

LAB 13

Thầy Mai Hoàng Đình
Trường đại học FPT

Người thực hiện

Đặng Hoàng Nguyên

Data Encoding

Trong bài lab này chúng ta sẽ tìm hiểu xem các kỹ thuật mã hóa của malware sẽ như thế nào.

Để thuận tiện trong bài lab này thì chúng ta sẽ sử dụng Windows 10 hoặc một con máy Windows Server 2008 để có thể thực hiện một cách hoàn chỉnh

Beacons

Bây giờ chúng ta sẽ kích hoạt inetsim lên, một môi trường mạng ảo để có thể test malware. Để có thể bật inetsim, chúng ta cần có một máy kali để kích hoạt inetsim và bên máy windows sẽ sử dụng dns của máy kali làm nơi chuyển đích tois

- Sudo inetsim

The screenshot shows a Kali Linux terminal window with the following content:

```

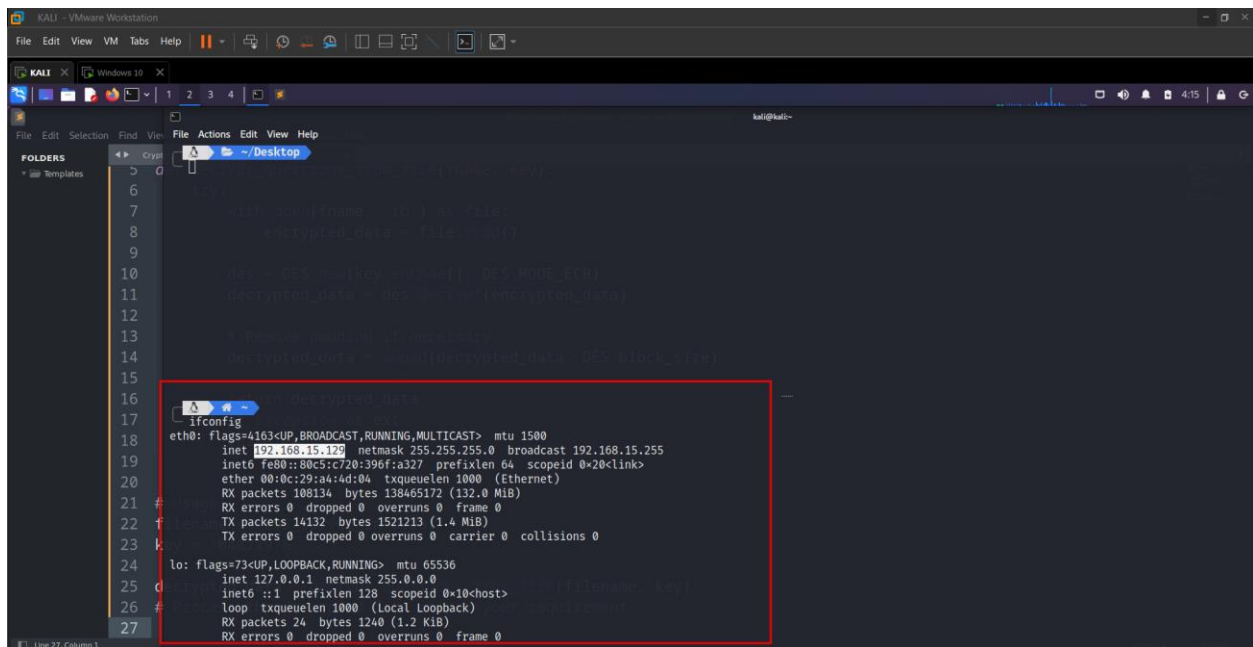
File Edit Selection Find View
FOLDERS
  Templates
1  inetsim
2  Sorry, this program must be started as root!
3  sudo inetsim
4  [sudo] password for kali:
5  INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
6  Using log directory: /var/log/inetsim/
7  Using data directory: /var/lib/inetsim/
8  Using report directory: /var/log/inetsim/report/
9  Using configuration file: /etc/inetsim/inetsim.conf
10 Parsing configuration file.
11 Configuration file parsed successfully.
12 == INetSim main process started (PID 461456) ==
13 Session ID: 461456
14 Listening on: 0.0.0.0
15 Real Date/Time: 2023-06-30 04:07:14
16
17 eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
18   inet 192.168.15.129 netmask 255.255.255.0 broadcast 192.168.15.255
19   inet6 fe80::80c5:720:396f:a327 prefixlen 64 scopeid 0<20<link>
20   ether 00:0c:29:a4:4d:04 txqueuelen 1000 (Ethernet)
21   RX packets 108134 bytes 138465172 (132.0 MiB)
22   RX errors 0 dropped 0 overruns 0 frame 0
23   TX packets 14192 bytes 1521213 (1.4 MiB)
24   TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
25
26 lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
27   inet 127.0.0.1 netmask 255.0.0.0
28   inet6 ::1 prefixlen 128 scopeid 0<10<host>
29   loop txqueuelen 1000 (Local Loopback)
30   RX packets 24 bytes 1240 (1.2 KiB)
31   RX errors 0 dropped 0 overruns 0 frame 0
32   TX packets 24 bytes 1240 (1.2 KiB)
33   TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

A red box highlights the command prompt and the file path `~/Desktop` in the terminal window.

Nếu trong trường hợp mà máy của chúng ta chưa có inetsim thì chúng ta sẽ bắt đầu tiến hành cài như sau:

Bước đầu tiên chúng ta sẽ kiểm tra ip của máy thông qua câu lệnh `ifconfig` hoặc `ip a`

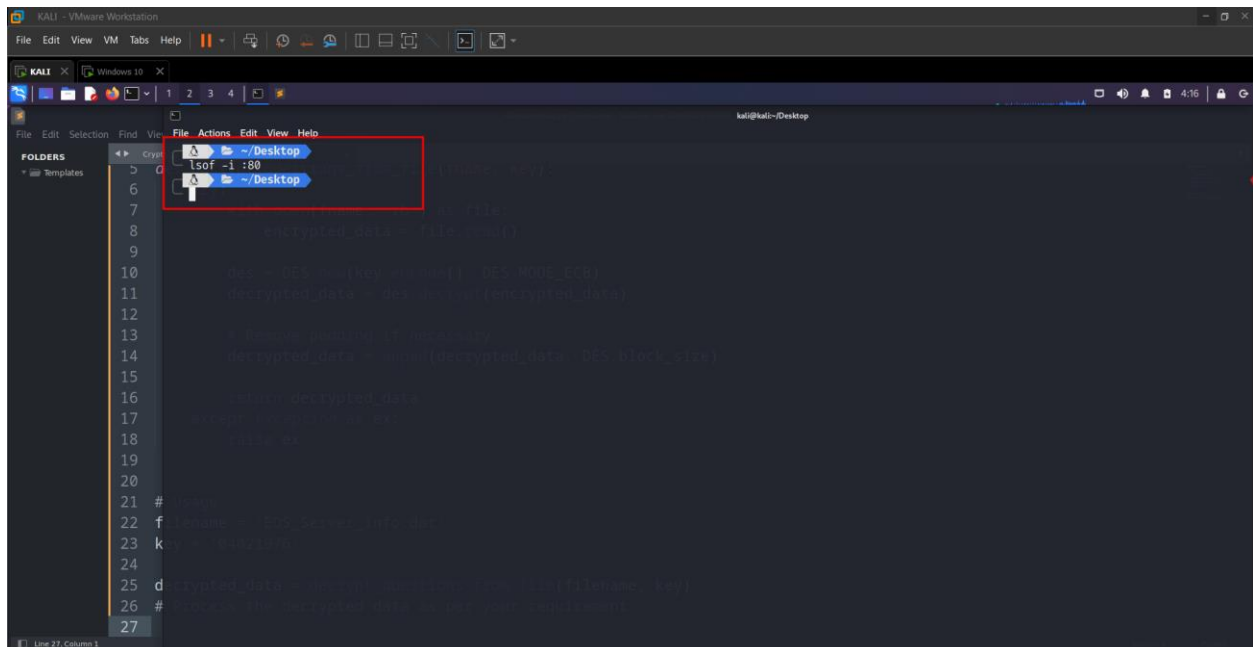


```
kali@kali:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.15.129  netmask 255.255.255.0  broadcast 192.168.15.255
    inet6 fe80::80c5:c720:396f:a327  prefixlen 64  scopeid 0<link>
    ether 00:0c:29:a4:4d:04  txqueuelen 1000  (Ethernet)
    RX packets 108134  bytes 138465172 (132.0 MiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 14132  bytes 1521213 (1.4 MiB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0<host>
    loop txqueuelen 1000  (Local Loopback)
    RX packets 24  bytes 1240 (1.2 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
```

Sau đó sử dụng lệnh "`lsof -i :80`" là một lệnh được sử dụng để liệt kê tất cả các socket TCP/IP đang mở mà đang lắng nghe trên cổng 80, cổng tiêu chuẩn được sử dụng cho lưu lượng web HTTP. Lệnh này sẽ hiển thị thông tin về các quy trình hoặc ứng dụng đang sử dụng cổng 80, bao gồm cả ID quá trình (PID) và tên của ứng dụng hoặc quá trình đó. Điều này có thể hữu ích trong việc xử lý sự cố liên quan đến máy chủ web và kết nối mạng.

Trong trường hợp này: chúng ta không có bất kỳ điều gì, vì vậy lệnh "`lsof -i :80`" sẽ không hiển thị bất kỳ thông tin gì.



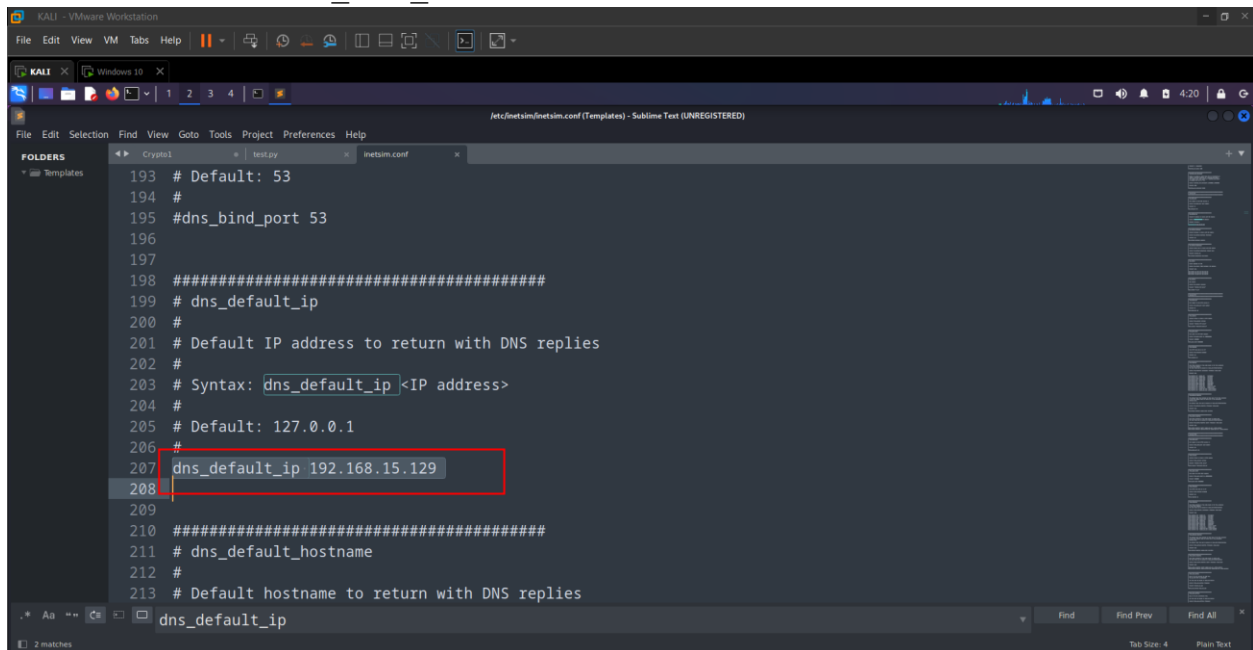
```
kali@kali:~/Desktop$ lsof -i :80
```

Sau đó chúng ta sẽ bắt đầu cấu hình inetsim bằng cách cấu hình lại file inetsim. Sử dụng câu lệnh sau:

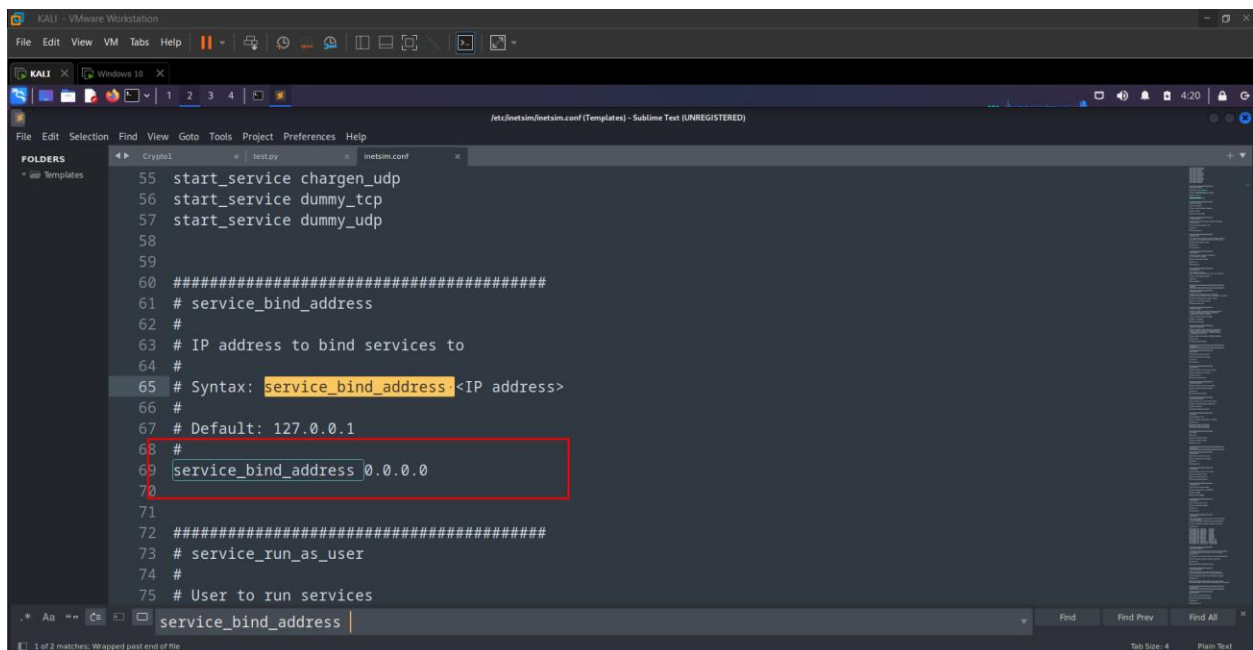
- Nano /etc/inetsim/inetsim.conf

Và chỉnh những thông số sau:

- Chỉnh dns_default_ip về ip chính của máy kali
- Đổi luôn service_bind_address về 0.0.0.0

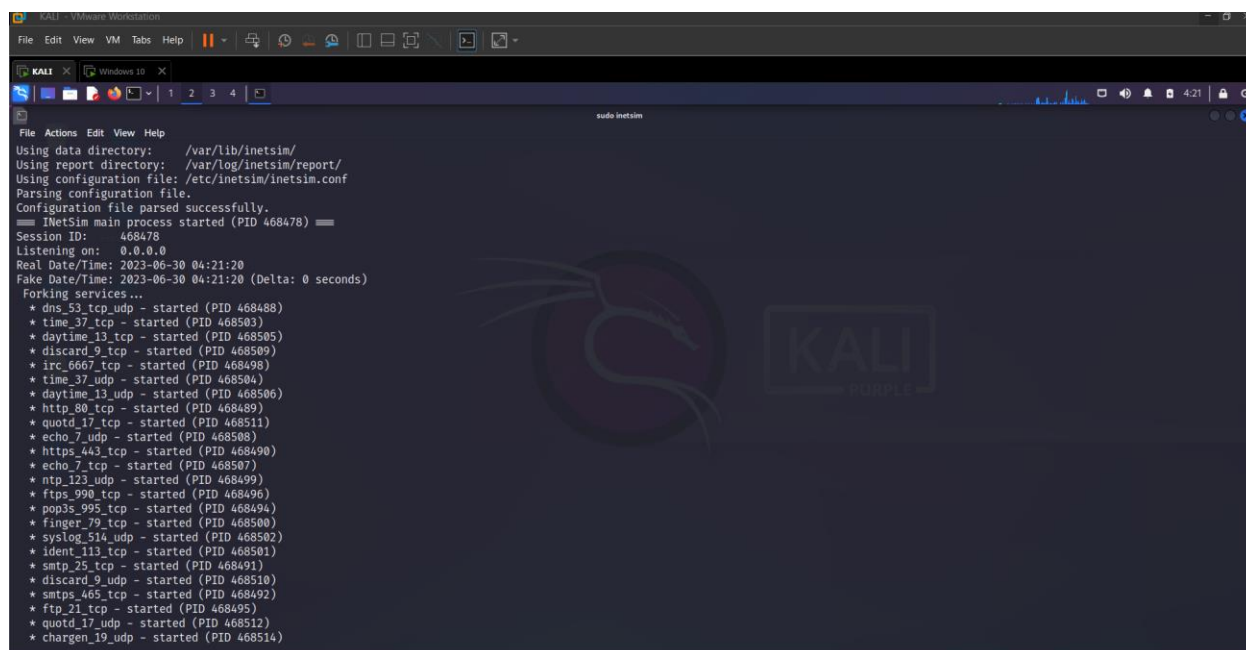


```
193 # Default: 53
194 #
195 #dns_bind_port 53
196
197
198 #####
199 # dns_default_ip
200 #
201 # Default IP address to return with DNS replies
202 #
203 # Syntax: dns_default_ip <IP address>
204 #
205 # Default: 127.0.0.1
206 #
207 dns_default_ip 192.168.15.129
208
209
210 #####
211 # dns_default_hostname
212 #
213 # Default hostname to return with DNS replies
214 dns_default_ip
```



```
55 start_service chargen_udp
56 start_service dummy_tcp
57 start_service dummy_udp
58
59
60 #####
61 # service_bind_address
62 #
63 # IP address to bind services to
64 #
65 # Syntax: service_bind_address <IP address>
66 #
67 # Default: 127.0.0.1
68 #
69 service_bind_address 0.0.0.0
70
71
72 #####
73 # service_run_as_user
74 #
75 # User to run services
76 service_bind_address
```

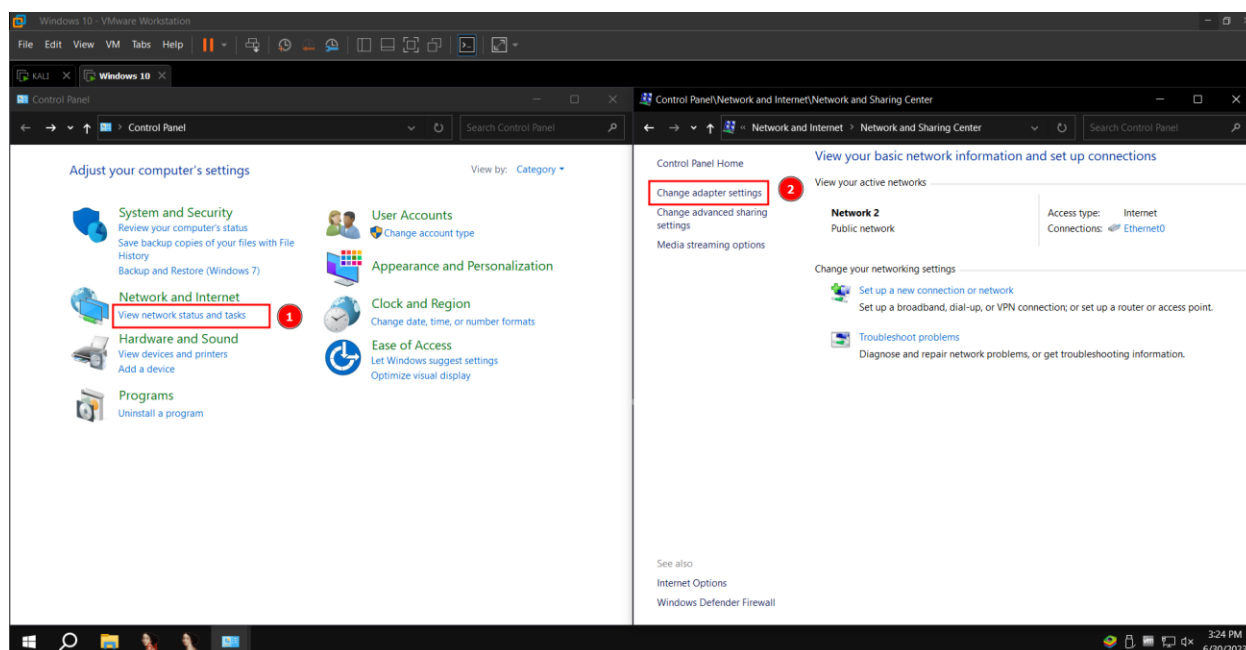
Sau đó lưu lại và kích hoạt inetsim lên. Như ta thấy bên hình dưới thì chúng ta đã kích hoạt thành công inetsim trên máy kali



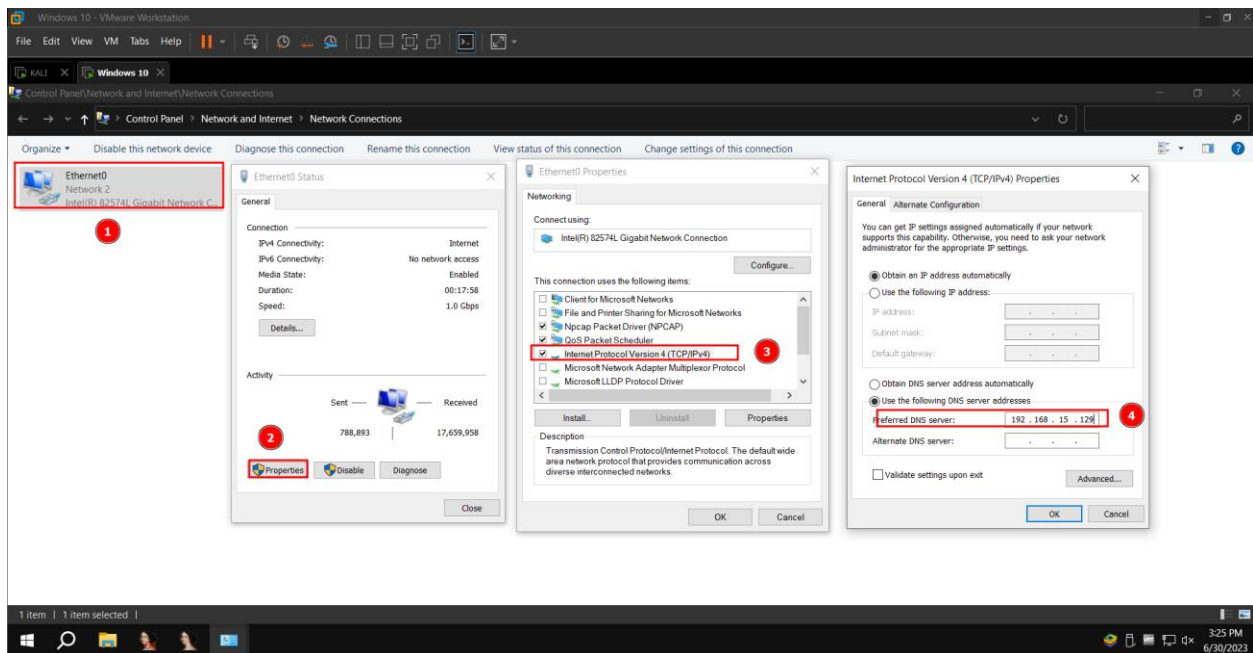
```
File Actions Edit View Help
Using data directory: /var/lib/inetsim/
Using report directory: /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
== Inetsim main process started (PID 468478) ==
Session ID: 468478
Listening on: 0.0.0.0
Real Date/Time: 2023-06-30 04:21:20
Fake Date/Time: 2023-06-30 04:21:20 (Delta: 0 seconds)
Forking services...
* dns_53_tcp_udp - started (PID 468488)
* time_37_tcp - started (PID 468503)
* daytime_13_tcp - started (PID 468505)
* discard_9_tcp - started (PID 468509)
* irc_6667_tcp - started (PID 468498)
* time_37_udp - started (PID 468504)
* daytime_13_udp - started (PID 468506)
* http_80_tcp - started (PID 468489)
* quotd_17_tcp - started (PID 468511)
* echo_7_udp - started (PID 468508)
* https_443_tcp - started (PID 468490)
* echo_7_tcp - started (PID 468507)
* ntp_123_udp - started (PID 468499)
* ftps_990_tcp - started (PID 468496)
* pop3s_995_tcp - started (PID 468494)
* finger_79_tcp - started (PID 468500)
* syslog_514_udp - started (PID 468502)
* ident_113_tcp - started (PID 468501)
* smtp_25_tcp - started (PID 468491)
* discard_9_udp - started (PID 468510)
* smtps_465_tcp - started (PID 468492)
* ftp_21_tcp - started (PID 468495)
* quotd_17_udp - started (PID 468512)
* chargen_19_udp - started (PID 468514)
```

Bây giờ chúng ta sẽ thay đổi địa chỉ DNS trên máy Windows trở về máy Kali để có thể kích hoạt inetsim

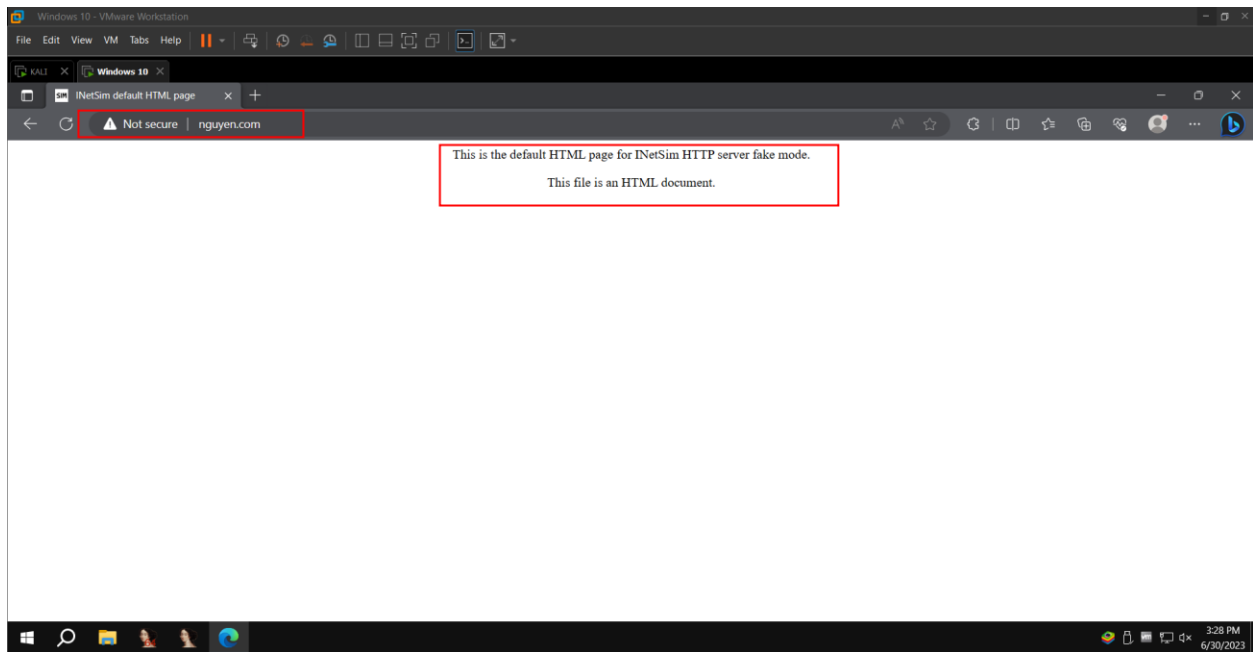
Vào bên trong Control Panel, tại **Network and Internet** chúng ta sẽ chọn **View network and tasks**. Sau đó bên mục bên tay trái sẽ nhấn vào **Change adapter settings**.



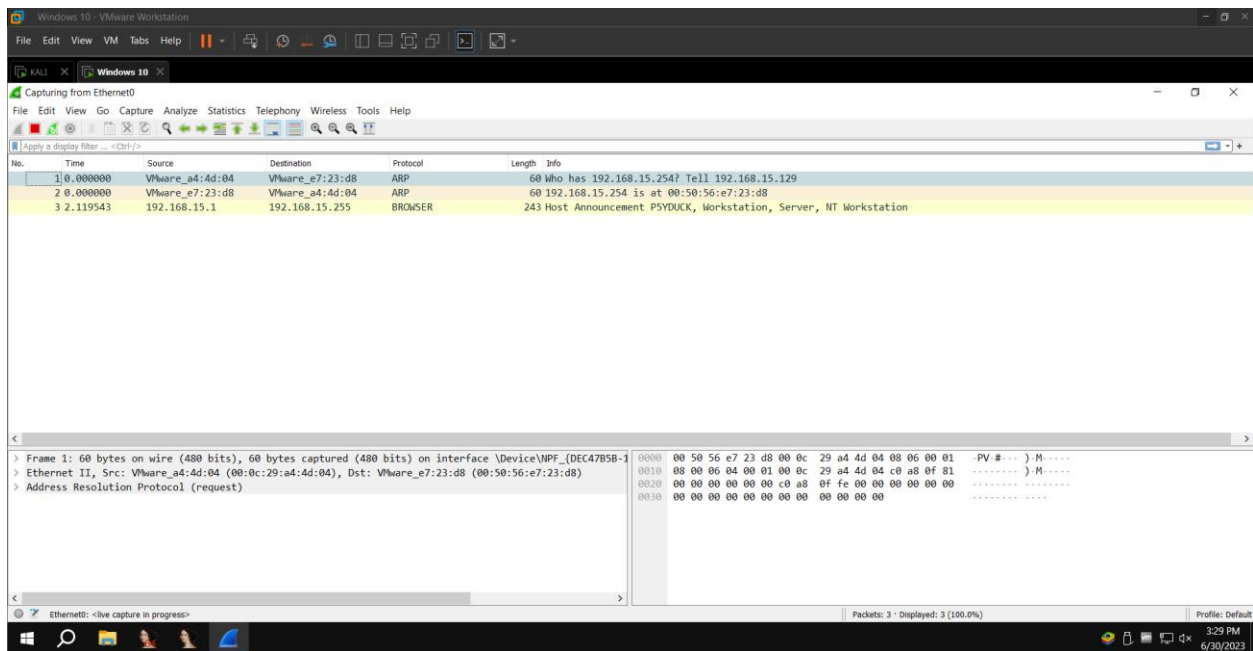
Sau đó chúng ta sẽ chọn vào NIC chúng ta đang sử dụng, sau đó chọn **Properties** → **IPv4**. Trong IPv4, chúng ta sẽ dùng địa chỉ DNS của máy Kali để test.



Sau khi setup xong, chúng ta thử vào trình duyệt web và đánh thử một tên. Trong trường hợp này sẽ đánh `nguyen.com` và ra kết quả như hình bên dưới

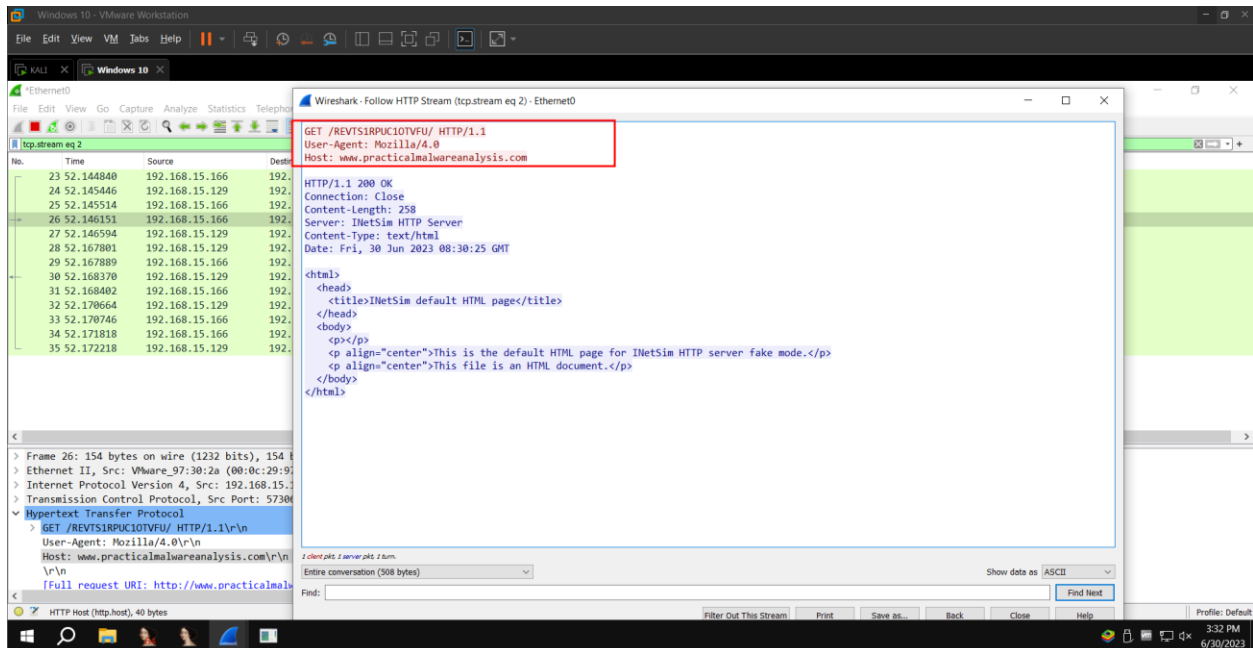


Sau đó chúng ta sẽ kích hoạt Wireshark lên để bắt gói tin mà chúng gửi ra ngoài trước khi kích hoạt con malware



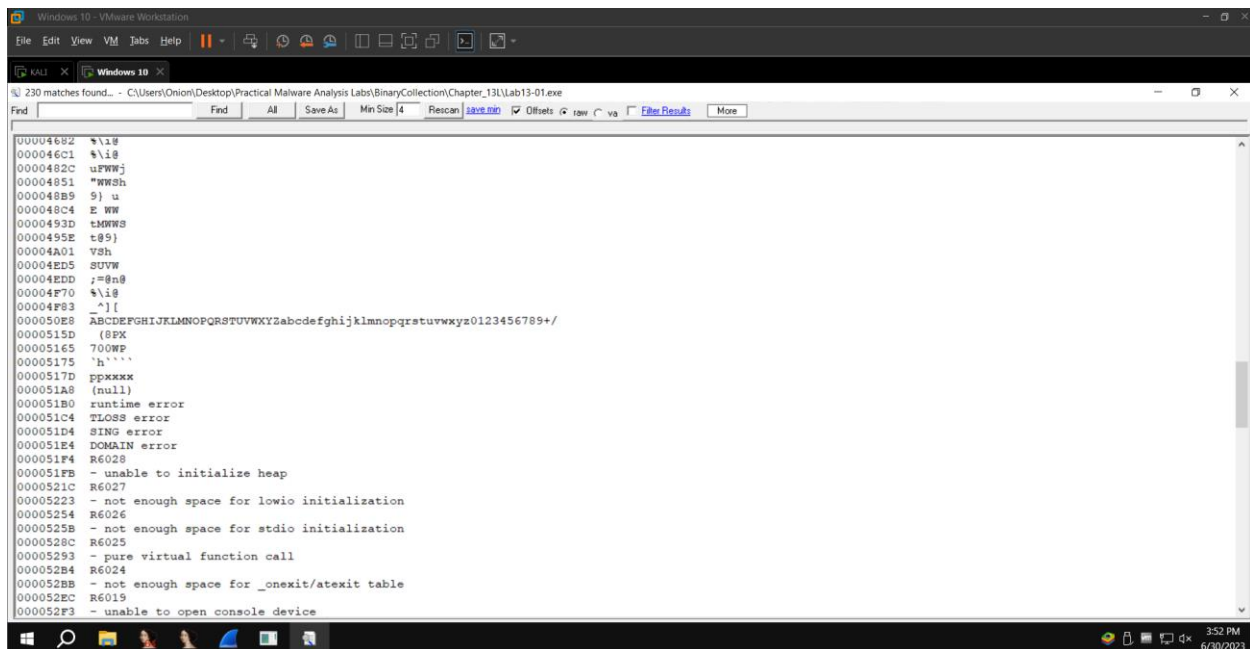
Sau đó chúng ta kích hoạt con malware **Lab13-01.exe** lên và xem chúng làm những gì

Sau khi chạy, chúng ta sẽ filter các gói PDU bằng filter http. Chúng ta có thể dễ dàng thấy được rằng các gói tin từ host là www.practicalmalwareanalysis.com và thấy được kết nối thành công với mã 200 tới inetsim chính là môi trường ảo của chúng ta



Strings

Bây giờ chúng ta sẽ xem strings, phân tích tĩnh của phần mềm xem bên trong nó có những gì. Chúng ta thấy rằng nó sử dụng các ký tự bình thường, và đồng thời nó cũng sử dụng các vùng nhớ để thực hiện một cái gì đó.

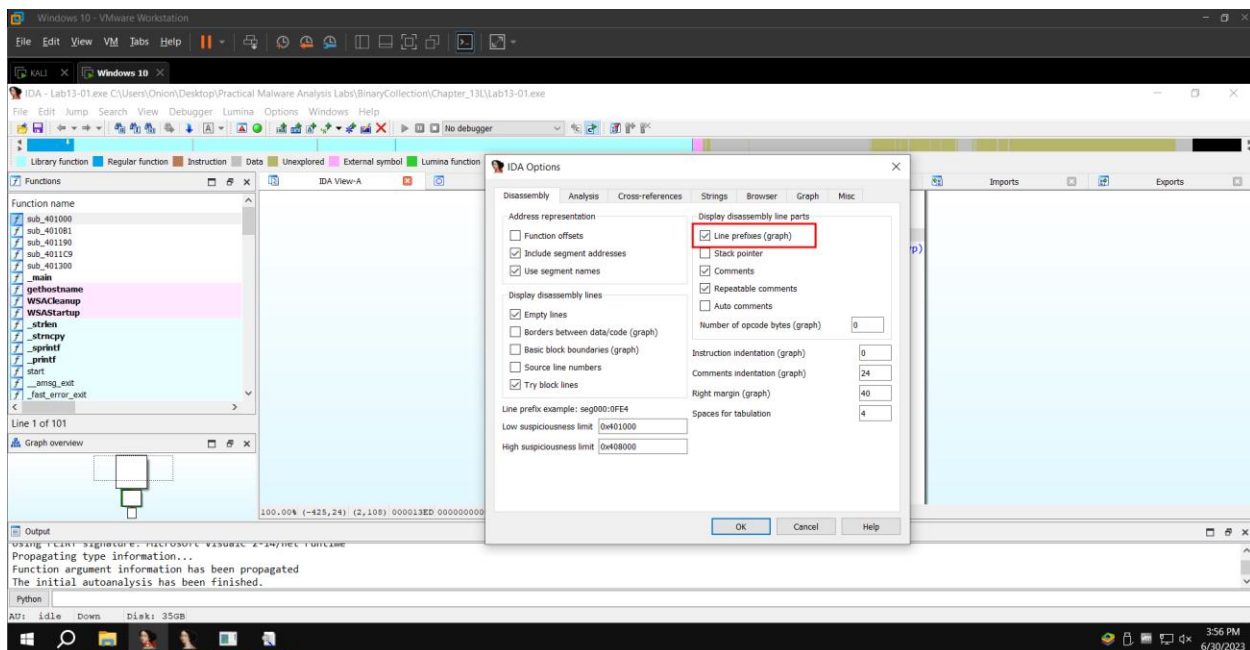


IDA Pro

Sau khi xem string, chúng ta sẽ tiến hành load vào bên trong IDA để phân tích động

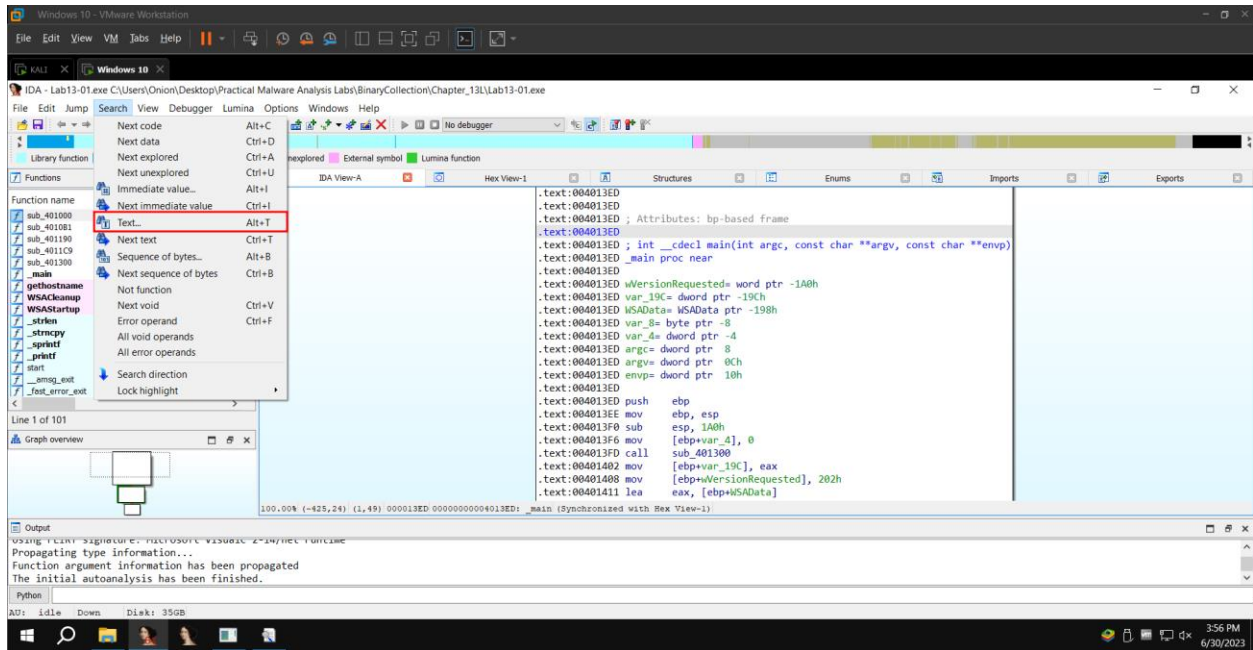
Mở tệp Open Lab13-01.exe trong IDA Pro.

Nhấp vào Options, General. Đánh dấu "Line Prefixes" và nhấp OK.

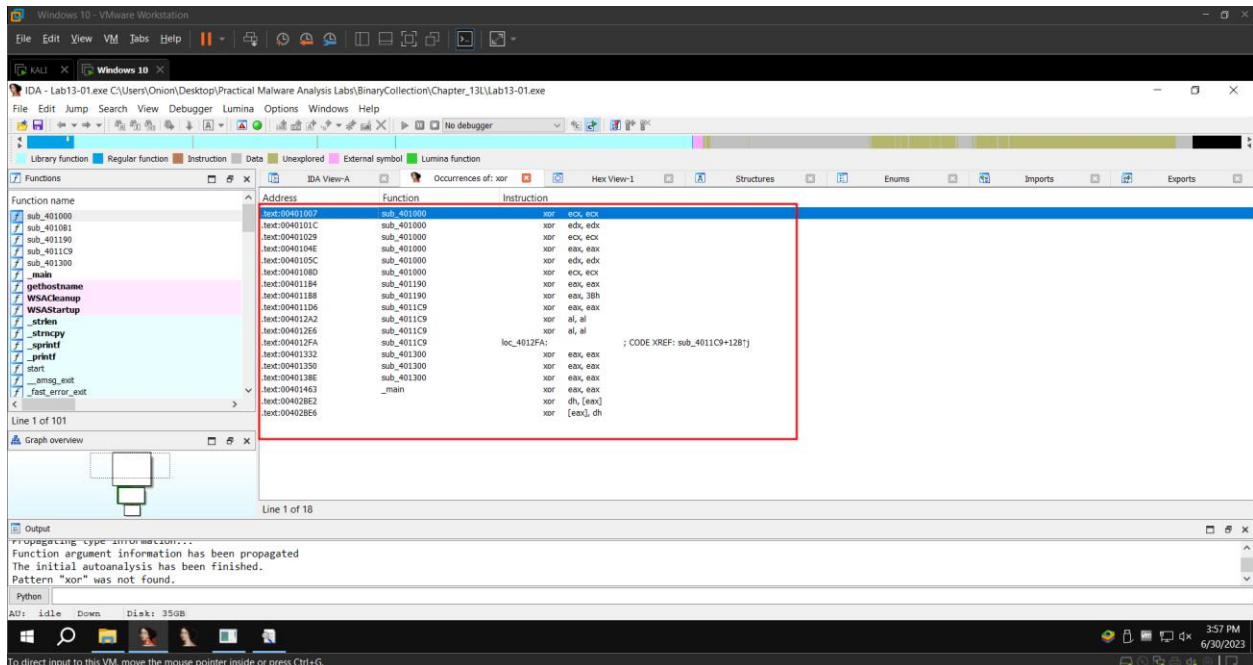


Nhấp vào cửa sổ "IDA View-A"

Từ thanh menu, nhấp vào Search, text....



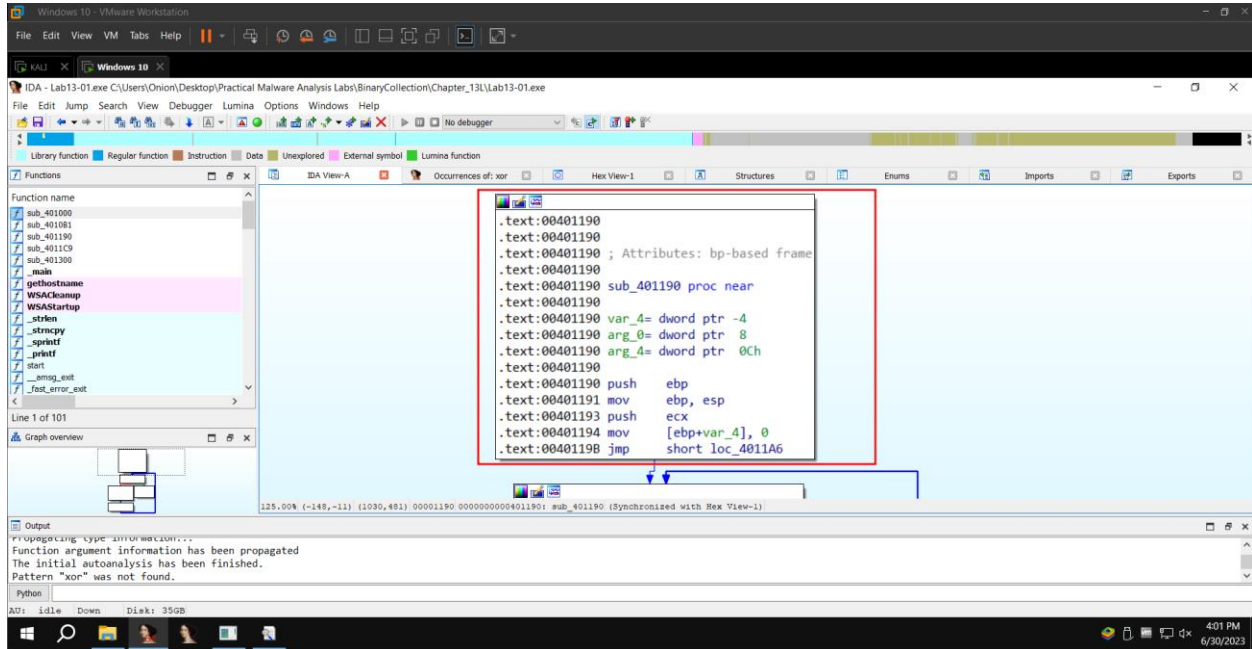
Trong hộp thoại Text Search, nhập xor và đánh dấu "Find all occurrences", như được hiển thị bên dưới.



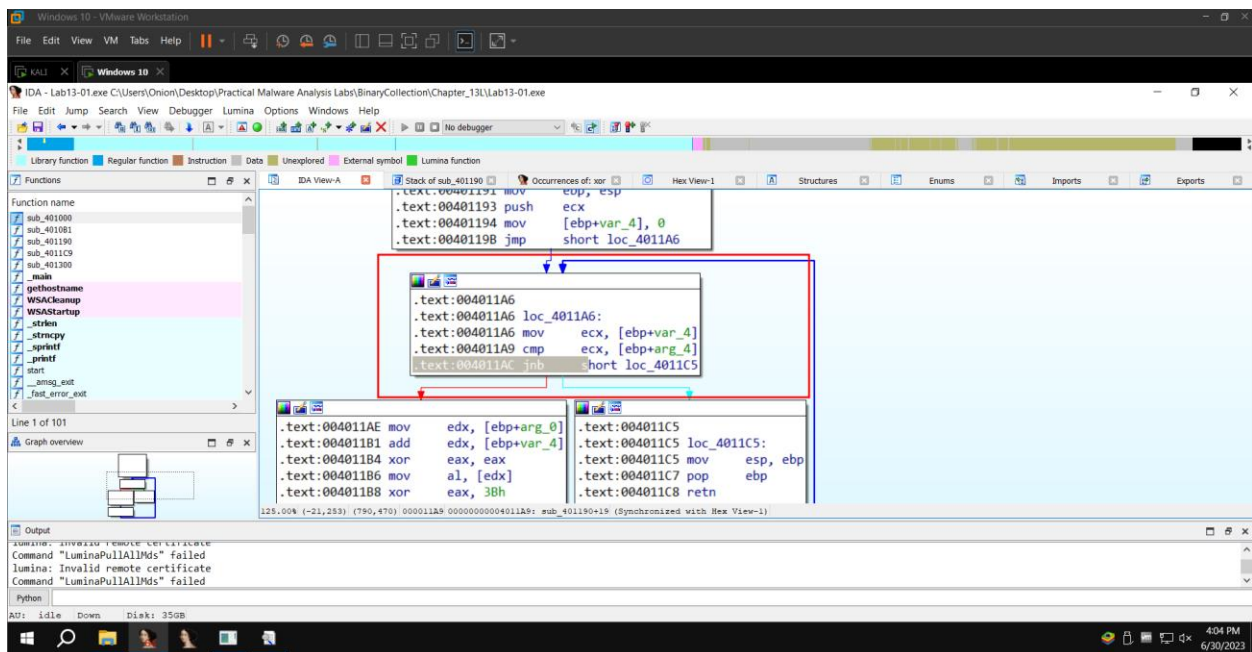
Ta chú ý thấy hàm xor ở vị trí hàm được gọi lsub_401191 khá là khác so với các hàm xor khác bởi vì các hàm xor kia đều trả về giá trị là chính nó nhưng chỉ có hàm xor kia khá là kì bởi vì nó dùng eax để xor với 3B trong hệ 16

Nhấn vào bên trong đó để đến hàm sub_401191 để xem trong đó chứa gì và ta sẽ thấy được cách nó thực thi như hình dưới đây:

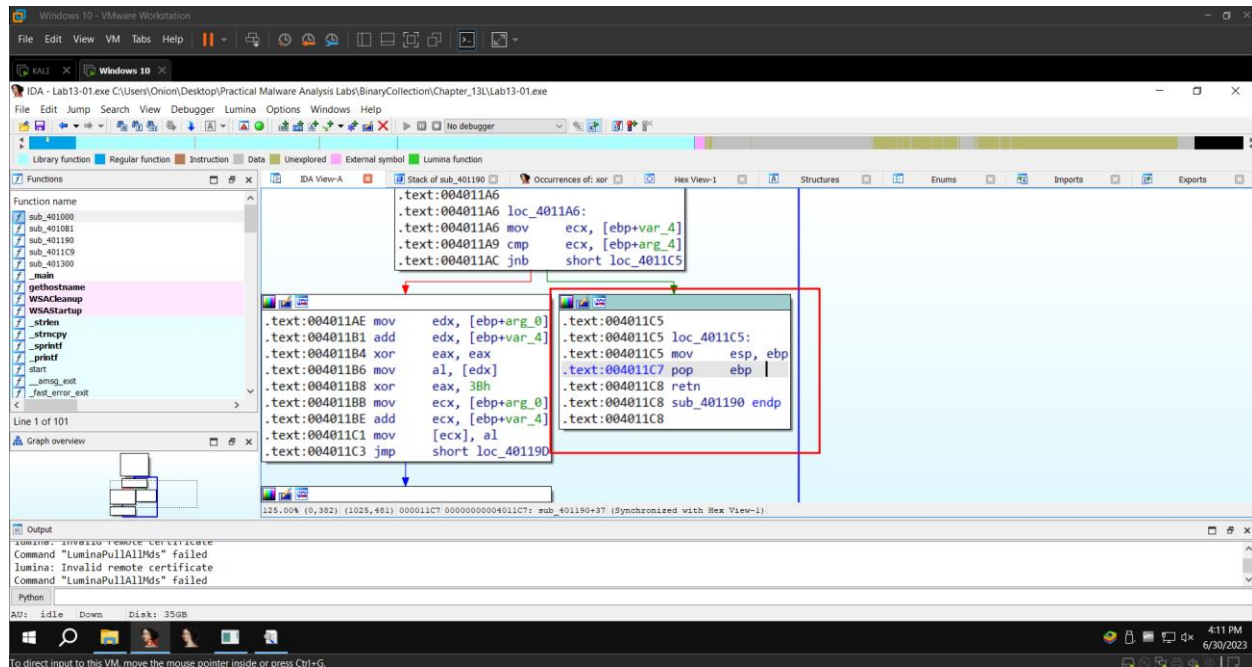
Tại block đầu tiên sẽ khai báo biến và tạo ra một stack và gán giá trị 0 cho vùng nhớ base pointer -4 (ở phía trên base pointer). Và sau đó sẽ nhảy xuống loc_4011A6



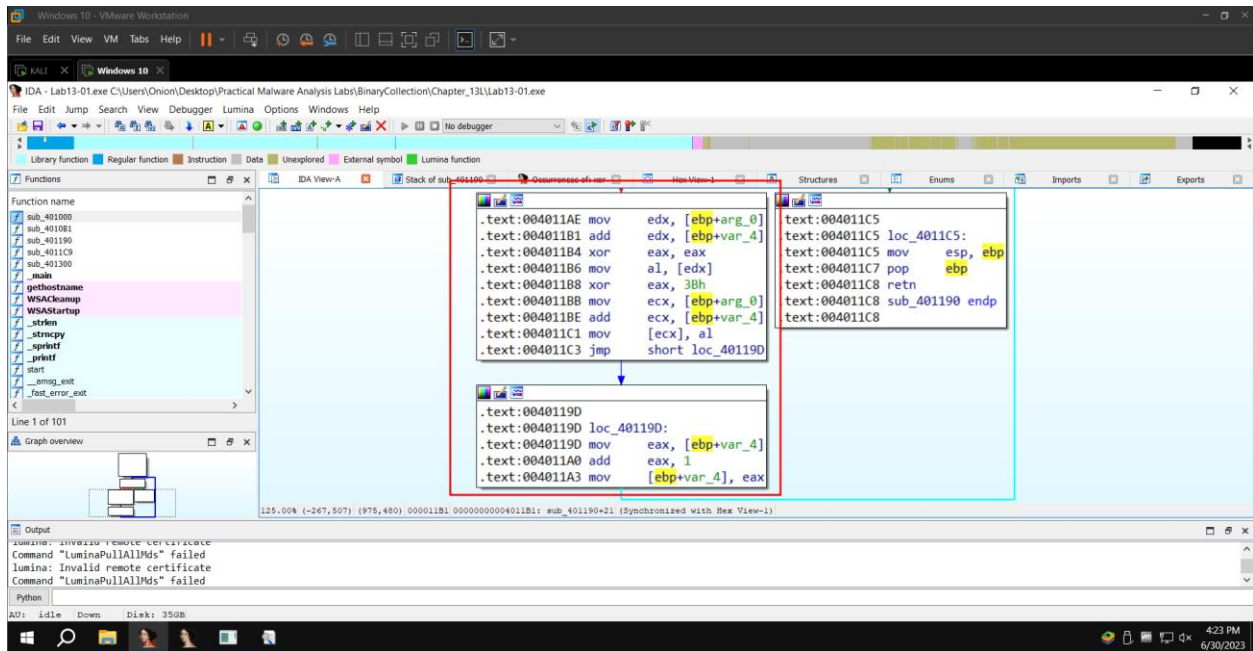
Sau đó nó sẽ thực hiện dịch chuyển so sánh giữa ecx và trên base pointer + 0C mã hex. Nó tiếp theo sẽ thực hiện câu lệnh jump if not below. Nếu nó bé hơn hoặc bằng bằng ebp + 0c thì nó sẽ chuyển tới loc_4011C5 block bên phải còn không thì nó sẽ tiếp tục thực hiện block bên trái



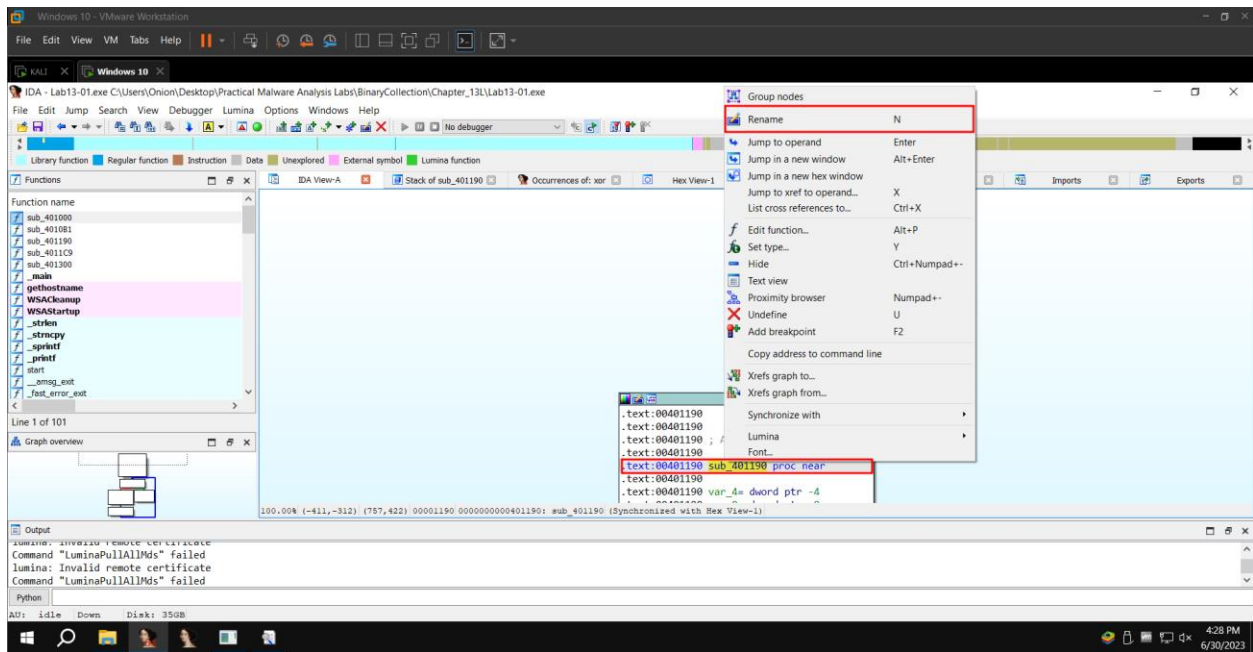
Tại phần nếu đúng thì nhảy, câu lệnh theo em hiểu sẽ gán cho base pointer = stack pointer và sau đó pop ra bên ngoài làm cho stack biến mất và kết thúc câu lệnh



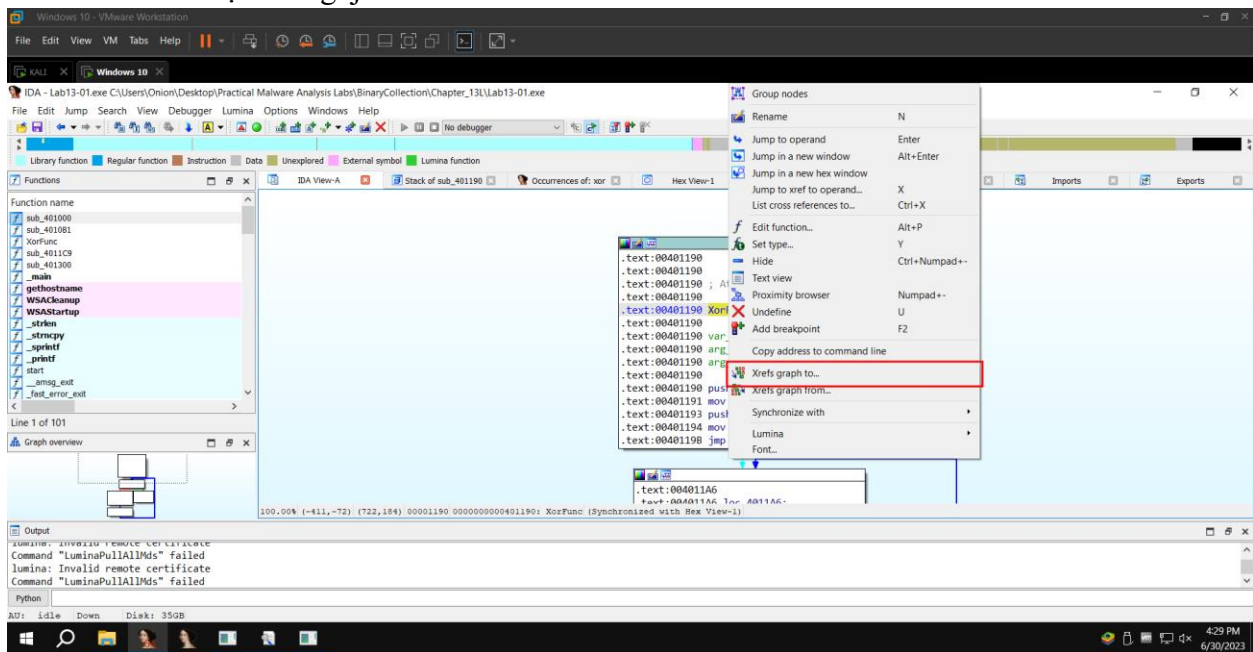
Còn bên chiều hướng ngược lại, chúng sẽ chuyển `ebp+arg_0` vào bên trong thanh ghi `edx`, sau đó cộng thêm với `ebp+var_4`, thực hiện phép toán xor đơn giản cho `eax` và `eax = 0`, sau đó chuyển thanh ghi `edx` vào giá trị `al` vừa `eax`, sau bước đó chúng sẽ thực hiện phép toán xor `eax` với 3B hệ 9 sau đó chúng sẽ chuyển `ebp+arg_0` vào bên trong thanh ghi `edx`, sau đó cộng thêm với `ebp+var_4`, sau đó rồi chuyển `al` vào bên trong `ecx` và nhả xuống `loc_400119D`, chuyển giá trị vào `eax` và sau đó add thêm 1 vào bên trong `eax` và chuyển giá trị đó lên trên stack tại địa chỉ `ebp+var_4`. Sau đó vòng lặp được thực hiện liên tục cho tới khi nào mà thực hiện được `jnb` thì thôi.



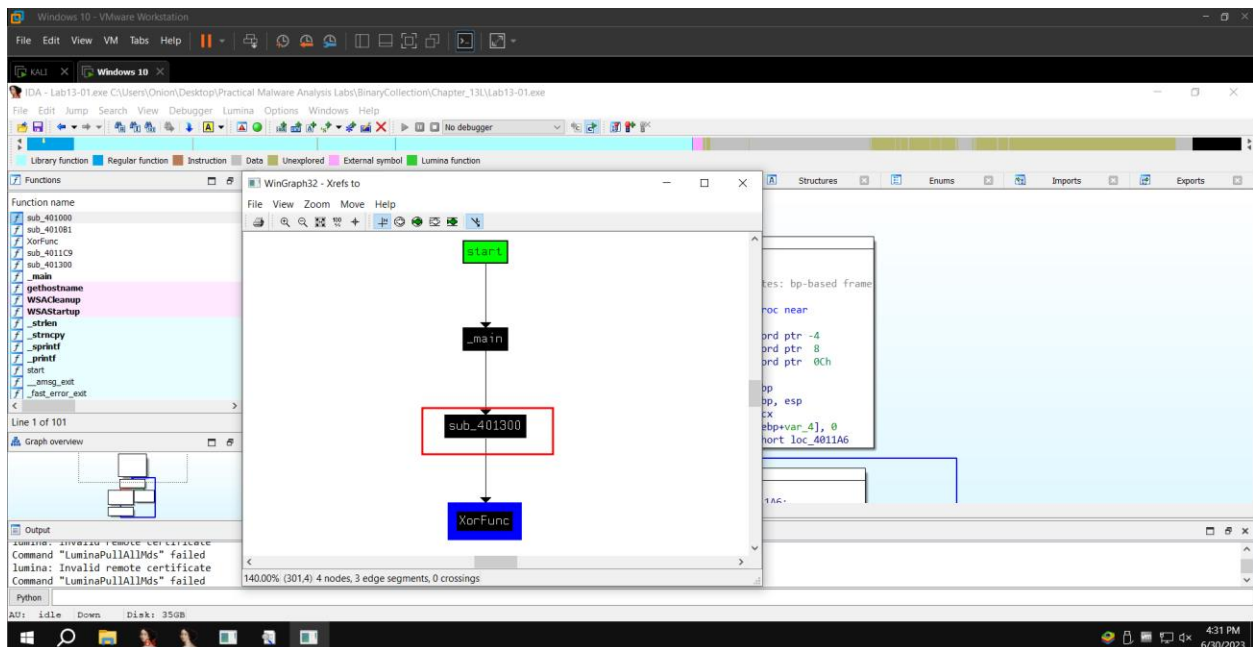
Đây theo như lý thuyết là đang thực hiện hàm xor, vì thế ta sẽ đổi tên hàm lại là XOR. Chuột phải vào sub_401190 và đổi tên thành XORFunction.



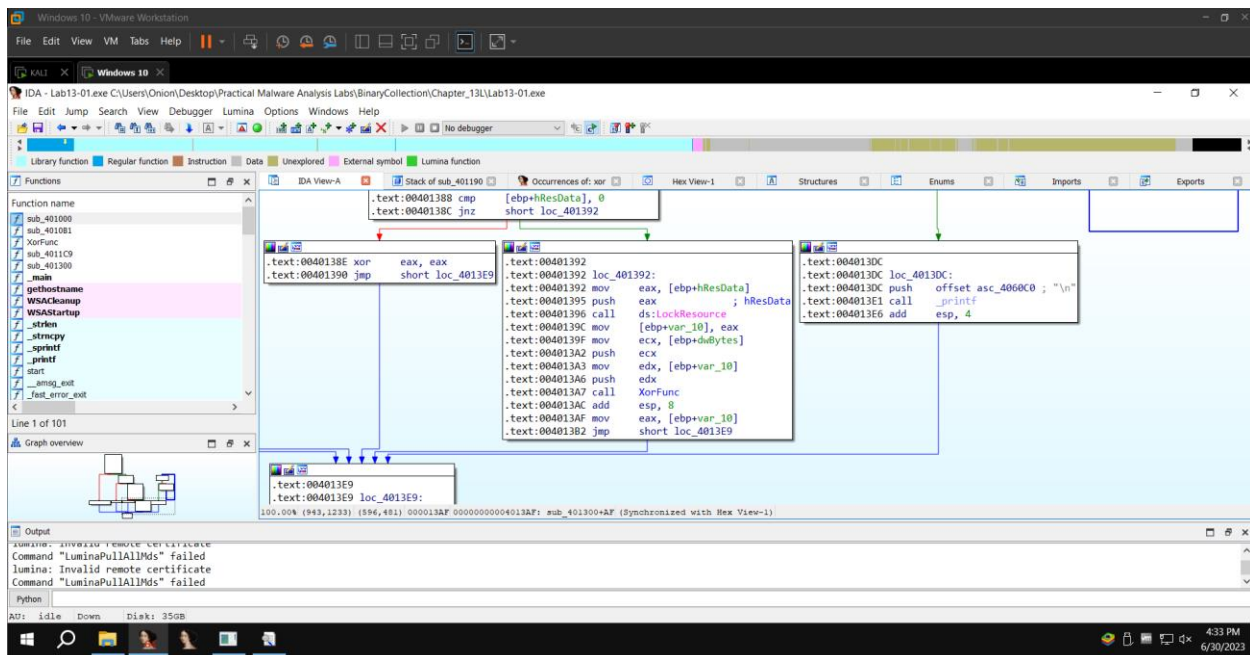
Sau khi xong, chúng ta sẽ nhấn chuột phải và chọn tiếp vào xrefs graph to để tạo ra một cái graph để có thể xem hoạt ddoognj của nó



Thì ta thấy rằng trước khi gọi tới hàm XOR này thì nó đã gọi một hàm khác đó chính là hàm sub_4013000 như hình bên dưới



Vì đoạn code khá dài nên chúng ta chỉ có thể nói nôm na là nó sẽ lấy giá trị và bắt đầu tiến hành encode nó bằng cách push vào bên trong stack và thực hiện quá trình xor.

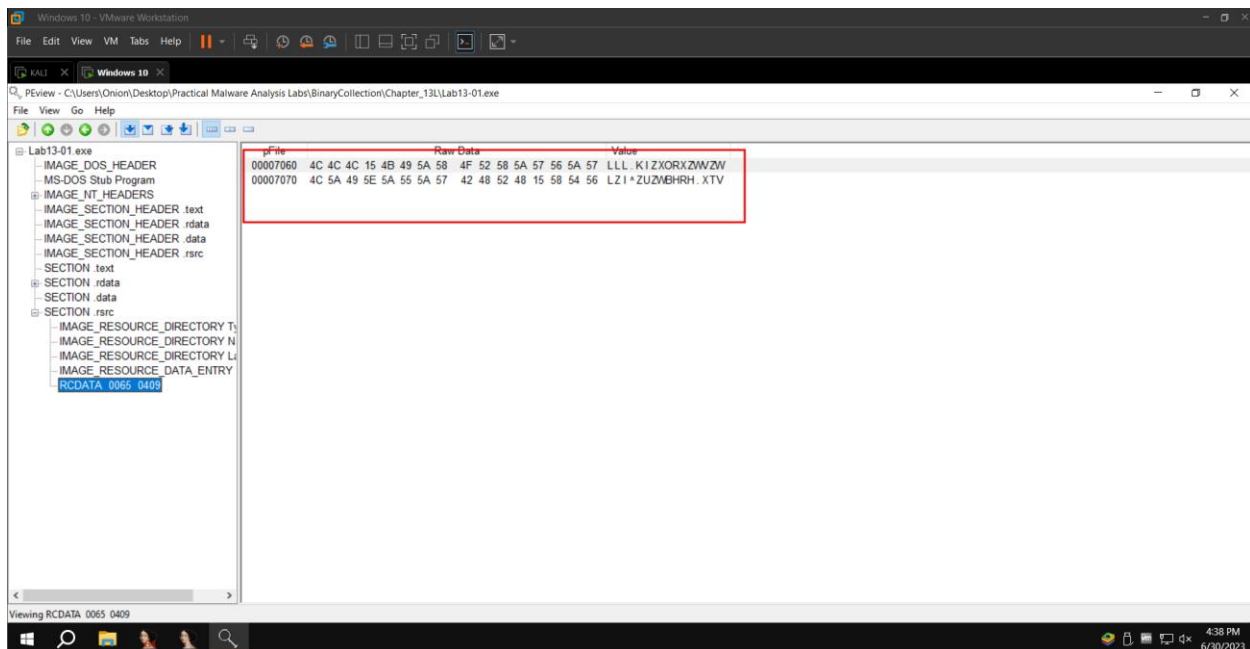


PEview

Chúng ta tiến hành mở chương trình trong PEview để xem.

Trên tay trái, nhấp vào nguồn tài nguyên RCDATA 0065 0409.

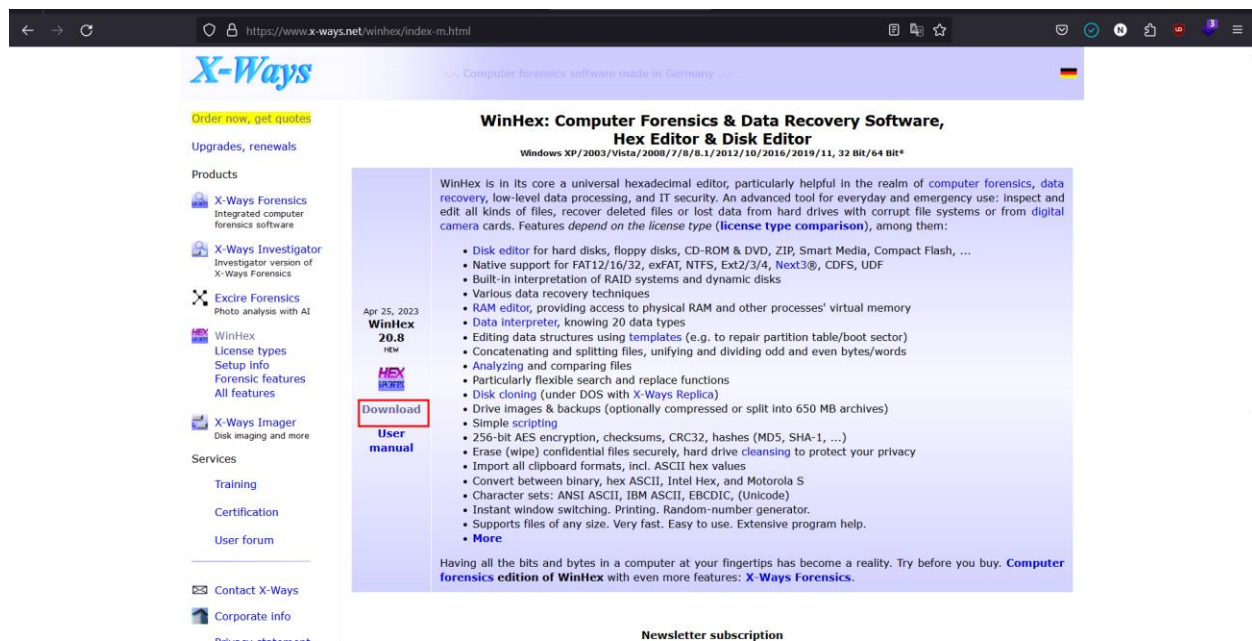
Trên tay phải, tìm địa chỉ bắt đầu 00007060, như được hiển thị dưới đây. Đây là đoạn string đã được encode lên và chúng ta phải tìm cách để có thể giải mã nó



WinHex

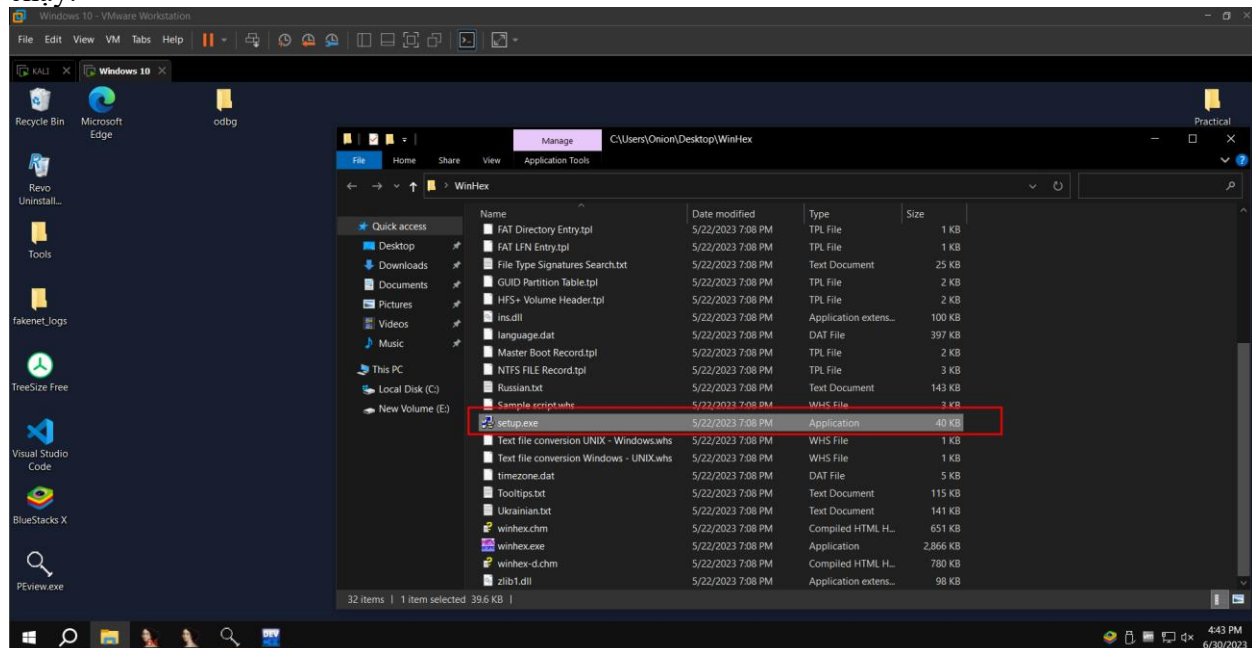
"Trong trình duyệt Web, hãy truy cập vào: <http://winhex.com/winhex/>

Ở phía trái, nhấp vào nút Tải xuống (Download), như được hiển thị dưới đây.

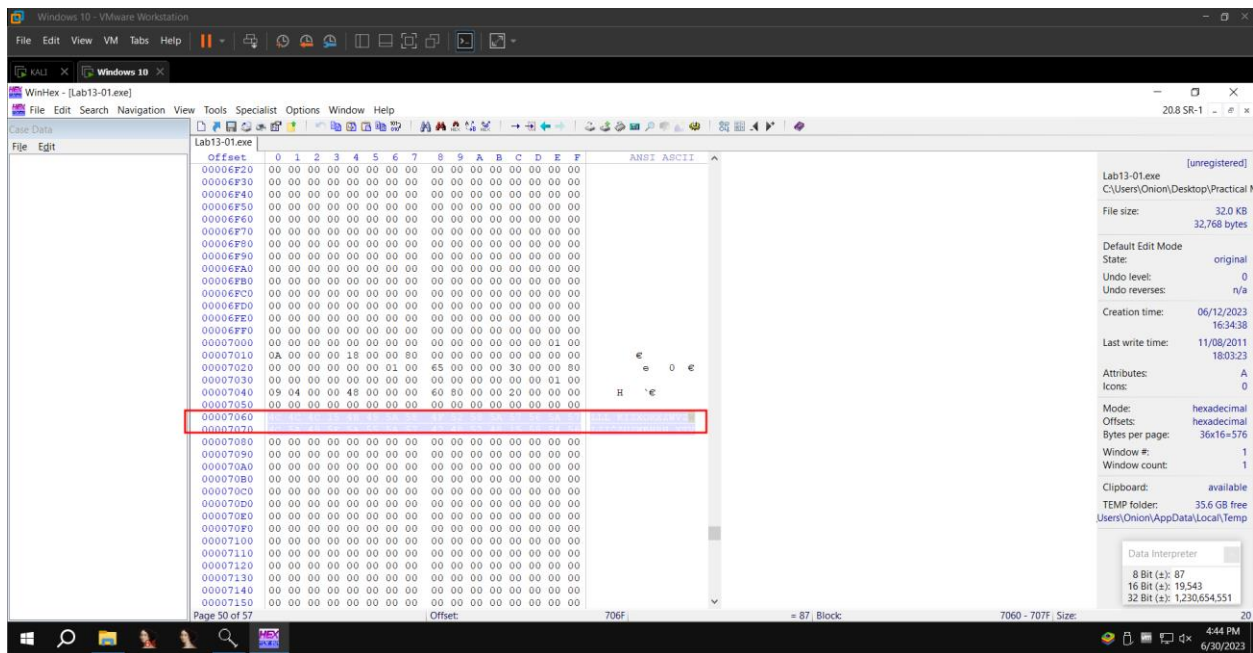


Nhấp chuột phải vào tệp winhex.zip, chọn "Giải nén tất cả", và nhấp Giải nén.

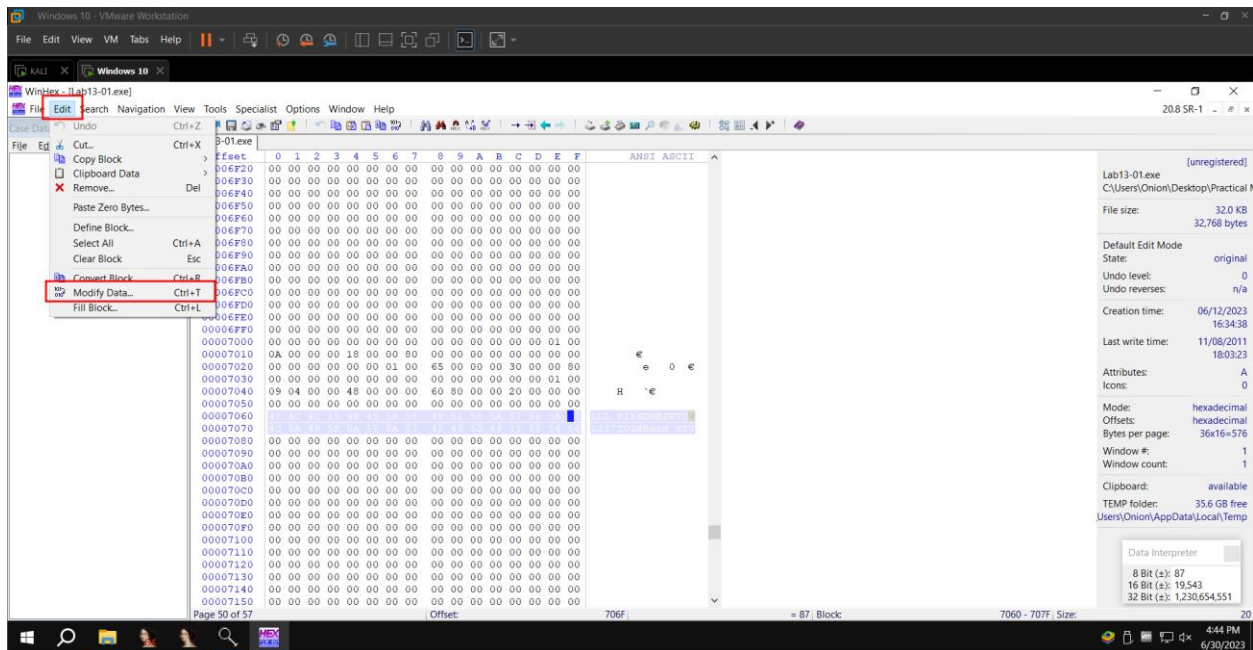
Một cửa sổ hiển thị các tệp tin nằm trong lưu trữ winhex xuất hiện. Nhấp đúp vào file setup.exe. Chấp nhận các tùy chọn mặc định để cài đặt WinHex. Khi quá trình cài đặt hoàn tất, WinHex sẽ chạy.

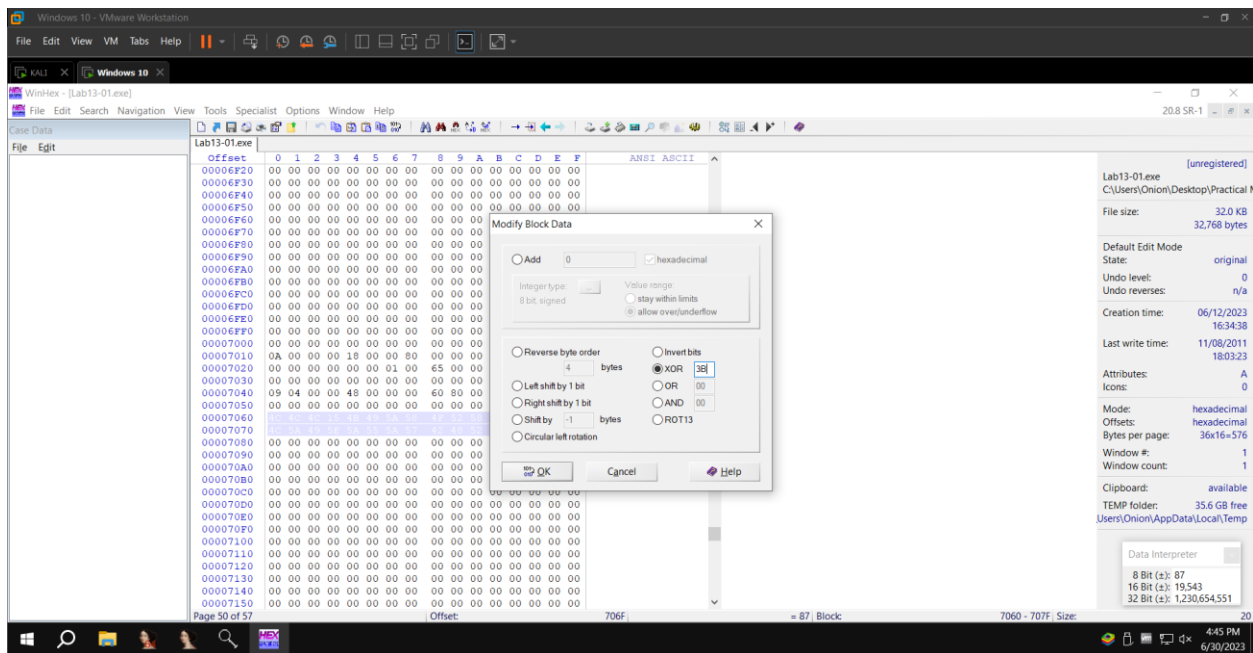


Trong WinHex, nhấp vào File, Open. Mở file Lab13-01.exe trong WinHex. Tô sáng các byte từ 7060 đến 707F, như được hiển thị dưới đây.



Nhấp vào Edit, "Modify Data". Trong hộp "Modify Block Data", chọn nút radio XOR và nhập khóa là 3B, tại vì nó được xor với 3B trong hex:





Và ta ra được đoạn mã ban đầu đó chính là www.practicalmalwareanalysis.com

