**PRACTICAL MALWARE ANALYSIS**

***Lab 1-1***

1. **File Lab01-01.dll**

A screenshot of a computer

Description automatically generated

**Match 44/72 antivirus signatures**

**File Lab01-01.exe**

A screenshot of a computer

Description automatically generated

**Match 57/72 antivirus signatures**

1. **File Lab01-01.dll  
   A screenshot of a computer

   Description automatically generated**

Thời gian compiled

**Mở PEview, load file, chọn vào mục IMAGE\_FILE\_HEADER, sang mục Time Date Stamp, giá trị đó chính là thời điểm file được compile(vd như trong hình là: 2010/12/19 Sun 16:16:38 UTC)**

**File Lab01-01.exe**

**A screenshot of a computer

Description automatically generated**

**Của Lab01-01.exe cách làm tương tự, thời gian compile như ở trên hình.**

1. **Kiểm tra trong PEbear**

**A screenshot of a computer

Description automatically generated**

**Lab01-01.dll**

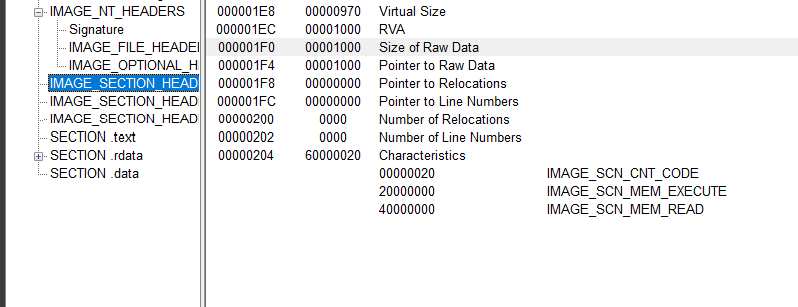
**A screenshot of a computer

Description automatically generated**

**Lab01-01.exe**

**Số lượng hàm Import không quá ít, nếu bị packed có thể sẽ không thấy các hàm được Import hoặc có rất ít hàm được Import.**

**Kiểm tra SizeOfRawData và VirtualSize trong IMAGE\_SECTION\_HEADER:**

****

**Lab01-01.exe**

**A screenshot of a computer

Description automatically generated**

**Lab01-01.dll**

**Kiểm tra thì thấy VirtualSize < SizeOfRawData và 2 trường không lệch nhau quá nhiều**

**Nên từ đó có thể đánh giá 2 file này không bị packed.**

1. **File Lab01-01.exe**

**A screenshot of a computer

Description automatically generated**

**Load vào Dependency Walker, ta thấy có các hàm FindFirstFileA, FindNextFileA, FindClose để duyệt các file trong thư mục, CreateFileMappingA, MapViewOfFile, UnMapViewOfFile để tạo và ánh xạ file vào bộ nhớ, CreateFileA để mở hoặc tạo 1 file, CopyFileA để sao chép 1 file vào 1 file mới.**

**Dựa vào đó ta có thể đoán được rằng file Lab01-01.exe sẽ duyệt các file trong 1 thư mục và tạo bản sao của các file đó.**

**File Lab01-01.dll**

**A screenshot of a computer

Description automatically generated**

**CreateProcessA để tạo và chạy 1 tiến trình mới, CreateMutexA, OpenMutexA để đảm bảo tại 1 thời điểm chỉ có 1 tiến trình được chạy, Sleep để tạm dừng thực thi chương trình trong 1 khoảng thời gian nhất định.**

**Dựa trên những hàm vừa phân tích, file Lab01-01.dll sẽ tạo và chạy các tiến trình và có thể tạm dừng tiến trình đang thực thi trong 1 khoảng thời gian.**

**File Lab01-01.dll còn có 1 dll khác là WS2\_32.dll**

**A screenshot of a computer

Description automatically generated**

**Ta thấy có sử dụng 1 số hàm liên quan đến socket, như: connect để thiết lập kết nối giữa client và server, accept để chấp nhận kết nối, getsockopt để lấy thông tin cấu hình socket,getpeername để lấy địa chỉ IP và port của bên còn lại, bind để gắn socket vào địa chỉ IP/port cụ thể, htons/htonl để chuyển đổi byte order giữa hệ thống host và network, closesocket để đóng socket.**

**Dựa trên đó ta có thể đoán file trên sẽ có thể tạo socket để thiết lập kết nối C2, gửi và nhận packet tới attacker/victim.**

1. **File Lab01-01.exe**

**Load vào IDA:**

**A screenshot of a computer

Description automatically generated**

**Shift + F12 để tìm Strings, ta thấy có chuỗi C:\\Windows\System32\Kernel32.dll là đường dẫn đến dll kernel32 trong windows, đây là 1 dấu hiệu host-based indicators có thể tìm thấy trên hệ thống bị lây.**

1. **File Lab01-01.dll**

**A screenshot of a computer

Description automatically generated**

**Cũng làm tương tự như câu 5, ta thấy một địa chỉ IP 127.26.152.13 có thể là địa chỉ tới server C2, là 1 dấu hiệu network-based indicatiors có thể tìm thấy trên hệ thống bị lây.**

1. **Dựa theo các câu đã phân tích ở trên, có thể đoán file Lab01-01.exe là 1 loại trojan, nó chạy file dll để có thể backdoor hoặc chạy như 1 file RAT, nó tìm kiếm xem kernel32.dll qua đường dẫn xem nó có tồn tại không, và nếu không nó có thể copy dll độc hại tới C:\windows\system32\kerne132.dll. Sau khi thực thi dll, nó sẽ kết nối tới server C2 tại địa chỉ IP 127.26.152.13.**

***Lab 1-2***

1. **File Lab01-02.exe**

A screenshot of a computer

Description automatically generated

**Match 59/72 antivirus signatures**

1. **Load vào PEid**

**A screenshot of a computer

Description automatically generated**

**Dòng UPX ->** [**www.upx.sourceforge.net\***](http://www.upx.sourceforge.net*) **-> file bị packed bởi UPX**

**Load vào DIE**

**A screenshot of a computer program

Description automatically generated**

**Packer: UPX(3.04)[NRV,best] -> file bị packed bởi UPX 3.04.**

**Load vào Dependency Walker:**

**Kernel32.dll**

**A screenshot of a computer

Description automatically generated**

**Advapi32.dll**

****

**Msvcrt.dll**

**A white rectangular object with a black border

Description automatically generated**

**Wininet.dll**

**A white rectangular object with a black border

Description automatically generated**

**Ta thấy có rất ít hàm được import so với 1 file thực thi bình thường.**

**Load vào PEbear**

**A screenshot of a computer

Description automatically generated**

**Thông thường các section sẽ là .text, .rdata, .data, .reloc, nhưng ở đây lại là UPX0, UPX1, UPX2 -> dấu hiệu bị pack bởi UPX. Thêm nữa, ở các section, VirtualSize lại > SizeOfRawData, và ở UPX0, SizeOfRawData = 0, mà VirtualSize lại tận 0x4000, chứng tỏ khi load vào memory, section UPX0 sẽ là vùng được unpack. Thêm nữa, 2 section đầu lại có cả 3 quyền rwx, trong khi 1 section không thể có đồng thời cả 3 quyền như thế**

**Dựa trên các dấu hiệu trên, có thể thấy file đã bị pack.**

**Để unpack file, ta dùng upx**

**Nhập “upx -d Lab01-02.exe”**

**A screenshot of a computer

Description automatically generated**

**File sau khi unpack thì tên section đã trở thành .text, .rdata, .data như bình thường.**

1. **Load vào IDA:**

**A screenshot of a computer

Description automatically generated**

**A black rectangular object with a white border

Description automatically generated**

**Ta thấy file sao khi unpack có import 3 thư viện là kernel32.dll và advapi32.dll, wininet.dll**

**Kernel32.dll thì ta thấy có CreateThread để tạo thread ngầm, có thể chạy payload / shellcode tự động, CreateMutexA, OpenMutexA để tránh chạy nhiều lần, GetModuleFileNameA để lấy tên đường dẫn tới file thực thi hiện tại, CreateWaitableTimerA, SetWaitableTimer, WaitForSingleObject, 3 hàm này có thể nhằm mục đích để trì hoàn thời gian thực thi, có thể là để anti-debug, SystemTimeToFileTime để chuyển thời gian hệ thống sang thời gian file (dạng nano second).**

**Advapi32.dll ta thấy có các hàm liên quan đến Service như CreateServiceA, StartServiceCtrlDispatcherA, OpenSCManagerA, cùng với đó, trong Wininet.dll có 2 hàm: InternetOpenA để tạo 1 handle, nhằm tạo phiên kết nối, InternetOpenUrlA để mở kết nối HTTP/HTTPS(hoặc FTP) tới 1 URL cụ thể.**

**Qua phân tích sơ bộ, ta có thể đoán malware này sẽ giả dạng như 1 service thật trong Windows, sau đó tạo luồng, load shellcode từ server C2 ở xa thông qua InternetOpenUrlA, và có các hàm Timer để delay tránh bị phát hiện, debug.**

1. **Tìm Strings trong IDA:**

**A screenshot of a computer

Description automatically generated**

**MalService có thể là 1 Service mà malware tìm kiếm trong Service Control Manager, đây chính là 1 host-based indicator. Url** [**http://www.malwareanalysisbook.com**](http://www.malwareanalysisbook.com) **có thể là server C2 chứa shellcode.**

***Lab 1-3***

1. **Upload lên VirusTotal:**

A screenshot of a computer

Description automatically generated

**Match 66/72 antivirus signatures.**

1. **Load vào PEid:**

**A screenshot of a computer

Description automatically generated**

**FSG 1.0 -> dulek/xt -> File bị packed bởi FSG 1.0**

**Load file packed vào OllyDbg**

A screenshot of a computer

Description automatically generated

**Load vào OllyDbg thì chương trình nhảy tới Entry Point của wrapper, vì các packer thường giải nén phần unpack vào section khác rồi jmp vào đó. Vì vậy ta sẽ theo dõi chương trình nhảy từ stub -> code thật bằng cách trace over đến khi nhảy qua section mới.**

**Trong Plugin, chọn OllyDump -> Find OEP by Section Hop (Trace over) để tìm OEP khi nhảy qua section mới**

A screenshot of a computer

Description automatically generated

**Chương trình nhảy tới địa chỉ 0x401090, chính là OEP gốc. Nhưng chỉ thấy hiện các byte nên ta chuột phải, Analysis -> Analysis code để hiện code**

A screenshot of a computer

Description automatically generated

**Tiếp tục vào Plugin->OllyDump->Dump debugged process để dump phần unpack ra 1 file mới**

**A screenshot of a computer

Description automatically generated**

**Chọn Rebuild Import để fix IAT, chọn Metho2 để tìm tên DLL và API trong file dumped.**

**A screenshot of a computer

Description automatically generated**

A screenshot of a computer

Description automatically generated

**Kiểm tra trong CFF, thấy không còn chỉ mỗi GetProcAddress và LoadLibraryA như lúc đầu nữa mà xuất hiện nhiều hàm được Import hơn lúc đầu -> đúng với khi file thực thi bình thường cũng không thể chỉ import có 2 hàm -> unpack thành công.**

1. **Các hàm Import đa số liên quan đến COM – kiến trúc được thiết kế để xây dựng các thành phần phần mềm có thể tái sử dụng, ngôn ngữ-độc lập, và giao tiếp dễ dàng giữa các tiến trình hoặc module.**
2. **Trong Strings chỉ thấy 1 dấu hiệu network based indicator là url** [**http://www.malwareanalysisbook.com/ad.html**](http://www.malwareanalysisbook.com/ad.html)

***Lab 1-4***

1. **Load lên VirusTotal**

A screenshot of a computer

Description automatically generated

**Match 63/72 antivirus signatures**

1. **Load vào PEid**

**A screenshot of a computer program

Description automatically generated**

**File không bị pack**

**Load vào PEbear**

**A screenshot of a computer

Description automatically generated**

**Tên các section đúng và cũng theo thứ tự như các file PE bình thường, VirtualSize và RawSize cũng không lệch nhau quá nhiều.**

**A screenshot of a computer

Description automatically generated**

**Về phần Import cũng không thấy có gì bất thường -> File này không bị packed.**

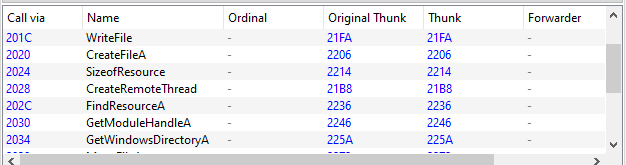
1. **Load vào PEview, chọn IMAGE\_NT\_HEADERS -> IMAGE\_FILE\_HEADER -> Time Date Stamp**

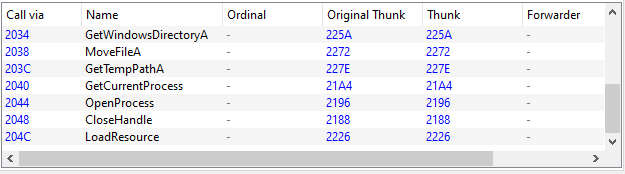
**A screenshot of a computer

Description automatically generated**

**Thời gian compiled của file là 2019/08/30 Fri 22:26:59 UTC**

1. **Ta có thể thấy chương trình import 2 thư viện chính là kernel32.dll và advapi32.dll**

****

****

**A screenshot of a computer

Description automatically generated**

**Từ ảnh ta có thể phân loại hành vi theo các nhóm hàm sau:**

* **CreateFileA, WriteFile, MoveFileA, GetWindowsDirectoryA, GetTempPathA**
* **FindResourceA, LoadResource, SizeofResource, GetModuleHandleA**

**Cho thấy chương trình:**

* **Ghi file xuống ổ đĩa (thường là dropper hoặc loader)**
* **Dùng resource như PE nhúng, shellcode, ảnh, dữ liệu → có thể sẽ giải nén hoặc unpack**
* **Biết được môi trường thư mục hệ thống → giúp giấu hoặc ngụy trang**
* **CreateRemoteThread, OpenProcess, GetCurrentProcess:**
* **Có thể là PE Injection, chèn shellcode vào tiến trình đang chạy**
* **FindResourceA, LoadResource, SizeofResource, GetModuleHandleA**
* **Bật đặc quyền SeDebugPrivilege**
* **Cho phép chương trình:**
  + **Mở tiến trình SYSTEM (như lsass.exe)**
  + **Dump memory, inject, bypass UAC**
  + **Truy cập file/registry được bảo vệ**
* **Dùng để leo thang đặc quyền (Privilege Escalation)**

1. **Trong Strings**

**A close up of a computer screen

Description automatically generated**

**Dựa trên ảnh trên thì đây là những host-based indicators, nhiều khả năng:**

* **Drop file xuống với các tên này (từ tài nguyên hoặc shellcode giải nén ra).**
* **Thực thi chúng bằng CreateProcess, WinExec, ShellExecute, v.v.**
* **Hoặc inject shellcode vào các file này nếu chúng là tiến trình hợp pháp (DLL sideloading, process hollowing...).**

****

**Còn đây là network indicator, url này có thể là nơi malware update hoặc nơi chứa payload.**

1. **Ném vào Resource Hacker**

**A screenshot of a computer

Description automatically generated**

**Ta thấy resource có signature của PE 4D 5A -> resource này cũng là 1 file PE.**

**Từ đó ta có thể đoán được rằng file PE này được giấu trong resource để có thể thực hiện 1 hành vi độc hại nào đó.**

***Lab 3-1***

1. **Load vào PEid:**

**A screenshot of a computer

Description automatically generated**

**File bị packed bởi PEncrypt 3.1 Final**

**Load vào PEbear:**

**A screenshot of a computer

Description automatically generated**

**Vì file bị packed nên phần Import chỉ thấy có mỗi kernel32.dll**

**Load vào PEstudio:**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a cell phone screen

Description automatically generated**

**Mặc dù bị packed, nhưng ta vẫn có thể thấy được 1 vài strings có liên quan đến Registry, 1 url có thể là server C2 chứa shellcode, 1 vài dll như advapi32, user32 có thể sẽ được load khi chạy mặc dù xem trong Import thì không có.**

1. **Mở Process Monitor và Process Explorer**

**Chạy Lab03-01.exe**

**Trong Procmon, Filter->Filter**

**A screenshot of a computer

Description automatically generated**

**Sửa lại như ảnh rồi Add->Apply để chỉ hiện tiến trình Lab03-01.exe**

****

**A computer screen shot of a computer

Description automatically generated**

**Khi đã run file, ta thấy 1 loạt Operation:**

**A screenshot of a computer

Description automatically generated**

**CreateFile: Mở file Lab03-01.exe**

**A screenshot of a computer

Description automatically generated**

**ReadFile: Đọc file Lab03-01.exe**

**A screenshot of a computer

Description automatically generated**

**CreateFile: Tạo file vmx32to64.exe trong thư mục C:\WINDOWS\system32\vmx32to64.exe**

**A screenshot of a computer

Description automatically generated**

**WriteFile: Ghi file Lab03-01.exe vào file mới vmx32to64.exe để tạo bản sao.**

**A screenshot of a computer

Description automatically generated**

**RegOpenKey: Mở registrykey HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run**

**A screenshot of a computer

Description automatically generated**

**Tạo 1 value với tên là VideoDriver cho registry key HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run, giá trị trỏ tới C:\WINDOWS\system32\vmx32to64.exe, value này có type là REG\_SZ, tức là chuỗi Unicode hoặc ANSI, kết thúc bằng ký tự null, có độ dài tối đa là 510.**

**Trong Process Explorer, View-> Lower Pane View -> Handles để hiện các handle được tạo**

****

**Ta thấy có tạo 1 mutex. BaseNamedObjects là namespace mặc định nơi lưu trữ các đối tượng có tên được tạo bởi các tiến trình user-mode, WinVMX32 là tên mutex. File tạo mutex để tại 1 thời điểm, chỉ có 1 instance được chạy.**

1. **Đọc gói tin trong Wireshark:**

****

A screenshot of a computer

Description automatically generated

**Malware cố gắng phân giải tên url** [**www.practicalmalwareanalysis.com**](http://www.practicalmalwareanalysis.com) **sang IP.**

**Sau khi phân giải tên miền, nó cố gắng gửi 256 bytes (có thể đã bị encrypt) tới IP đã phân giải (ở đây là 15.197.225.128)**

**A screenshot of a computer program

Description automatically generated**

***Lab 3-2***

1. **Load vào PEbear**

**A screenshot of a computer

Description automatically generated**

**Ta thấy trong Export Functions có hàm Install**

**Ta phải chạy rundll32.exe có trong thư mục C:\Windows\System32 để chạy dll trên, rồi thêm Install hoặc ordinal để install dll**

**A screenshot of a computer

Description automatically generated**

**Regshot 1 lần trc khi install để so sánh sự thay đổi registry khi regshot lần 2 sau khi install dll**

**A computer screen shot of a computer program

Description automatically generated**

**Install dll**

1. **Take regshot thứ 2**

**A screenshot of a computer

Description automatically generated**

**Ta thấy có thêm 1 số key được add**

**Ta thấy đã thêm 1 service mới: HKLM\SYSTEM\CurrentControlSet\Services\IPRIP**

**Để chạy service này, vào cmd, sc start IPRIP**

**A screenshot of a computer

Description automatically generated**

**Service đã được start**

1. **Service start có PID là 3480**

**A screenshot of a computer

Description automatically generated**

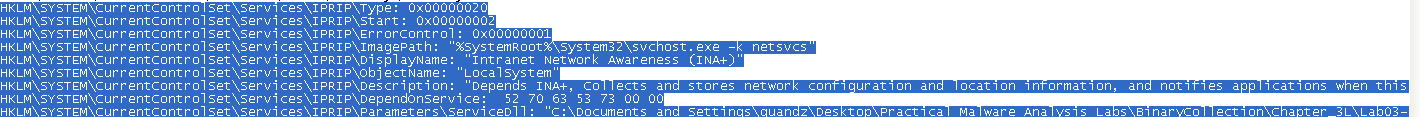
**Mở procexp, ta thấy có 1 tiến trình svchost.exe có PID 3480 đang chạy**

**Ctrl+F để tìm dll**

****

**Ta thấy dll của chúng ta đang chạy dưới tiến trình svchost.exe**

1. **Ta filter theo PID của tiến trình vì khi start service có hiện PID vì vậy ta sẽ filter theo PID để thu thập thông tin.**
2. **Trong log Regshot:**

****

**Khi install dll, trong registry, có 1 vài key và value của nó đã được add vào:**

* **HKLM\SYSTEM\CurrentControlSet\Services\IPRIP: Tạo service mới có tên là IPRIP**
* **ImagePath: "%SystemRoot%\System32\svchost.exe -k netsvcs": Service sẽ chạy trong nhóm netsvcs (nhóm svchost host nhiều service hệ thống)**
* **Type: 0x00000020: SERVICE\_WIN32\_SHARE\_PROCESS → Service chạy trong svchost.exe, dùng chung process với service khác(theo** [**https://learn.microsoft.com/en-us/windows-hardware/drivers/install/hklm-system-currentcontrolset-services-registry-tree**](https://learn.microsoft.com/en-us/windows-hardware/drivers/install/hklm-system-currentcontrolset-services-registry-tree) **)**
* **Start: 0x00000002: Auto start khi Windows khởi động**
* **DisplayName: Tên hiển thị của service(Intranet Network Awareness (INA+))**
* **ObjectName: "LocalSystem": Chạy với quyền System**
* **Description: Mô tả**
* **DependOnService: Service này phụ thuộc vào RpcSs (Remote Procedure Call)**
* **Parameters\ServiceDll: Đường dẫn tới dll thực tế**

1. **Trong PEbear  
   A screenshot of a computer

   Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Ta thấy có url practicalmalwareanalysis.com và các API send request , open request có thể sẽ gửi các request đến server C2 là practicalmalwareanalysis.com**

***Lab 3-3***

1. **Mở procexp trước, sau đó chạy file:**

**A screenshot of a computer

Description automatically generated**

**Lúc đầu có hiện tiến trình Lab03-03.exe, nhưng sau đó mất đi, và chỉ còn mỗi tiến trình svchost.exe, có thể khi chạy file Lab03-03.exe, trong registry đã thêm key để có thể chạy 1 service mới.**

1. **Trong procexp, chọn process svchost.exe, chuột phải Properties, vào Tab String**

**A screenshot of a computer program

Description automatically generated**

**Đây là string của file thực thi khi trên đĩa**

**A screenshot of a computer program

Description automatically generated**

**Đây là string của file khi đã run ở trong memory**

**Ta thấy có 1 vài hàm như SetWindowsHookExA, CallNextHookEx, UnhookWindowsEx, GetWindowTextA, GetForegroundWindow -> file này có thể là 1 keylogger giám sát hoạt động đánh keyboard từ người dùng và lấy tên các application khi người dùng mở 1 cửa sổ ứng dụng đó.**

1. **Ta để ý trong string của tiến trình khi load vào memory**

**A screenshot of a computer program

Description automatically generated**

**Có 1 file log tên là practicalmalwareanalysis.log**

**A computer screen shot of a computer

Description automatically generated**

**Và ta để ý sau khi chạy Lab03-03.exe, file log với tên đó cũng được tạo trong chính thư mục chạy file thực thi**

**Thật vậy, kiểm tra trong procmon:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Có xuất hiện operation tạo file log practicalmalwareanalysis.log ở trong chính đường dẫn chạy file thực thi**

1. **Như đã trình bày ở câu trước, vì khi kiểm tra trong string in live memory của tiến trình, có xuất hiện các hàm SetWindowsHookExA, CallNextHookEx, và như khi kiểm tra trong procmon, có operation CreateFile tạo file log practicalmalwareanalysis.log, ta có thể nghĩ đến đây là 1 loại keylogger.**

**Kiểm tra trong file log:**

**A screenshot of a computer

Description automatically generated**

**Có hiện tên các cửa sổ vì có sử dụng các hàm GetForegroundWindow + GetWindowTextA**

**Khi ta thử mở Wordpad nhập 1 đoạn chuỗi:**

**A screenshot of a computer

Description automatically generated**

**Mở lại file log:**

**A screenshot of a computer

Description automatically generated**

**Ta có thể chắc chắn đây là keylogger.**

***Lab 3-4***

1. **Ta thấy khi vừa chạy file, 1 cửa sổ cmd hiện lên sau đó mất đi, sau đó file thực thi cũng bị xóa**

**Kiểm tra trong procmon:**

****

**A screen shot of a computer

Description automatically generated**

**Dòng lệnh cmd trên dùng để xóa file thực thi và đóng cửa sổ cmd ngay sau khi chạy lệnh, >> NUL để ẩn thông báo xóa file thành công của lệnh del.**

1. **Có thể do thiếu tham số truyền vào khi chạy file, hoặc file phát hiện nó đang chạy trên máy ảo**
2. **Ném vào IDA**

**A computer screen shot of a program

Description automatically generated**

**Hàm main của file check tham số truyền vào:**

**Khi (argc == 1), tức là chỉ 1 tham số, tức là chỉ có tên file thực thi**

**A screenshot of a computer program

Description automatically generated**

**Hàm sub\_401000 kiểm tra sự tồn tại của 1 key trong HKLM, nếu tìm thấy key và value thì return 1, nếu không thấy key hoặc value thì trả về 0**

**A computer screen shot of a program

Description automatically generated**

**Hàm sub\_402410 sẽ thực hiện xóa chính file thực thi đó**

**A computer screen shot of a code

Description automatically generated**

**Cuối cùng là gọi hàm sub\_402360, để Sleep trong 1 khoảng thời gian.**

**Nếu ta chỉ truyền 1 tham số là file thực thi để chạy, thì nó sẽ chạy theo luồng trên.**

**A computer screen shot of a program

Description automatically generated**

**Phía dưới là ta truyền > 1 tham số, thì sẽ chạy theo các luồng khác nhau.**

**A screenshot of a computer program

Description automatically generated**

**Có thể chạy với tham số là in**

****

**Hoặc tham số là re**

****

**Hoặc là cc**

**Vậy ngoài cách chạy với 1 tham số là file thực thi, ta có thể chạy cùng với các tham số như: in, re, cc**

***Lab 5-1***

1. **Trong tab Function name, Ctrl + F tìm tên hàm, tìm DllMain:**

**A screen shot of a computer program

Description automatically generated**

**DllMain ở địa chỉ 0x1000D02E**

1. **Trong Import, tìm gethostbyname**

**A screenshot of a computer

Description automatically generated**

**Hàm gethostbyname nằm ở địa chỉ 0x100163CC**

1. **Bấm x để xref tới các hàm gọi tới hàm gethostbyname**

**A screenshot of a computer

Description automatically generated**

**Ta thấy có 18 lần hàm gethostbyname được gọi, và được gọi bởi 5 hàm khác nhau**

1. **G và paste địa chỉ 0x10001757 để nhảy tới địa chỉ đó**

****

**Đoạn sau truyền địa chỉ của off\_10019040 vào eax**

****

**Off\_10019040 trở tới địa chỉ của chuỗi “[This is RDO]pics.practicalmalwareanalysis.com’, 0**

**Sau đó add eax, 0Dh, sau đó push eax và call gethostbyname, tức là lấy địa chỉ của chuỗi off\_10019040 từ index 13 trở đi**

**Tức là DNS được request đến là pics.practicalmalwareanalysis.com**

1. **Dựa vào ảnh:**

**A screenshot of a computer program

Description automatically generated**

**Có 24 variables được IDA Pro nhận ra với subroutine ở địa chỉ 0x10001656**

1. **Không có parameter nào được IDA Pro nhận ra với subroutine ở địa chỉ 0x10001656**
2. **Dựa vào ảnh:**

****

**Located ở địa chỉ 0x10095b34**

1. **Dựa vào ảnh:**

**A screen shot of a computer program

Description automatically generated**

**Chuỗi \\cmd.exe /c được push như 1 tham số trong hàm strcat để nối chuỗi đó vào 1 con trỏ chuỗi Destination.**

**A screen shot of a computer program

Description automatically generated**

**Sau đó chuỗi Destination được xem như chuỗi dest được nối với src, con trỏ chuỗi Buf1, để sau đó Destination được push vào làm tham số thứ 2, lpCommandLine để gọi hàm CreateProcessA, nhằm chạy 1 tiến trình mới.**

1. **Xref lại đoạn đầu tiên dword\_1008E5C4:**

****

**Có 1 hàm được gọi ngay trước đoạn đó, có thể làm thay đổi eax, và vì dword\_1008e5c4 được gán cho eax, ta nên trace vào hàm đó xem nó làm gì**

**A screen shot of a computer program

Description automatically generated**

**Hàm trên gọi hàm GetVersionExA để kiểm tra phiên bản của Windows, sau đó kiểm tra xem dwPlatformId có bằng 2(tức là Windows NT) không, nếu bằng 2 thì setz al, al = 1, còn không thì al = 0**

**Sau đó eax được set cho dword\_1008E5C4**

1. **Nếu memcmp return 0, chương trình sẽ gọi hàm sub\_100052a2**

**A screen shot of a computer code

Description automatically generated**

**Hàm sub\_100052a2**

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer code

Description automatically generated**

**Dựa vào ảnh, hàm sub\_100052a2 sẽ gọi RegOpenKeyExA mở registry key HKLM\\SOFTWARE\\Microsoft\\CurrentVersion, sau đó gọi RegQueryValueExA để Type và Data với mỗi value có trong key đó.**

1. **Dựa vào ảnh:**

**A computer screen shot of a program

Description automatically generated**

**Đầu tiên chương trình gọi hàm sub\_100036c3**

**A computer screen shot of a program code

Description automatically generated**

**Gọi hàm GetVersionExA để lấy version của Windows, nếu không phải là Windows NT thì sẽ thoát**

**Sau đó PSLIST gọi hàm sub\_10006518**

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**Sau đó gọi các hàm CreateToolHelp32Snapshots, ProcessFirst, EnumProcessModules, GetModuleFileNames, ProcessNext để list thông tin các tiến trình đang chạy trên OS, bao gồm PID, Handles, tên Modules,…**

1. **G để nhảy tới địa chỉ của hàm sub\_10004E79, Sau đó View -> Graph -> Xrefs from để xem các hàm Windows API được gọi khi vào hàm này**

**A computer screen shot of a computer

Description automatically generated**

**Dựa trên các hàm API, có thể đổi tên hàm này thành LanguageIdentifier\_Send**

1. **Sử dụng xref charts:**

**A screenshot of a computer

Description automatically generated**

**Chỉnh recursive = 1 để xem những hàm DllMain gọi trực tiếp và untick cross references to để loại những hàm mà gọi tới hàm DllMain**

**A screenshot of a computer

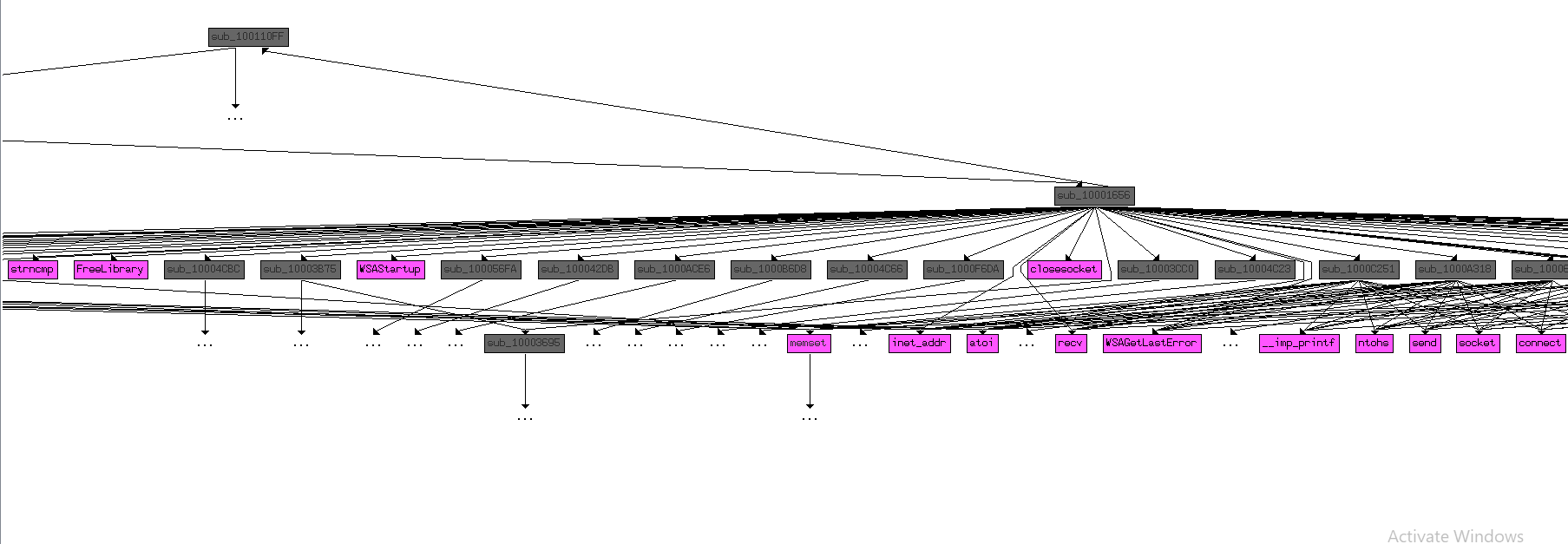
Description automatically generated**

**Ta thấy có 1 hàm Windows API được gọi trực tiếp là CreateThrread**

**Chỉnh recursive = 2 để xem những hàm được gọi gián tiếp từ DllMain, tức là những hàm được gọi trực tiếp bởi những hàm ở mức recursive = 1**

**A diagram of a diagram

Description automatically generated**

****

**A black and white image of a black and white image of a black and white image of a black and white image of a black and white image of a black and white image of a black and

Description automatically generated**

**Ta thấy có 33 hàm khi chỉnh recursive = 2.**

1. **G để nhảy tới địa chỉ đó**

**A screen shot of a computer code

Description automatically generated**

**Eax trỏ tới off\_100019020 chứa địa chỉ của chuỗi “[This is CTI]30”, sau đó add eax, 0Dh, thì eax trỏ tới địa chỉ của chuỗi “30”. Push eax rồi call atoi sẽ convert chuỗi 30 thành số 30, sau đó imul eax, 3E8h, thì eax = eax\*0x3e8 = 30000. Vậy chương trình sau khi gọi Sleep sẽ sleep 30000 milliseconds giây.**

1. **G để nhảy tới địa chỉ đó**

**A screen shot of a computer code

Description automatically generated**

**Tham số được truyền theo cdecl convention, tức là tham số cuối cùng sẽ được push cuối cùng.**

**3 tham số của hàm số socket theo thứ tự dựa vào ảnh là 2, 1, 6**

1. **Link msdn:** [**https://learn.microsoft.com/en-us/windows/win32/api/winsock2/nf-winsock2-socket**](https://learn.microsoft.com/en-us/windows/win32/api/winsock2/nf-winsock2-socket)

**A black and white text on a black background

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A black screen with white text

Description automatically generated**

**A black screen with white text

Description automatically generated**

**Từ ảnh, ta có thể thấy tham số truyền vào là AF\_INET, SOCK\_STREAM, IPPROTO\_TCP, tức là socket sử dụng địa chỉ IPv4, socket kiểu stream(luồng), và sử dụng giao thức TCP.**

**Để chuyển dạng Id của tham số thành dạng tên như trên, chọn vào tham số M để dùng enum**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer code

Description automatically generated**

1. **Alt+B để tìm binary khớp với opcode 0xED, ta thấy chỉ có 1 kết quả sử dụng instruct “in”**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Xref đến hàm sử dụng instruction “in”, đoạn trên thực hiện I/O read từ cổng 0x5658, nếu malware đang chạy trong Vmware, ebx sẽ bị biến đổi thành ‘VMXh’ và khi đó sẽ detect malware đang chạy trong vmware.**

1. **G để nhảy tới địa chỉ:**

**A screenshot of a computer program

Description automatically generated**

**Ta thấy 1 dãy các byte**

1. **Alt + F7 để chạy script**

**A screenshot of a computer

Description automatically generated**

**Chọn script Lab05-01.py**

**Sau khi run script, đoạn data sẽ biến đổi thành như này**

**A screenshot of a computer program

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

1. **A để convert sang dạng ASCII**

****

1. **Scirpt:**

**A screenshot of a computer program

Description automatically generated**

**Script sẽ duyệt lấy byte tại địa chỉ cursor -> cursor + 0x50 -1 , sau đó xor với 0x55, patch byte đó tại chính địa chỉ đó. Mục đích là để decode.**