode	30	-	x	cmd.exe /c ping 0 -n 2 & del *
directories (5/15)	unicode	35	-	x	netsh firewall add allowedprogram *
sections (3)	unicode	13	-	x	Execute ERROR
libraries (1)	unicode	14	-	x	Download ERROR
imports (1)	unicode	5	-	x	start
exports (n/a)	unicode	12	-	x	Update ERROR
exceptions (n/a)	unicode	7	-	x	[ENTER]
tls-callbacks (n/a)	ascii	40	-	-	!This program cannot be run in DOS mode.
resources (1)	ascii	5	-	-	.text
strings (61/372)	ascii	7	-	-	@.reloc
debug (n/a)	ascii	4	-	-	3)r
manifest (invoker)					

3.5 Obfuscation and packer checking

- Obfuscation là quá trình biến đổi tệp nhị phân và văn bản để làm cho chúng không đọc được hoặc khó hiểu/ khó phát hiện/ khó phân tích được.
- Mục tiêu của việc obfuscate là che giấu mã độc khỏi các phần mềm antivirus và các chương trình phần mềm bảo mật khác.



3.5 Obfuscation and packer checking

- Obfuscated strings: là các chuỗi đã bị “che giấu”, bị xáo trộn làm cho các công cụ phân tích chuỗi như strings utility không thể trích xuất ra các chuỗi.
 - Các kỹ thuật Obfuscated strings: XOR Encryption, Base64 Encoding, Unicode Escaping, String Concatenation, Dynamic String Decryption.
- sử dụng FireEye Labs Obfuscated String Solver (FLOSS). FLOSS có thể phân tích chuỗi thông thường cũng như các chuỗi bị obfuscated.

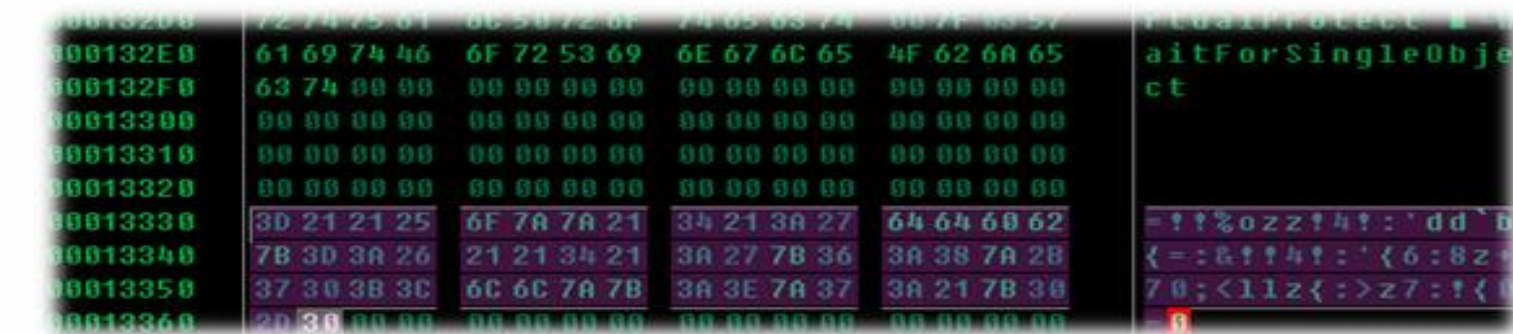


3.5 Obfuscation and packer checking

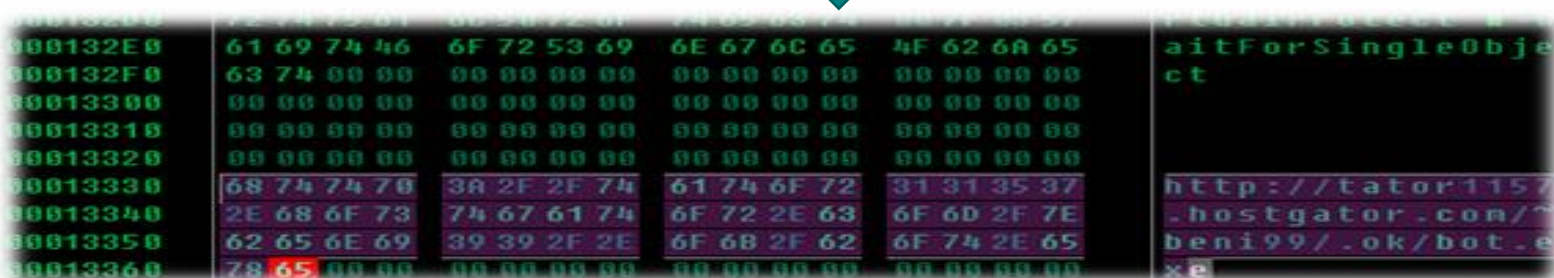
- Exclusive OR (XOR)

✓ Những chuỗi nên tìm kiếm là “http” và “this program”

Input		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	0




Sử dụng phép XOR với 0x55



3.5 Obfuscation and packer checking

• Base64 Encoding

- ✓ Mã hóa Base64 được sử dụng để chuyển đổi dữ liệu nhị phân (mã máy) qua một hệ thống chỉ hỗ trợ xử lý văn bản. Biểu diễn dữ liệu nhị phân dưới dạng chuỗi ASCII bằng cách chuyển đổi nó thành một dạng biểu diễn hệ số 64.
- ✓ Trong quá trình mã hóa Base64, mỗi 3 byte (24 bit) dữ liệu nhị phân được chia thành 4 đoạn dữ liệu, mỗi đoạn gồm 6 bit. Sau đó, mỗi giá trị 6 bit được ánh xạ đến một ký tự ASCII tương ứng trong bảng mã Base64. Quá trình này lặp đi lặp lại cho toàn bộ dữ liệu nhị phân đến khi còn sót dư ít hơn 3 byte.

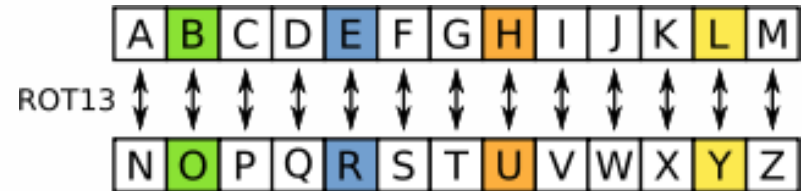
Source	Text (ASCII)	M						a						n											
	Octets	77 (0x4d)						97 (0x61)						110 (0x6e)											
Bits		0	1	0	0	1	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	1	0	
Base64 encoded	Sextets	19						22						5						46					
	Character	T						W						F						u					
	Octets	84 (0x54)						87 (0x57)						70 (0x46)						117 (0x75)					

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

3.5 Obfuscation and packer checking

- ROT13 (rotate 13)

✓ Thay thế ký tự đầu vào bằng ký tự cách nó 13 vị trí.



1	Plain-text:	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
2	ROT13 output:	UXRL_YBPNY_ZNPUVAR\Fbsgjner\Zvpebfbsg\Jvaqbjf\PheeragIrefvba\Eha
3	Lookup Table:	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
4		NOPQRSTUVWXYZABCDEFGHIJKLMnopqrstuvwxyzabcdefghijklm
5		

3.5 Obfuscation and packer checking

• STRING OBFUSCATION

Obfuscation Example	Explanation
<pre>public class HelloWorld { public static void main(String[] args) { System.out.println("Hello World!"); } }</pre>	Normal code for "Hello World!"
<pre>public class HelloWorld { public static void main(String[] args) { System.out.println("48656c6c6f20576f726c6421"); } }</pre>	Data Obfuscation with Hex Hex encoding turns "Hello World!" into 48656c6c6f20576f726c6421.
<pre>public class HelloWorld { public static void main(String[] args) { System.out.println("48","65en","6c","6c(fd","6f","2054","57g","6f5h","72 __t","6c","64'h","21"); } }</pre>	Data fragmentation Adding "" around each digit and then packing it with other additional characters - in decoding only the first two bytes are read.
<pre>public class HelloWorld { cHVibGljIHNOYXRpYyB2b2lk main(String[] args) { System.out.println("48","65en","6c","6c(fd","6f","2054","57g","6f5h","72 __t","6c","64'h","21"); } }</pre>	Code Obfuscation with Base64 Using Base64 to encode 'public static void' into cHVibGljIHNOYXRpYyB2b2lk hides the variables that determine how the "Hello World!" script is run.

3.5 Obfuscation and packer checking

- Decoding obfuscated strings using FLOSS (FireEye Labs Obfuscated String Solver)
- ✓ Xác định và phân tích obfuscated strings từ mã độc một cách tự động.



<https://github.com/fireeye/flare-floss/releases>



3.5 Obfuscation and packer checking

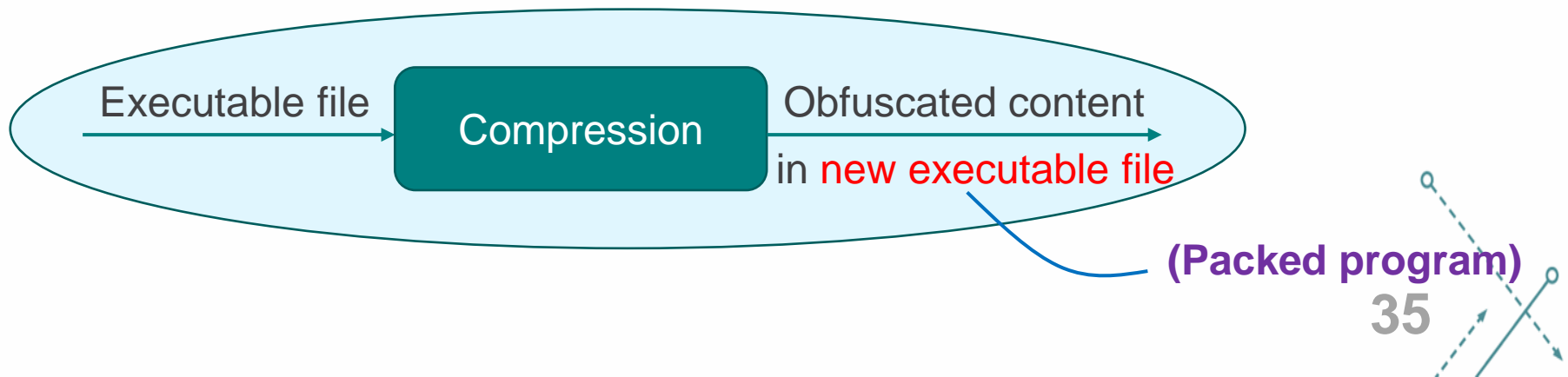
- **Obfuscated Binary:** Hầu hết các mã độc đều được che giấu để ẩn mã nguồn thực sự của một chương trình thông qua một hoặc nhiều lớp nén/mã hóa (parker/Cryptor).
- Mã độc mà đã packed hay obfuscated thường chứa rất ít chuỗi (vì nó đã bị che dấu).



3.5 Obfuscation and packer checking

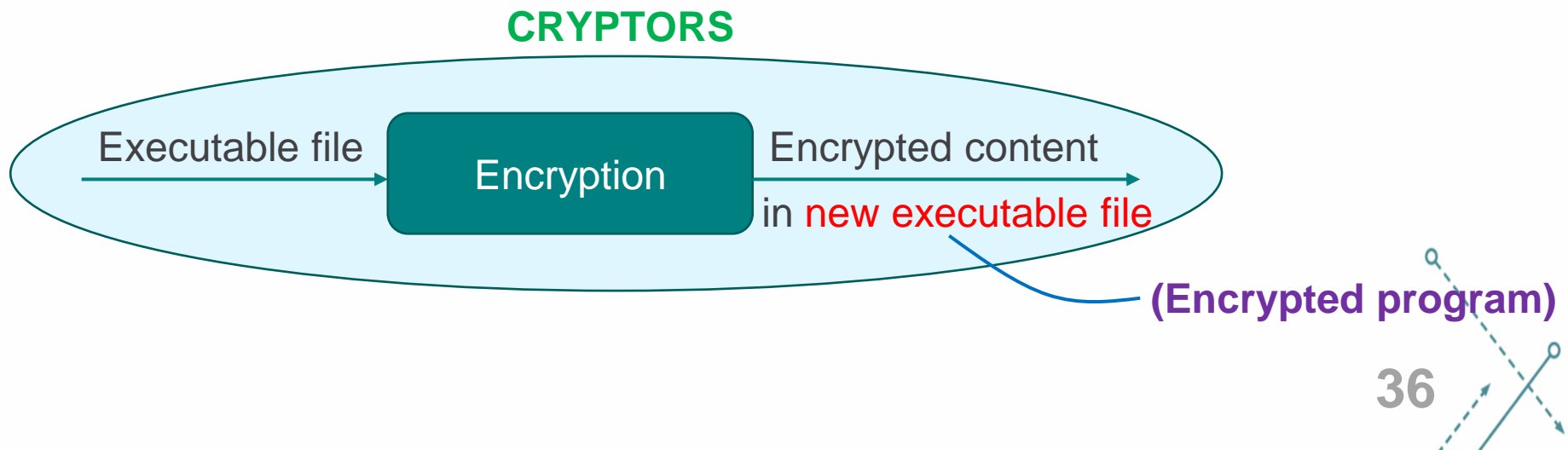
- **Packer** là một chương trình nhận chương trình thực thi làm đầu vào và sử dụng việc nén để làm mờ nội dung của chương trình thực thi. Khi thực thi chương trình đã được đóng gói, nó chạy một quá trình giải nén và trích xuất chương trình nhị phân gốc trong bộ nhớ trong khi chạy và kích hoạt việc thực thi.

PACKER



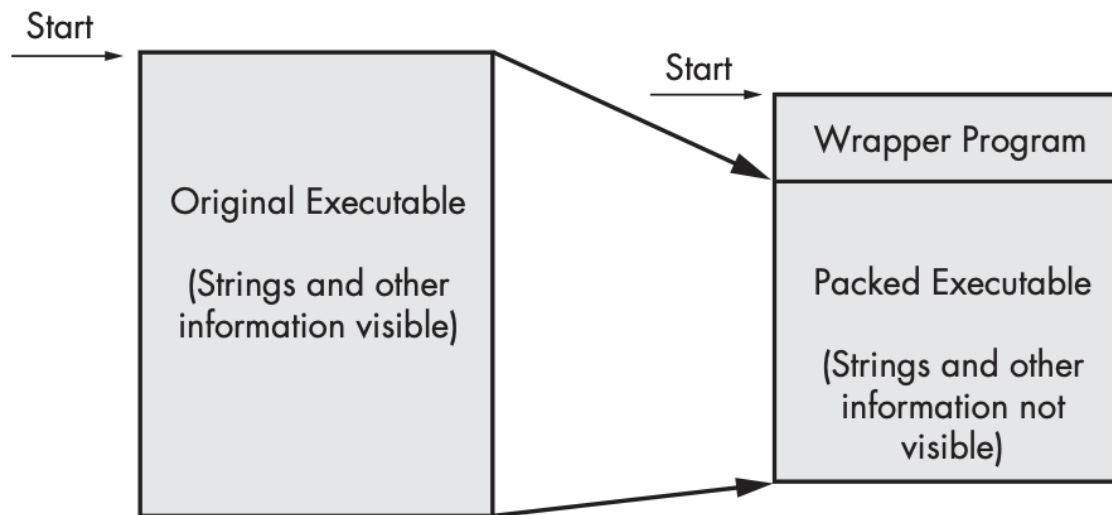
3.5 Obfuscation and packer checking

- **Cryptor** sử dụng việc mã hóa để làm mờ nội dung của chương trình thực thi. Khi thực thi chương trình đã được mã hóa, nó chạy một quá trình giải mã để trích xuất chương trình nhị phân gốc trong bộ nhớ và sau đó kích hoạt việc thực thi.



3.5 Obfuscation and packer checking

- Khi chạy chương trình được đóng gói (packed), một chương trình wrapper nhỏ cũng chạy để giải nén tập tin đã được đóng gói và sau đó chạy tập tin đã được giải nén.
- Phân tích chương trình đã được đóng gói, chỉ có thể hiểu rõ chương trình wrapper và từ đó có thể phân tích chi tiết nó.



3.5 Obfuscation and packer checking

- Phân tích chuỗi từ mã độc spybot.exe chưa đóng gói, nhận được:
 - Nhiều tên chương trình thực thi nghi ngờ.
 - Nhiều địa chỉ IP.

```
$ strings -a spybot.exe  
[....removed....]  
EDU_Hack.exe  
Sitebot.exe  
Winamp_Installer.exe  
PlanetSide.exe  
DreamweaverMX_Crack.exe  
FlashFXP_Crack.exe  
Postal_2_Crack.exe  
Red_Faction_2_No-CD_Crack.exe  
Renegade_No-CD_Crack.exe  
Generals_No-CD_Crack.exe  
Norton_Anti-Virus_2002_Crack.exe  
Porn.exe  
AVP_Crack.exe  
zoneallarm_pro_crack.exe  
[...REMOVED...]  
209.126.201.22  
209.126.201.20
```

3.5 Obfuscation and packer checking

- Đóng gói chương trình `skybot.exe` sử dụng **UPX** và lưu ra tệp mới `skybot_packed.exe`

```
$ upx -o spybot_packed.exe spybot.exe
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2013
UPX 3.91 Markus Oberhumer, Laszlo Molnar & John Reiser Sep 30th 2013
File size Ratio Format Name
-----
44576 -> 21536 48.31% win32/pe spybot_packed.exe
Packed 1 file.
```

```
$ ls -al
total 76
drwxrwxr-x 2 ubuntu ubuntu 4096 Jul 9 09:04 .
drwxr-xr-x 6 ubuntu ubuntu 4096 Jul 9 09:04 ..
-rw-r--r-- 1 ubuntu ubuntu 44576 Oct 22 2014 spybot.exe
-rw-r--r-- 1 ubuntu ubuntu 21536 Oct 22 2014 spybot_packed.exe
```

3.5 Obfuscation and packer checking

- Kết quả nhận được khi phân tích lại chuỗi từ mã độc `spybot.exe` đã đóng gói thành `skybot_packed.exe`

```
$ strings -a spybot_packed.exe
!This program cannot be run in DOS mode.
UPX0
UPX1
.rsrc
3.91
UPX!
t ;t
/t:VU
]^M
9-lh
:A$m
hAgo .
C@@f.
Q*vPCi
%_I;9
PVh29A
[...REMOVED...]
```

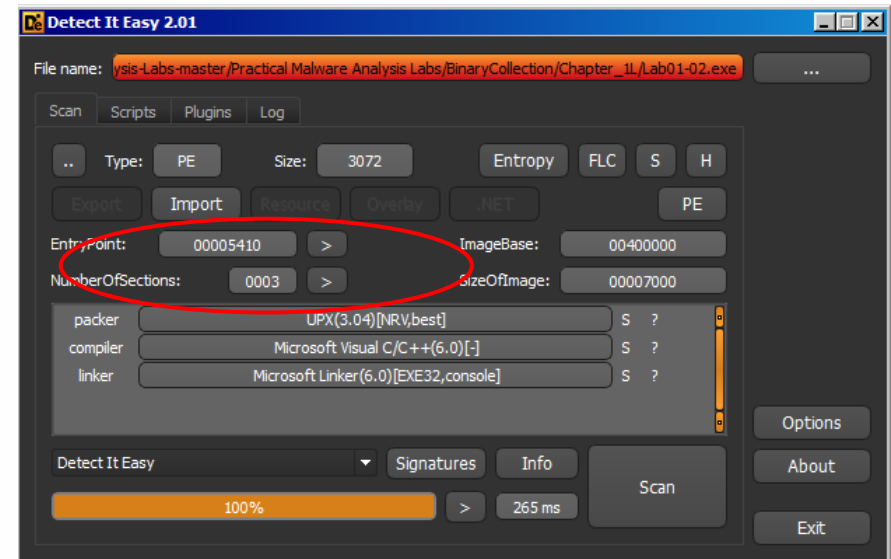
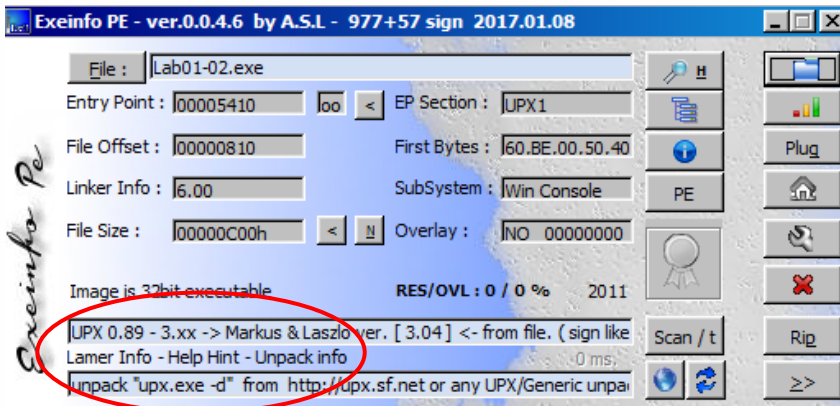

3.5 Obfuscation and packer checking

- Các dấu hiệu thường gặp:
 - Mã độc đã được packed hay obfuscated thường sẽ chứa rất ít chuỗi và ngược lại.
 - Code đã được packed và obfuscated thì thường chứa rất ít hàm **LoadLibrary** (để gọi file .dll) và **GetProcAddress** (lấy địa chỉ của một hàm trong một thư viện chia sẻ) được sử dụng để tải và truy cập các hàm bổ sung. Ngoài ra 1 số hàm khác: **LdrGetProcAddress**, **LdrLoadDll**.
 - So sánh kích thước ảo với kích thước dữ liệu thô trong header PE: Nếu kích thước ảo lớn hơn rất nhiều thì khả năng mã đã bị packed.
 - Entropy của tệp tin mà cao (>6) thì có thể đã bị packed.



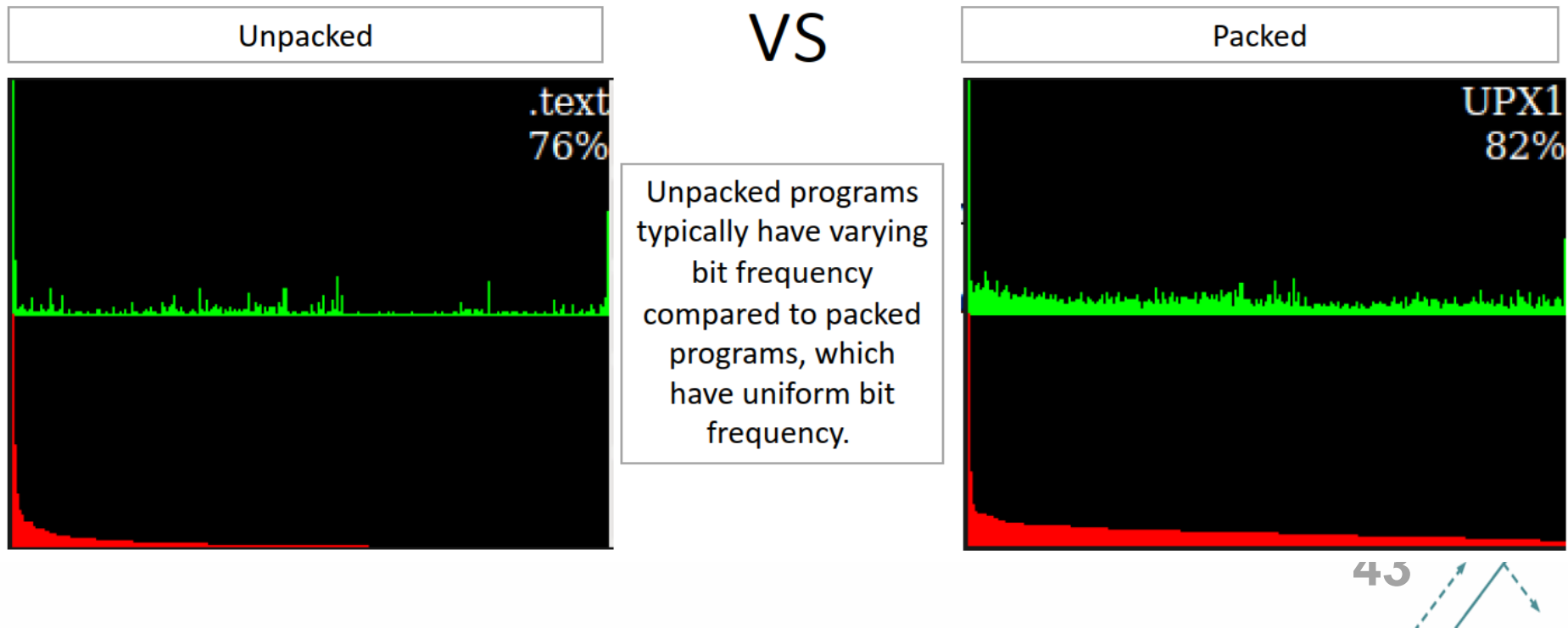
3.5 Obfuscation and packer checking

- Tools thông dụng:
 - ExeinfoPE
 - Detect It Easy (To identify encrypted file)
 - RDG Packer Detector v0.7.6
 - PEiD



3.5 Obfuscation and packer checking

Tool: Bytehist (histogram)



3.5 Obfuscation and packer checking

Tool: PEStudio (sections/permissions)

Unpacked

property	value	value	value
name	.text	.rdata	.data
md5	ED50825E62CB0AE9B7AD4C...	DAD4A33C107834511542305...	20ECDAS3D2AB9891293FF35...
entropy	6.329	4.088	3.334
file-ratio (99.21%)	9.52 %	1.98 %	1.19 %
raw-address	0x00000400	0x00003400	0x00003E00
raw-size (128000 bytes)	0x00003000 (12288 bytes)	0x00000A00 (2560 bytes)	0x00000600 (1536 bytes)
virtual-address	0x00401000	0x00404000	0x00405000
virtual-size (126903 bytes)	0x00002FB2 (12210 bytes)	0x00000892 (2194 bytes)	0x00000564 (1380 bytes)
entry-point	0x00002320	-	-
writable	-	-	x
executable	x	-	-

VS

Packed

property	value	value	value
name	UPX0	UPX1	UPX2
md5	n/a	4EEF9B8D9D0CB60D7D19DF...	48F04F5E64C9D1428BA25FC...
entropy	n/a	7.827	1.708
file-ratio (56.08%)	n/a	54.38 %	1.70 %
raw-address	0x00000200	0x00000200	0x00004200
raw-size (16896 bytes)	0x00000000 (0 bytes)	0x00004000 (16384 bytes)	0x00000200 (512 bytes)
virtual-address	0x00401000	0x00414000	0x00418000
virtual-size (98304 bytes)	0x00013000 (77824 bytes)	0x00004000 (16384 bytes)	0x00001000 (4096 bytes)
entry-point	-	0x00017D70	-
writable	x	x	x
executable	x	x	-

Tool: PEStudio (sections/size)

Unpacked

property	value	value	value
name	.text	.rdata	.data
md5	ED50825E62CB0AE9B7AD4C...	DAD4A33C107834511542305...	20ECDA53D2AB9891293FF35...
entropy	6.329	4.088	3.334
file-ratio (99.21%)	9.52 %	1.98 %	1.19 %
raw-address	0x00000400	0x00002400	0x00003E00
raw-size (128000 bytes)	0x00003000 (12288 bytes)	0x00001800 (6144 bytes)	0x00000600 (1536 bytes)
virtual-address	0x00401000	0x00401000	0x00405000
virtual-size (126903 bytes)	0x00002FB2 (12210 bytes)	0x00000892 (2154 bytes)	0x00000564 (1380 bytes)

Similar Size

VS

Packed

property	value	value	value
name	UPX0	UPX1	UPX2
md5	n/a	4EEE9B8D9D0CB60D7D19DF...	4BF04E5E64C9D142BBA25FC...
entropy	n/a	7.827	1.708
file-ratio (56.08%)	n/a	54.38 %	1.70 %
raw-address	0x00000200	0x00000200	0x00004200
raw-size (16896 bytes)	0x00000000 (0 bytes)	0x00000200 (512 bytes)	0x00000200 (512 bytes)
virtual-address	0x00401000	0x00401000	0x00418000
virtual-size (98304 bytes)	0x00013000 (77824 bytes)	0x00013000 (77824 bytes)	0x00010000 (4096 bytes)

Different Size

Tool: PEStudio (sections/entropy)

Unpacked

property	value	value	value
name	.text	.rdata	.data
md5	ED50825E62CB0AE9B7AD4C...	DAD4A33C107834511542305...	20ECDA53D2AB9891293FF35...
entropy	6.329	4.088	3.334
file-ratio (99.21%)	9.52 %	1.98 %	1.19 %
raw-address	0x00000400	0x00003400	0x00003E00
raw-size (128000 bytes)	0x00003000 (12288 bytes)	0x00000A00 (2560 bytes)	0x00000600 (1536 bytes)
virtual-address	0x00401000	0x00404000	0x00405000
virtual-size (126903 bytes)	0x00002FB2 (12210 bytes)	0x00000892 (2194 bytes)	0x00000564 (1380 bytes)

VS

Packed

property	value	value	value
name	UPX0	UPX1	UPX2
md5	n/a	4EEE9B8D9D0CB60D7D19DF...	4BF04E5E64C9D142BBA25FC...
entropy	n/a	7.827	1.708
file-ratio (56.08%)	n/a	54.38 %	1.70 %
raw-address	0x00000200	0x00000200	0x00004200
raw-size (16896 bytes)	0x00000000 (0 bytes)	0x00004000 (16384 bytes)	0x00000200 (512 bytes)
virtual-address	0x00401000	0x00414000	0x00418000
virtual-size (96304 bytes)	0x00013000 (77824 bytes)	0x00004000 (16384 bytes)	0x00001000 (4096 bytes)

3.7 Tập tin PE

- Các tệp thực thi Windows (như .exe, .dll, .sys, .ocx và .drv) phải tuân thủ theo định dạng PE (Portable Executable).
- **Tập PE** là một loạt các cấu trúc và thành phần con chứa thông tin cần thiết cho hệ điều hành để tải nó vào bộ nhớ.
- **PE header** chứa cấu trúc tệp PE, do đó nó được sử dụng bởi trình tải hệ điều hành khi tệp binary được thực thi để lấy nội dung ghi vào bộ nhớ.
- PE header chứa thông tin như nơi tệp thực thi cần được tải vào bộ nhớ, địa chỉ bắt đầu thực thi, danh sách thư viện/chức năng mà ứng dụng phụ thuộc vào, và các tài nguyên được sử dụng bởi tệp thực thi (binary).

3.7 Tập tin PE

- Công cụ:
- CFF Explorer: <http://www.ntcore.com/exsuite.php>
- PE Internals: <http://www.andreybazhan.com/pe-internals.html>
- PPEE(puppy): <https://www.mzrst.com/>

3.7 Tập tin PE

- Mã độc khi hoạt động cần phải tương tác với các tệp, registry, mạng và các thành phần khác.
- Do vậy, mã độc cần sử dụng các hàm của hệ điều hành, vd trên Windows là Application Programming Interfaces (API) và được cung cấp trong các tệp Dynamic Link Library (DLL).



3.7 Tập tin PE

- Các tệp thực thi sẽ nhập (import) và gọi (call) các hàm này từ các DLL khác nhau, chúng cung cấp các chức năng khác nhau.
- Các hàm mà tệp thực thi nhập từ các tệp khác (phần lớn là DLL) được gọi là các hàm nhập (hoặc **imports**).
- VD: Nếu mã độc muốn tạo một file trên ổ đĩa Windows nó cần sử dụng API `CreateFile()`, được cung cấp bởi `kernel32.dll`, nó cần tải `kernel32.dll` vào bộ nhớ và sau đó gọi hàm `CreateFile()`.



3.7 Tập tin PE

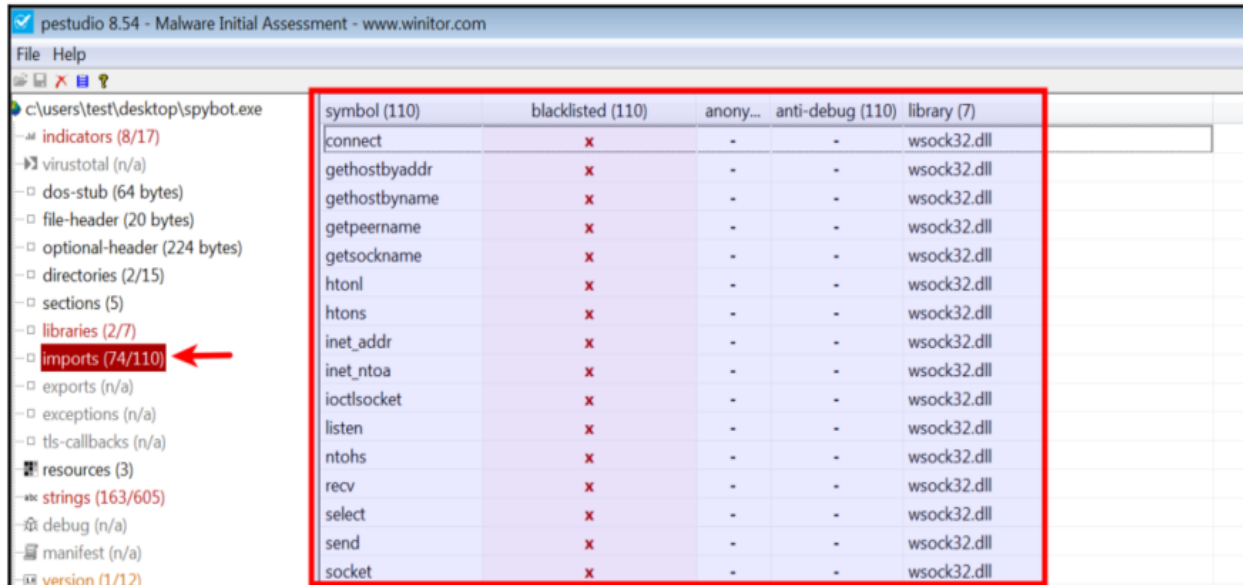
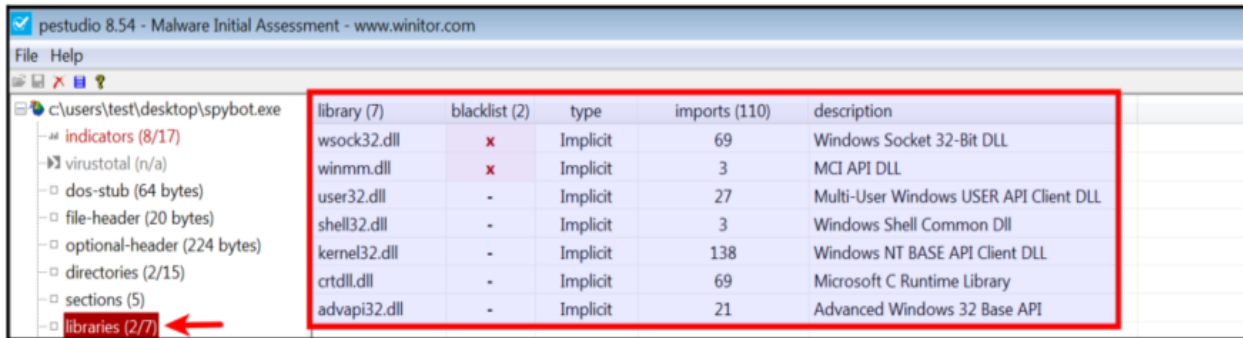
❓ Kiểm tra các **imports** có thể:

- cung cấp một thông tin về chức năng và khả năng của mã độc và giúp dự đoán những hoạt động của nó trong quá trình thực thi.
- Xác định xem mã độc có được che giấu hay không?



3.7 Tập tin PE

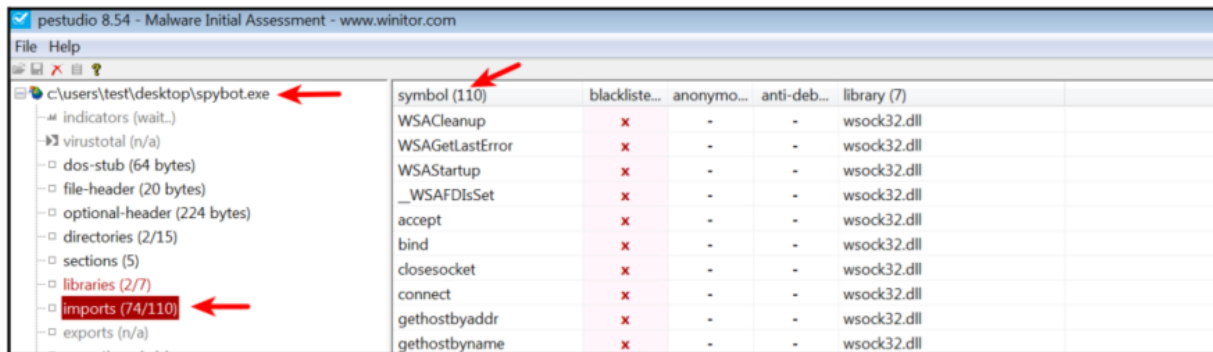
- Xác định các chức năng của mã độc



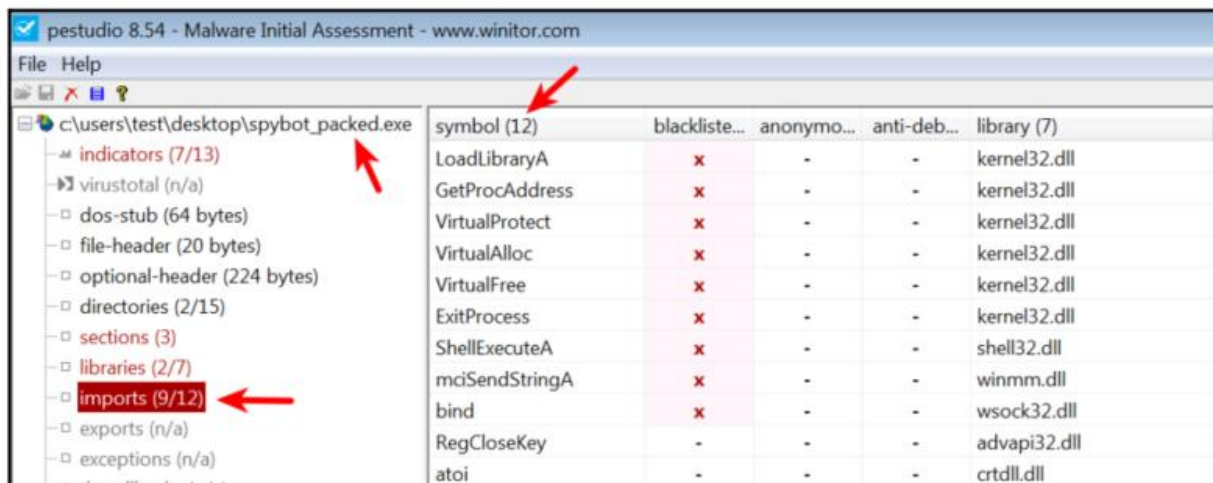
API: connect, socket, listen, send... nhập từ wsock32.dll □
mã độc cần kết nối internet và các hoạt động mạng.

3.7 Tập tin PE

- Xác định xem mã độc có được che giấu.



Mã độc sử dụng rất ít imports □ khả năng cao là đã được đóng gói (packed).



pestudio 8.54 - Malware Initial Assessment - www.winitor.com

File Help

c:\users\test\desktop\rmn.dll

index name (22) address blacklist... duplicate... anonymo... gap (0) forwarder...

22 RemoveDevice 0x000019F0 - - - -

21 RegisterCoInstaller_EX 0x00002E20 - - - -

20 RegisterCoInstaller 0x00001A... - - - -

19 KillProcess 0x000014B0 - - - -

18 InstallDrvFiles 0x00002F00 - - - -

17 GetProcessID 0x000014... - - - -

16 GetOS 0x00002470 - - - -

15 EnumerateDevice 0x000019E0 - - - -

14 EditRegistry 0x000017E0 - - - -

13 DuplicateFile 0x000019B0 - - - -

12 DeleteRegistryforME 0x000022E0 - - - -

3.7 Tập tin PE

- **USER32.DLL:** Quản lý giao diện người dùng Windows. Chịu trách nhiệm về các yếu tố như cửa sổ, nút bấm, thanh cuộn,...
- **KERNEL32.DLL:** Cung cấp các hàm hệ thống cơ bản cho ứng dụng Windows. Bao gồm các chức năng như thời gian, bộ nhớ,...
- **GDI32.DLL:** Quản lý đồ họa 2D cho Windows. Cung cấp các chức năng vẽ hình, chữ,...
- **ADVAPI32.DLL:** Cung cấp các chức năng bảo mật cho ứng dụng Windows.
- **OLE32.DLL:** Quản lý các đối tượng OLE và COM trong Windows.
- **ODBC32.DLL:** Hỗ trợ truy cập cơ sở dữ liệu thông qua giao diện ODBC.
- **MSCTF.DLL:** Hỗ trợ xử lý chữ cho ứng dụng Windows.
- **SHELL32.DLL:** Quản lý giao diện vỏ của Windows Explorer và các chức năng khác.
- **RPCRT4.DLL:** Hỗ trợ giao tiếp từ xa thông qua Microsoft RPC.

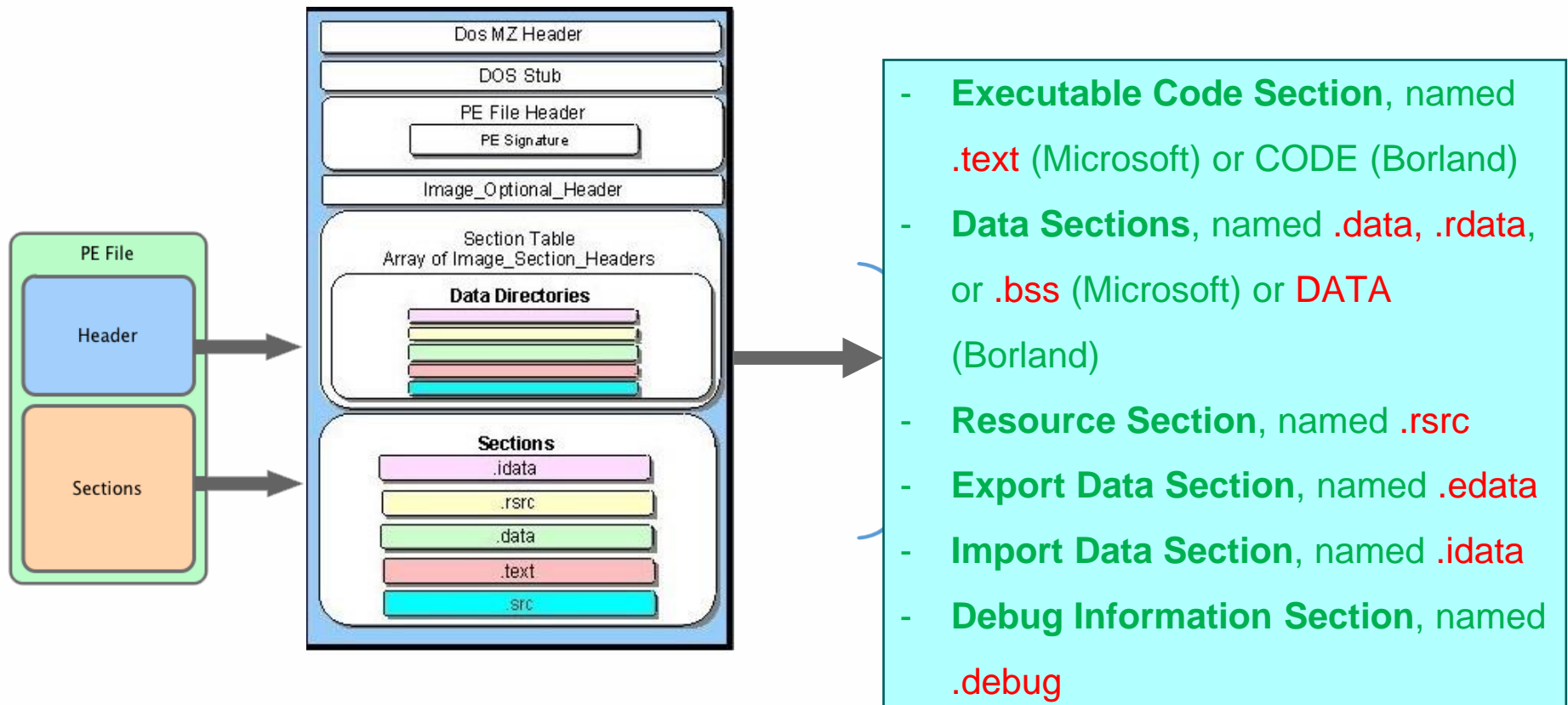


3.7 Tập tin PE

- **KERNEL32.DLL**: Cung cấp các hàm hệ thống cơ bản cho ứng dụng Windows. Bao gồm các chức năng như thời gian, bộ nhớ,...
- **WS2_32.DLL & WSOCK32.DLL** : Chức năng mạng (TCP/IP)
- **MSVCRT.DLL**: chứa code bọc - wrapper code - cho phép code được biên dịch gọi các hàm chuẩn C mà không cần quan tâm đến gọi hệ thống thực
- **USER32.DLL**: Quản lý giao diện người dùng Windows. Chịu trách nhiệm về các yếu tố như cửa sổ, nút bấm, thanh cuộn,...
- **GDI32.DLL**: Quản lý đồ họa 2D cho Windows. Cung cấp các chức năng vẽ hình, chữ,...
- **ADVAPI32.DLL**: Cung cấp các chức năng bảo mật cho ứng dụng Windows.
- **OLE32.DLL**: Quản lý các đối tượng OLE và COM trong Windows.
- **MSCTF.DLL**: Hỗ trợ xử lý chữ cho ứng dụng Windows.
- **SHELL32.DLL**: Quản lý giao diện vỏ của Windows Explorer và các chức năng khác.
- **RPCRT4.DLL**: Hỗ trợ giao tiếp từ xa thông qua Microsoft RPC.

3.7 Tập tin PE

- Bảng phân vùng và các phân vùng trong tập tin PE



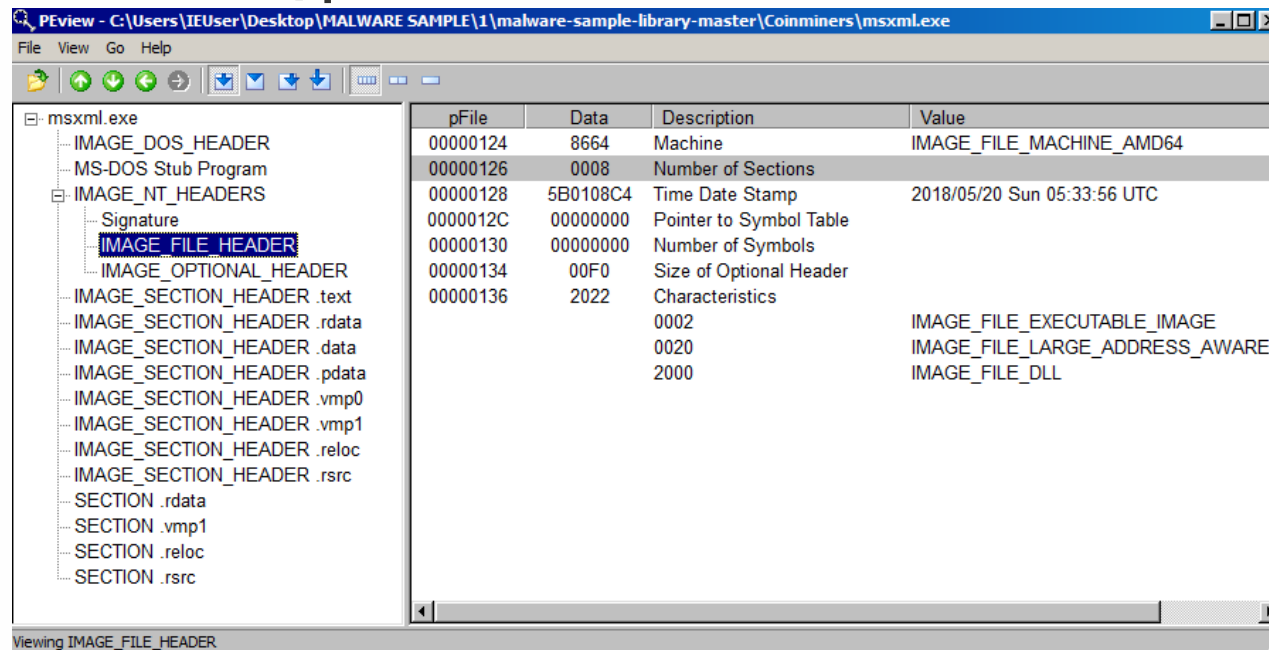
3.7 Tập tin PE

- Các Sections chung trong một tập PE

Section Name	Description
<code>.text</code> or <code>CODE</code>	Chứa mã thực thi
<code>.data</code> or <code>DATA</code>	Thường chứa dữ liệu có thể đọc/ghi và biến toàn cục.
<code>.rdata</code>	Chứa dữ liệu chỉ đọc. Đôi khi cũng chứa thông tin nhập và xuất.
<code>.idata</code>	Nếu có, chứa bảng nhập. Nếu không, thông tin nhập được lưu trong section <code>.rdata</code> .
<code>.edata</code>	Nếu có, chứa thông tin xuất. Nếu không, thông tin xuất được tìm thấy trong section <code>.rdata</code> .
<code>.rsrc</code>	Phần này chứa các tài nguyên được sử dụng bởi tập thực thi như biểu tượng, hộp thoại, menu, chuỗi ...

3.7 Tập tin PE

- Kiểm tra tập tin PE với PEsview



IMAGE_DOS_HEADER, MS-DOS Stub, Signature □ có thể bỏ qua

IMAGE_FILE_HEADER □ chứa thông tin cơ bản về tập

IMAGE_OPTIONAL_HEADER □ mô tả các thông số tùy chọn

IMAGE_SECTION_HEADER □ **virtual size**: không gian được cấp phát cho phân vùng trong quá trình nạp module; **Size of Raw Data**: kích thước thực tế của phân vùng như được lưu trữ trên đĩa.

3.8 Phân loại mã độc

- Phân tích mẫu mã độc có thể cho biết mẫu đó có thuộc họ mã độc (family), hoặc có tính chất giống với những mẫu đã phân tích trước đó hay không. Cụ thể:
 - Phân tích mẫu mã độc có thể xác định nó thuộc gia đình malware nào, chẳng hạn như ransomware, trojan, backdoor, ...
 - So sánh mẫu đó với các mẫu đã biết để xác định nó có đặc trưng giống với mẫu đã biết trước đó hay không, ví dụ so sánh mã, hành vi, kỹ thuật lây lan, ...
 - Nếu so sánh có kết quả gần giống thì có thể kết luận đó là một mẫu thuộc cùng gia đình (family) hoặc biến thể của một mã đã biết.



Phân loại mã độc sử dụng Fuzzy Hashing

- Fuzzy hashing là một phương pháp để so sánh sự tương đồng giữa các tập tin.
- Kỹ thuật này so sánh một nhị phân nghi ngờ với các mẫu trong kho lưu trữ để xác định các mẫu tương tự □ xác định các mẫu thuộc cùng gia đình malware hoặc cùng nhóm tác nhân.



Phân loại mã độc sử dụng Fuzzy Hashing

- ssdeep (<http://ssdeep.sourceforge.net>) là một công cụ hữu ích để tạo ra fuzzy hash cho một mẫu, và nó cũng giúp xác định tỷ lệ tương đồng % giữa các mẫu.

```
$ ssdeep veri.exe
```

```
ssdeep,1.1--blocksize:hash:hash,filename
```

```
49152:op398U/qCazcQ3iEZgcwwGF0iWC28pUtu6On2spPHlDB:op98USfcy8cwF2bC28pUtsRp  
tDB, "/home/ubuntu/Desktop/veri.exe"
```

- Ví dụ: Thư mục chứa 3 mã độc với mã MD5 khác nhau

```
48c1d7c541b27757c16b9c2c8477182b aiggs.exe  
92b91106c108ad2cc78a606a5970c0b0 jnas.exe  
ce9ce9fc733792ec676164fc5b2622f2 veri.exe
```

1. So sánh 2 tệp

```
$ ssdeep -pb *
```

```
aiggs.exe matches jnas.exe (99)
```

```
jnas.exe matches aiggs.exe (99)
```



Phân loại mã độc sử dụng Fuzzy Hashing

2. So sánh nhiều mẫu trong 1 thư mục

```
$ ssdeep -lrpa samples/  
samples//aiggs.exe matches samples//crop.exe (0)  
samples//aiggs.exe matches samples//jnas.exe (99)  
  
samples//crop.exe matches samples//aiggs.exe (0)  
samples//crop.exe matches samples//jnas.exe (0)  
  
samples//jnas.exe matches samples//aiggs.exe (99)  
samples//jnas.exe matches samples//crop.exe (0)
```

3. So sánh sử dụng mã hash

```
$ ssdeep * > all_hashes.txt  
$ ssdeep -m all_hashes.txt blab.exe  
/home/ubuntu/blab.exe matches all_hashes.txt:/home/ubuntu/aiggs.exe (99)  
/home/ubuntu/blab.exe matches all_hashes.txt:/home/ubuntu/jnas.exe (100)
```

Phân loại mã độc sử dụng Import Hash (imphash)

- phân tích mã độc sử dụng các hàm API (hay còn gọi là imphash) là một kỹ thuật có thể được sử dụng để xác định mẫu liên quan và mẫu được sử dụng bởi các nhóm tác nhân đe dọa.
- Đặc điểm này dựa trên việc tính toán giá trị băm dựa trên tên hàm/ hàm nhập (API) và thứ tự của chúng trong tập tin thực thi. Nếu các tập tin được biên dịch từ cùng một nguồn và theo cùng một cách, những tập tin đó sẽ có giá trị imphash giống nhau.
- Trong quá trình điều tra mã độc, nếu phát hiện các mẫu có cùng giá trị imphash, điều đó có nghĩa chúng có cùng bảng địa chỉ nhập và có khả năng liên quan đến nhau.



Phân loại mã độc sử dụng Import Hash (imphash)

```
$ md5sum *
```

```
3e69945e5865ccc861f69b24bc1166b6 maxe.exe
```

```
1f92ff8711716ca795fbd81c477e45f5 sent.exe
```

```
$ python get_imphash.py samples/maxe.exe
```

```
b722c33458882a1ab65a13e99efe357e
```

```
$ python get_imphash.py samples/sent.exe
```

```
b722c33458882a1ab65a13e99efe357e
```

pestudio 8.89 - Malware Initial Assessment - www.winitor.com [c:\users\ieuser\desktop\malware sample\2\malware-samples-master\eternalrocks\cf8533

property	value
md5	C52F20A854EFB013A0A1248FD84AAA95
sha1	8A2CFE220EEBDE096C17266F1BA597A1065211AB
sha256	CF8533849EE5E82023AD7ADBDBD6543CB6DB596C53048B1A0C00B3643A72DB30
md5-without-overlay	wait...
sha1-without-overlay	wait...
sha256-without-overlay	wait...
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
first-bytes-text	M Z@.....
size	5275648 (bytes)
size-without-overlay	wait...
entropy	7.695
imphash	F34D5F2D4577ED6D9CEEC516C1F5A744
signature	/
entry-point-hex	n/a
file-version	1.0.0.0

Phân loại mã độc sử dụng Section Hash

- Tương tự như Import Hash, Section Hash cũng giúp xác định các mã độc có liên quan đến nhau.

pestudio 8.89 - Malware Initial Assessment - www.winitor.com [c:\users\ieuser\desktop\malware sample\2\malware-samples-master\eternalrocks\cf8533849ee5e82023ad7adbdbd6543c...

property	value	value	value	value	value
name	EQUR%	text	.rsrc	.reloc	n/a
md5	BE04A4BF7D19CFE17A3BF94...	4B1021B1C5D6298239D549E...	F11B5096C69A2FB241765D7...	6CD33443FF7F9FFF530DF44...	72E3077865EE5...
file-ratio (99.98 %)	89.77 %	10.16 %	0.03 %	0.01 %	0.01 %
file-cave (1140 bytes)	4736000 bytes	536064 bytes	1536 bytes	512 bytes	512 bytes
entropy	8.000	4.335	4.173	0.078	0.122
raw-address	0x00000400	0x00484800	0x00507600	0x00507C00	0x00507E00
raw-size (5274624 bytes)	0x00484400 (4736000 bytes)	0x00082E00 (536064 bytes)	0x00000600 (1536 bytes)	0x00000200 (512 bytes)	0x00000200 (512 bytes)
virtual-address	0x00402000	0x00888000	0x0090C000	0x0090E000	0x00910000
virtual-size (5273484 bytes)	0x004843F0 (4735984 bytes)	0x00082D90 (535952 bytes)	0x000005F0 (1520 bytes)	0x0000000C (12 bytes)	0x00000010 (16 bytes)
entry-point (0x0051000A)	-	-	-	-	-

Phân loại mã độc sử dụng Yara

- YARA là một công cụ mạnh mẽ để nhận diện và phân loại mã độc.
- Có thể tạo các quy tắc YARA dựa trên thông tin văn bản hoặc nhị phân có trong mẫu mã độc.
- Các quy tắc YARA này bao gồm một tập hợp các chuỗi và một biểu thức logic. Những quy tắc sau khi được viết có thể dùng để quét tệp tin bằng công cụ YARA hoặc có thể sử dụng yara-python để tích hợp với các công cụ khác.
- <http://virustotal.github.io/yara/>
- <http://yara.readthedocs.io/en/v3.7.0/writingrules.html>



Phân loại mã độc sử dụng Yara

- Quy tắc YARA bao gồm các thành phần sau:
 - Rule identifier (Định danh quy tắc): Đây là tên mô tả quy tắc. Các định danh quy tắc có thể chứa bất kỳ ký tự chữ và số nào cùng với ký tự gạch dưới, nhưng ký tự đầu tiên không thể là một chữ số. Các định danh quy tắc phân biệt chữ hoa chữ thường và không vượt quá 128 ký tự.
 - String Definition (Định nghĩa chuỗi): Đây là phần mà các chuỗi sẽ được định nghĩa và sử dụng trong quy tắc. Phần này có thể bị bỏ qua nếu quy tắc không phụ thuộc vào bất kỳ chuỗi nào. Mỗi chuỗi có một định danh được tạo bởi ký tự \$ theo sau là một chuỗi gồm các ký tự chữ và số cùng với ký tự gạch dưới.
 - Condition Section (Phần điều kiện): là nơi logic của quy tắc được đặt. Phần này phải chứa một biểu thức Boolean xác định điều kiện mà quy tắc sẽ khớp hoặc không khớp.

Phân loại mã độc sử dụng Yara

- Ví dụ luật YARA

```
rule suspicious_strings
{
  strings:
    $a = "Synflooding"
    $b = "Portscanner"
    $c = "Keylogger"

  condition:
    ($a or $b or $c)
}
```

- Khởi chạy YARA

```
$ yara -r suspicious.yara samples/
suspicious_strings samples//spybot.exe
suspicious_strings samples//wuamqr.exe
```

