

Mã độc

Chương 7. Các kỹ thuật chống phân tích

Mục tiêu

- Giới thiệu, phân tích một số kỹ thuật chống phân tích thường gặp trong mã độc

2

Tài liệu tham khảo

[1] Michael Sikorski, Andrew Honig, 2012, Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software, No Starch Press, (ISBN: 978-1593272906).

[2] Sam Bowne, Slides for a college course at City College San Francisco, https://samsclass.info/126/126_S17.shtml

3

Nội dung

1. Nén
2. Mã hóa
3. Làm rối mã
4. Một số cơ chế khác

4

Nội dung

1. Nén
2. Mã hóa
3. Làm rối mã
4. Một số cơ chế khác

5

Nén

- ☐ Packer (nén) là một kiểu chương trình nén hoặc che giấu file thực thi (executable file).
- ☐ Giảm kích thước của file, làm cho việc tải file nhanh hơn.

6

Nén

- ❑ Packer nén, mã hóa, lưu hoặc dấu code gốc của chương trình, tự động bổ sung một hoặc nhiều section, sau đó sẽ thêm đoạn code Unpacking Stub và chuyển hướng Entry Point (EP) tới vùng code này.
- ❑ Một file không đóng gói (Nonpacked) sẽ được tải bởi OS.

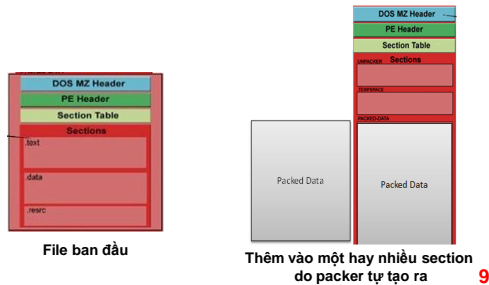
7

Nén

- ❑ File đã đóng gói: Unpacking Stub sẽ được tải bởi OS, sau đó chương trình gốc sẽ tải Unpacking Stub.
- ❑ Code EP của file thực thi sẽ trở tới Unpacking Stub thay vì trở vào mã gốc.
- ❑ Trong môi trường DOS, chương trình sẽ kiểm tra DOS header và nếu hợp lệ sẽ thực thi DOS Stub. Bình thường file chạy trên Windows thì 2 trường này có thể bỏ qua.

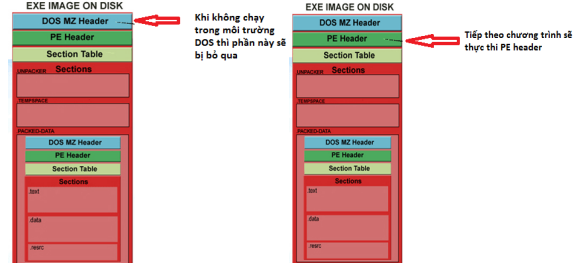
8

Nguyên lí hoạt động của Packer



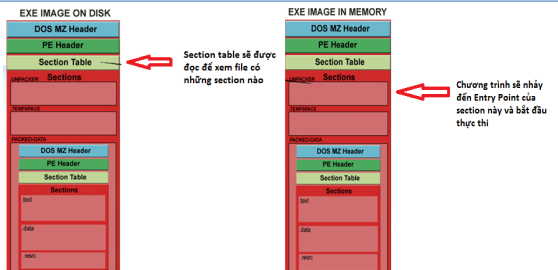
9

Thực thi tệp đã nén



10

Thực thi tệp đã nén



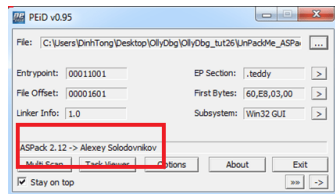
11

Thực thi tệp đã nén

- ❑ Nhảy đến Entry Point của section UNPACKER, lưu lại toàn bộ giá trị của các thanh ghi bằng lệnh PUSHAD.
- ❑ Tính toán lại Import Table.
- ❑ Khôi phục lại giá trị các thanh ghi đã được lưu trong Stack bằng lệnh POPAD.
- ❑ Nhảy đến Origin EntryPoint và thực thi như file lúc chưa bị đóng gói.

12

Nhận biết một tệp tin bị nén



13

Giải nén

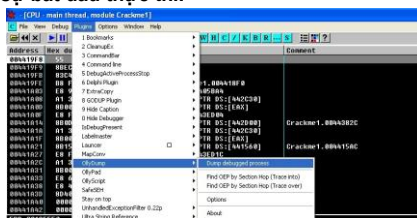
□ Các bước cơ bản để thực hiện giải nén



14

Tìm OEP

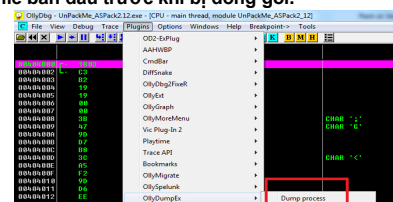
□ Original entry point là nơi mà chương trình gốc thực sự bắt đầu thực thi.



15

Dump file

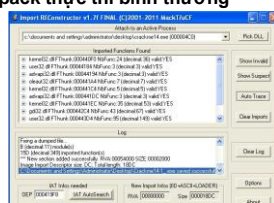
□ Sau khi nhảy đến OEP ta sẽ tiến hành dump file. Mục đích của việc này là fix lại các section và import table như file ban đầu trước khi bị đóng gói.



16

Fix IAT

□ Kiểm tra file xem còn các cơ chế Anti hoặc ngăn chặn việc thực thi hay không rồi tiến hành chỉnh sửa. Đảm bảo file sau unpack thực thi bình thường



17

Nội dung

1. Nén
2. Mã hóa
3. Làm rối mã
4. Một số cơ chế khác

18

Mã hóa

Lý do mã độc sử dụng các biện pháp mã hóa:

- ☐ Che giấu thông tin cấu hình, chẳng hạn như C&C Domain.
- ☐ Lưu thông tin vào tệp tin trước khi đánh cắp
- ☐ Che giấu mã độc bên trong một công cụ hợp pháp
- ☐ Ẩn các chuỗi bị nghi ngờ

19

Mã hóa

- ☐ Thuật toán mã hóa đơn giản.
- ☐ Thuật toán mã hóa mạnh
- ☐ Thuật toán mã hóa tùy chỉnh

20

Mã hóa

- ☐ Thuật toán mã hóa đơn giản
- ☐ Thuật toán mã hóa mạnh
- ☐ Thuật toán mã hóa tùy chỉnh

21

Thuật toán mã hóa đơn giản

- ☐ Kích thước nhỏ, phù hợp với các môi trường bị ràng buộc như khai thác bằng shellcode
- ☐ Ít ảnh hưởng đến hiệu năng
- ☐ Gây khó khăn trong việc phát hiện, tuy nhiên không thể qua mặt được các nhà phân tích có kỹ năng tốt

22

Mã Caesar

- ☐ Dịch chuyển từng chữ cái về trước 3 vị trí trong bảng chữ cái alphabet

ABCDEFGHIJKLMNOPQRSTUVWXYZ
 DEFGHIJKLMNOPQRSTUVWXYZABC

- ☐ Ví dụ

ATTACK AT NOON
 DWWDEN DW QRRQ

23

XOR

- ☐ Sử dụng một khóa để mã hóa
- ☐ Sử dụng một bit của dữ liệu và một bit của khóa trong một thời điểm
- ☐ Ví dụ: Encode Encode HI với khóa 0x3C

HI = 0x48 0x49 (ASCII encoding)

Data: 0100 1000 0100 1001
 Key: 0011 1100 0011 1100
 Kết quả: 0111 0100 0111 0101

| | | | | |
|---|-----|---|---|---|
| 0 | xor | 0 | = | 0 |
| 0 | xor | 1 | = | 1 |
| 1 | xor | 0 | = | 1 |
| 1 | xor | 1 | = | 0 |

24

XOR

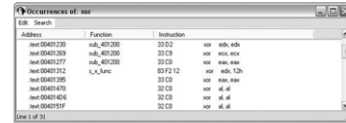
| | | | | | | | | | | | | |
|------|------|------|------|------|------|--|------|------|------|------|------|------|
| A | T | T | A | C | K | | A | T | | N | O | N |
| 0x41 | 0x54 | 0x54 | 0x41 | 0x43 | 0x48 | | 0x41 | 0x54 | 0x20 | 0x4E | 0x4F | 0x4E |
| ↓ | | | | | | | | | | | | |
| J | h | h | J | DEL | W | | FS | J | H | FS | r | s |
| 0x7A | 0x68 | 0x68 | 0x7A | 0x7F | 0x57 | | 0x1C | 0x7A | 0x68 | 0x1C | 0x72 | 0x71 |

The string ATTACK AT NOON encoded with an XOR of 0x3C
(original string at the top; encoded strings at the bottom)

25

Xác định vòng XOR trong IDA Pro

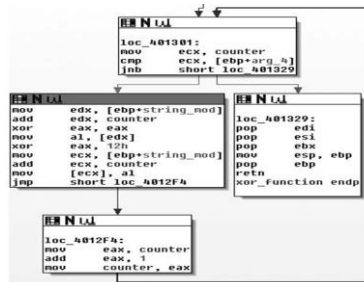
- ❑ Các vòng lặp với lệnh XOR bên trong
- ❑ Bắt đầu với “IDA View” (xem code)
- ❑ Click Search, Text
- ❑ Nhập xor và tìm tất cả các lần xuất hiện



Searching for XOR in IDA Pro

26

Xác định vòng XOR trong IDA Pro



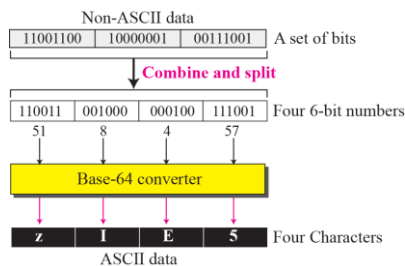
27

Base64

- ❑ Chuyển đổi 6 bits thành một ký tự trong bảng chữ cái 64 ký tự
- ❑ Sử dụng các khối 3 byte (24 bits)
- ❑ Chia thành 4 trường 6-bits
- ❑ Chuyển từng trường thành base64

28

Base64



29

Base64

Table 23.5 Base-64 Converting Table

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

30

Base64

Part of raw email message showing Base64 encoding

```
Content-Type: multipart/alternative;
  boundary="--002_4E36898966D7448815A3216ACF82AA201ED633ED1MBX3THNDRBIRD_"
MIME-Version: 1.0
--002_4E36898966D7448815A3216ACF82AA201ED633ED1MBX3THNDRBIRD_
Content-Type: text/html; charset=utf-8
Content-Transfer-Encoding: base64
```

SNYgeW91IGFYsBzyKwFkAM5nIHROaXMsIHlVdSwcnc91YwJSeSbZaG91bGQnVdzCBza2lWiRoAX
MgY2hhcHRlctBhbmQzG8dG8dGh1IG5lHQgBz5Li8EbyB5b3UgcmlVhbGxSXIghdmUgdGh1IHRp
bmUgdG8gdGhlwzSBOaGlzIHdb2xlIHN0cmUzYzBpbg9w91IGFYsBzyKwFkAM5nIHROaXMsIHlVdSwcnc91YwJSeSbZaG91bGQnVdzCBza2lWiRoAX
QuIE1heWJlIHlVdS8zaG91bGQy9vdGFjdCB0aGUyYXV0aG9ycyBhbmQzc2VlIGlnTH

31

Base64

```
GET /X29tbVEuYC8=/index.htm
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)
Host: www.practicalmalwareanalysis.com
Connection: Keep-Alive
Cookie: Ym90NTQxNjQ
```

```
GET /c2UsYi1kYmW0cnUjdF1vb1AjB21wbFU0YP==/index.htm
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)
Host: www.practicalmalwareanalysis.com
Connection: Keep-Alive
Cookie: Ym90NTQxNjQ
```

URL và Cookie được mã hóa bằng Base64

32

Giải mã các URLs

☐ Tùy chỉnh “indexing string:

aABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
rstuvwxy

❑ Tìm ký tự = ở chuỗi đã được encoding

X29tbVEuYC8= → _omnQ.~ /

c2UsYi1kYWM0cnUjdFlvb1Ajb21wbFU0YP== → se,b-dac4ru#tYon #omplU4'

Unsuccessful attempt to decode Base64 string due to nonstandard indexing string

33

Giải mã các URLs

X29tbVEuYC8= → command

`c2UsYi1kYWM0cnUjdFlvbiAjb21wbFU0YP==` → self-destruction complete

Successful decoding of Base64 string using custom indexing string

34

Mã hóa

❑ Thuật toán mã hóa đơn giản

Thuật toán mã hóa mạnh

❑ Thuật toán mã hóa tùy chỉnh

35

Các thuật toán mã hóa mạnh

- ❑ AES, RSA,


- ❑ Chống lại các tấn công brute-force.

☐ Các thư viện mật mã chuẩn dễ bị phát hiện (các hàm được import, function matching, các hằng số mật mã).

☐ Yêu cầu nhiều tài nguyên tính toán

36

Các thuật toán mã hóa mạnh



| Address | Ordinal | Name | Library |
|----------|---------|--------------------|----------|
| 00401068 | | FindNameKeyEak | ADVAPI32 |
| 00401069 | | CryptUnprotectData | ADVAPI32 |
| 00401070 | | CryptCreateHash | ADVAPI32 |
| 00401074 | | CryptHashData | ADVAPI32 |
| 00401076 | | CryptDecrypt | ADVAPI32 |
| 00401080 | | CryptCreateHash | ADVAPI32 |
| 00401080 | | CryptDecrypt | ADVAPI32 |
| 00401084 | | CryptDecrypt | ADVAPI32 |
| 00401088 | | RegOpenKeyEx | ADVAPI32 |

IDA Pro imports listing showing cryptographic functions

37

Mã hóa

- ☐ Thuật toán mã hóa đơn giản
- ☐ Thuật toán mã hóa mạnh
- ☐ Thuật toán mã hóa tùy chỉnh

38

Thuật toán mã hóa tùy chỉnh

- ☐ Do người dùng tự phát triển
- ☐ Tùy chỉnh của các thuật toán mã hóa được công bố
- ☐ Ví dụ: Một vòng XOR, sau đó Base64
- ☐ Gây khó khăn cho quá trình phân tích

39

Giải mã

- ☐ Lập trình các hàm giải mã
- ☐ Có thể sử dụng lại những hàm trong chính mã độc

40

Lập trình các hàm giải mã

- ☐ Một số hàm chuẩn sẵn có

```

Sample Python Base64 script
import string
import base64

example_string = 'VghpcyBpcyBhIHJlc3Qgc3RyaW5n'
print base64.decodestring(example_string)

Sample Python NULL-preserving XOR script
def null_preserving_xor(input_char, key_char):
    if (input_char == key_char or input_char == chr(0x00)):
        return input_char
    else:
        return chr(ord(input_char) ^ ord(key_char))

```

41

Lập trình các hàm giải mã

- ☐ Một số hàm chuẩn sẵn có

```

Sample Python custom Base64 script
import string
import base64

s = ""
custom = "9ZABCEFGHIJLKNOPQRSTUWXYZabcdefghijklnopqrstuwxzyB12345678+/"
Base64 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklnopqrstuwxzyB123456789+/"

ciphertext = "TEgobxZobxZgFPkb20="

for ch in ciphertext:
    if (ch in Base64):
        s = s + Base64[ciphertext.find(custom, str(ch))]
    elif (ch == '+'):
        s += '+'
    else:
        s += ' '

result = base64.decodestring(s)

```

42

Lập trình các hàm giải mã

- ❑ Một số thư viện mật mã (PyCrypto,...)

```

Sample Python DES script
from Crypto.Cipher import DES
import sys

obj = DES.new('password', DES.MODE_ECB)
cfile = open('encrypted_file', 'r')
cbuf = f.read()
print obj.decrypt(cbuf)

```

43

Nội dung

1. Nén
2. Mã hóa
3. Làm rối mã
4. Một số cơ chế khác

44

Làm rối mã

- ❑ Mục tiêu: Ngăn chặn việc phân tích, phát hiện dựa vào dấu hiệu (signature) và chống dịch ngược
- ❑ Code được Obfuscation và mutation
- ❑ Phát hiện debuggers và máy ảo nó sẽ dừng và không thực thi hành vi của nó nữa

45

Obfuscation

- ❑ Đặt lại các lệnh, điều kiện bị đảo ngược, tên các thanh ghi khác nhau, trật tự khác nhau,...
- ❑ Các lệnh JUMP và NOP được chèn vào những vị trí ngẫu nhiên
- ❑ Bộ Garbage opcodes được chèn vào các khu vực không thể chèn mã
- ❑ Các lệnh được thay thế bằng những lệnh khác nhưng vẫn đảm bảo giống chức năng

46

Polymorphic Viruses

- ❑ Mỗi bản sao tạo ra một bản mã ngẫu nhiên mới của cùng một virus
- ❑ Mã hóa > giải mã > Mã hóa > ...

47

Metamorphic Viruses

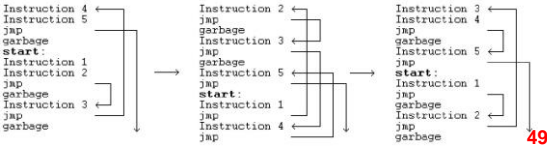
- ❑ Lai tạo và kết hợp nhiều kiểu đa hình khác nhau trong cùng một virus
- ❑ Tự động biến đổi, lai tạp, hình thành các thể hệ virus F1, F2, F3... Fn
- ❑ Các đoạn mã độc được rải rác và nằm ở nhiều nơi

48

Zmist

❑ “Islands” của mã được tích hợp vào các vị trí ngẫu nhiên trong chương trình của máy lưu trữ và liên kết bằng các lệnh nhảy

❑ Virus tự ghép các phần của nó (tích hợp mã)



Nội dung

1. Nén
2. Mã hóa
3. Làm rối mã
4. Một số cơ chế khác

50

Một số cơ chế khác

- ❑ Anti debugging
(PMA chapter 16)
- ❑ Anti VM
(PMA chapter 17)

51

Nội dung

1. Nén
2. Mã hóa
3. Làm rối mã
4. Một số cơ chế khác

52