# LỜI CẢM ƠN

Em xin chân thành cảm ơn các thầy, cô giáo Bộ môn An toàn Hệ thống thông tin đã tận giảng dạy và giúp đỡ em có những kiến thức cần thiết để hoàn thành bản báo cáo này. Em cảm ơn thầy Trịnh Minh Đức – giảng viên bộ môn An toàn Hệ thống thông tin đã giúp đỡ, hướng dẫn và chỉ bảo để em hoàn thành.

Vì điều kiện thời gian thực tập cơ sở có hạn cũng như hiểu biết còn hạn chế nên chắc chắn đề tài không tránh khỏi những sai sót. Rất mong các thầy cô góp ý kiến để bản thực tập cơ sở của em được hoàn thiện hơn.

Em xin chân thành cảm ơn!

*Thái Nguyên, ngày 3 tháng 3 năm 2017*

**Sinh viên thực hiện**

Vũ Thanh Bình

# MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc477529242)

[MỤC LỤC 2](#_Toc477529243)

[DANH MỤC HÌNH ẢNH 4](#_Toc477529244)

[DANH MỤC TỪ VIẾT TẮT 5](#_Toc477529245)

[MỞ ĐẦU 6](#_Toc477529246)

[CHƯƠNG 1.](#_Toc477529250) [PHƯƠNG PHÁP PHÁT HIỆN VÀ PHÂN TÍCH MÃ ĐỘC 7](#_Toc477529251)

[1.1 Giới thiệu về mã độc 7](#_Toc477529252)

[1.1.1 Khái niệm mã độc 7](#_Toc477529253)

[1.1.2 Lịch sử của mã độc 7](#_Toc477529254)

[1.2 Cơ chế hoạt động của mã độc 11](#_Toc477529255)

[1.2.1 Cơ chế hoạt động của Virus 11](#_Toc477529256)

[1.2.2 Cơ chế hoạt động của Worm 11](#_Toc477529257)

[1.2.3 Cơ chế hoạt động của Trojan Horse 11](#_Toc477529258)

[1.3 Các kiểu dấu hiệu mã độc và kỹ thuật phát hiện tương ứng 11](#_Toc477529259)

[1.3.1 String – Chuỗi 12](#_Toc477529260)

[1.3.2 Ký tự đại diện 12](#_Toc477529261)

[1.3.3 Mã băm 13](#_Toc477529262)

[1.3.4 Khung mã độc sẵn có 13](#_Toc477529263)

[1.3.5 Phương pháp dựa trên hành vi 13](#_Toc477529264)

[1.3.6 Kỹ thuật lọc 14](#_Toc477529265)

[1.3.7 Phát hiện bằng việc giải mã tĩnh 15](#_Toc477529266)

[1.3.8 Mã giả lập 15](#_Toc477529267)

[1.4 Quy trình phân tích mã độc 16](#_Toc477529268)

[1.4.1 Các phương pháp phân tích mã độc 16](#_Toc477529269)

[1.4.2 Các bước cơ bản phân tích mã độc 17](#_Toc477529270)

[1.4.3 Phân tích mã dịch ngược ( Static Analynis) 19](#_Toc477529271)

[1.4.4 Trace Code Debug 20](#_Toc477529272)

[1.4.5 Tìm signature của mã độc và đưa vào cơ sở dữ liệu 20](#_Toc477529273)

[CHƯƠNG 2.](#_Toc477529274) [CÁC PHƯƠNG PHÁP TẠO MẪU MÃ ĐỘC 21](#_Toc477529275)

[2.1 Nghiên cứu chuẩn trao đổi dữ liệu mã độc 21](#_Toc477529276)

[2.2 Nghiên cứu cơ sở dữ liệu mã độc Clam AV từ đó xây dựng cơ sở dữ liệu mẫu 24](#_Toc477529277)

[2.2.1 Clam Anti Virus 24](#_Toc477529278)

[2.2.2 ClamAV Virus Databases 25](#_Toc477529279)

[2.2.3 Debug thông tin từ libclamav 25](#_Toc477529280)

[2.2.4 Định dạng chữ ký của CLAMAV 26](#_Toc477529281)

[CHƯƠNG 3.](#_Toc477529282) [XÂY DỰNG CƠ SỞ DỮ LIỆU MẪU MÃ ĐỘC 35](#_Toc477529283)

[3.1 Xây dựng chương trình tạo mẫu mã độc 35](#_Toc477529284)

[KẾT LUẬN 37](#_Toc477529285)

[TÀI LIỆU THAM KHẢO 38](#_Toc477529286)

[PHỤ LỤC 39](#_Toc477529287)

# DANH MỤC HÌNH ẢNH

[Hình 1.1 Mô hình chương trình quét hành vi 14](#_Toc477529235)

[Hình 3.1 Giao diện chương trình khi tạo mẫu mã độc 35](#_Toc477529236)

[Hình 3.2 Dữ liệu mã độc trong file text 36](#_Toc477529237)

# DANH MỤC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| CPU | Central Processing Unit |
| Ddos | Distributed Denial of Service |
| MD5 | Message-Digest algrorithm 5 |
| MS-DOS | Microsoft Disk Operating System |
| HTML | Hypertext Tranfer Protocol |
| ICSG | Industry Connection Sercurity Group |
| XML | Extensible Markup Language |
| DLL | Dynamic-link Library |
| SHA | Message-Digest algrorithm 5 |
| Malware | Malicious software |
| PE | Portable Executable |
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| AV | Anti Virus |
| C# | C sharp |

## MỞ ĐẦU

Khi nhu cầu và việc sử dụng Internet của con người ngày càng tăng thì cũng là lúc những mối đe dọa xuất hiện càng nhiều, nổi bật là đe dọa của mã độc. Mã độc xuất hiện bất kỳ ở đâu trên môi trường của các thiết bị điện tử như các đĩa mền, 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…. Khi mã độc nhiễm vào một máy tính nào đó thì nó lây lan sang các máy tính khác và thiệt hại do mã độc gây ra khó có thể lường trước được.

Hiện tại để chống lại các loại mã độc người ta thường sử dụng các chương trình Antivirus. Tuy nhiên để có được một chương trình Antivirus một cách có hiệu quả cần có một thuật toán quét sao cho nhanh nhất và quan trọng là một cơ sở dữ liệu đầy đủ và cập nhật thường xuyên những mẫu mới. Ngoài ra cơ sở dữ liệu đó phải đảm bảo kiến trúc, định dạng và phương pháp lưu trữ một cách khoa học để cho phép các thuật toán đối sánh mẫu được thực hiện một cách hiệu quả nhất. Đồng thời nó phải có các cơ chế đáp ứng được các tiêu chuẩn để có thể trao đổi với các cơ sở dữ liệu cả các hãng khác. Việc xây dựng được cơ sở dữ liệu như vậy có thể sử dụng để phát triển các phần mềm phát hiện và ngăn chặn mã độc hiệu quả. Ngoài ra nó cũng có thể sử dụng như một công cụ để hỗ trỡ đắc lực cho những người làm công việc tác nghiệp phân tích mã độc.

Chính vì những lý do trên, nên mục đích của báo cáo này là để nghiên cứu, tìm hiểu các kiểu dấu hiệu để phát hiện ra các loại mã độc, các phương pháp phân tích mã độc, các chuẩn để trao đổi mẫu mã độc và cấu trúc chung của cơ sở dữ liệu mã độc. Đồng thời tìm hiểu cách thiết kế, xây dựng một cơ sở dữ liệu mẫu mã độc và một chương trình demo xây dựng mẫu mã độc để từ đó xây dựng cơ sở dữ liệu mẫu mã độc.

Bản báo cáo này được trình bày theo bố cục:

## Chương 1. Phương pháp phát hiện và phân tích mã độc

Trình bày tổng quan về mã độc, cơ chế hoạt động của mã độc và quy trình thu thập, phát hiện và phân tích mã độc.

## Chương 2. Các phương pháp tạo mẫu mã độc

Tìm hiểu nghiên cứu chuẩn trao đổi dữ liệu mẫu mã độc, cơ sở dữ liệu mã độc của chương trình CLAMAV. Trình bày về một số phương pháp phát hiện mã độc cơ bản mà các chương trình antivirus hay dùng để tìm và diệt mã độc.

## Chương 3. Xây dựng cơ sở dữ liệu mẫu mã độc

Xây dựng cơ sở dữ liệu theo chuỗi nhận dạng

# CHƯƠNG 1.

# PHƯƠNG PHÁP PHÁT HIỆN VÀ PHÂN TÍCH MÃ ĐỘC

Xây dựng cơ sở dữ liệu mẫu mã độc đạt yêu cầu thì cần phải trải qua nhiều quy trình, trong đó quan trọng nhất là quá trình phát hiện và phân tích mã độc. Nó giúp cho việc xây dựng cơ sở dữ liệu mẫu mã độc một cách hiệu qủa và nhanh chóng. Cùng với đó quá trình này đem lại cho ta cái nhìn rõ nét nhất về các hành vi cũng như là đặc điểm mà mã độc đó gây hại cho người dùng. Vì vậy chúng ta sẽ đi sâu vào quá trình phát hiện và phân tích mã độc.

## Giới thiệu về mã độc

* + 1. ***Khái niệm mã độc***

Malware [1] (từ ghép của Malicious: độc hại và Software: phần mềm) là tên gọi chung cho các phần mềm được thiết kế, lập trình đặc biệt để phá hoại hệ thống của bạn hoặc làm gián đoạn môi trường hoạt động mạng, có khả năng làm hại máy tính hay gây ảnh hưởng trái phép tới người dùng mà không có sự đồng ý của chủ sở hữu. Tuỳ theo cách thức sử dụng, các loại malware sẽ có mức độ nguy hiểm khác nhau.

* + 1. ***Lịch sử của mã độc***

Internet phát triển, nó thực hiện việc kết nối các máy tính, máy chủ, laptop, mobile phone trên khắp thế giới, thêm vào là sự phát triển của các website vì vậy mã độc cũng theo đó phát triển rộng khắp với số lượng ngày càng lớ và phức tạp hơn.

Một số mốc lịch sử [2] của mã độc trên thế giới.

* Kể từ lần đầu tiên virus Brain tấn công vào năm 1986, nhiều loại mã độc worm và trojan đã liên tiếp tấn công và lây nhiễm máy tính người dùng. Mối đe dọa lớn nhất đối với người dùng là việc xác định được một lượng đáng kể virus qua mỗi năm. Virus đầu tiên hoạt động trên nền tảng IBM PC và là virus đầu tiên sử dụng các cơ chế đa hình. Pakistani Brain lây nhiễm sector khởi động của đĩa mềm, và đã lan rộng trên phạm vi toàn cầu trong vài tuần.
* Năm 1987, STONED, virus lây nhiễm sector khởi động đã sớm lan truyền qua New Zealand và Australia. Máy tính bị lây nhiễm sẽ hiển thị: “Your PC is now Stoned” và “Legalize Marijuana”. Virus Stoned có nhiều biến thể và đã rất phổ biến trong những năm đầu thập niên 1990.
* Năm 1988, MORRIS WORM được phát triển bởi Robert Tappan Morris Jr., con trai của một nhà khoa học NCSC trước đây, Morris được coi là mã độc đầu tiên, có lẽ đã lây lan qua hàng ngàn hoặc hàng chục ngàn máy tính mini và trạm chạy VMS, BSD , SunOS.
* Năm 1989, DISK KILLER là một trong những loại virus phá hoại sớm nhất, nó lây nhiễm sector khởi động và từ từ làm hỏng ổ đĩa. Thỉnh thoảng một thông điệp: “Computer Orge”, sẽ nhấp nháy trên màn hình các máy tính bị nhiễm.
* Năm 1990, WHALE là virus tiên phong trong công nghệ anti-debugging, kỳ dị và hết sức hiệu quả. Một nhà nghiên cứu vào thời điểm đó đã mô tả bản sao của Whale như là việc : “các nhà nghiên cứu antivirus gửi vật mẫu cho nhau”. Thật không may, tác giả của mã độc đã học được nhiều điều kể từ đó.
* Năm 1991, MICHELANGELO, đáng kể nhất là vào 6/3, khi biến thể của virus Stoned được kích hoạt, lây nhiễm vào sector khởi động của đĩa mềm và MBR trên ổ đĩa cứng. Vì phần lớn mã độc được kích hoạt vào 6/3, rất khó có thể phát hiện máy tính bị nhiễm.
* Năm 1992, TRIDENT POLYMORPHIC ENGINE (TPE), một engine có thể chuyển đổi một chương trình thành một phiên bản mới bằng việc sử dụng các mã khác nhau nhưng vẫn giữ được chức năng ban đầu. Đây có thể là cách mà các virus sử dụng để tránh khỏi sự phát hiện.
* Năm 1993, DARK ANGEL’S MULTIPLE ENCRYTOR (DAME), một engine đa hình khác được phát hành bởi nhóm Phalcon/SKISM, có nguồn gốc từ Canada. Mã độc này được phân phối dưới dạng mã nguồn.
* Năm 1994, ONEHALF có thể được coi là virus Ransomware đầu tiên, ngoài trừ việc không đòi tiền chuộc hoặc mã ngừng hoạt động. Mã độc này mã hóa một chuỗi các sector đầu tiền trên ổ cứng. Nếu bạn sử dụng FDISK/MBR, MBR bị lây nhiễm sẽ đực thay thế bằng một cái hoàn toàn mới, hệ thống sẽ không thể khởi động.
* Năm 1995, WM/CONCEPT là virus macro đầu tiên lây nhiễm thông qua Microsoft word – WM/Concept đã tạo ra nhiều vấn đề. Ban đầu, Microsoft không phát hành các định dạng của Office File (OLE2). Tại một hội nghị EICAR ở Linz, các thành viên CARO đã cùng ngồi và dịch ngược để tìm ra đúng lỗi và khắc phục.
* Năm 1996, LAROUX mặc dù hông phải là virus đầu tiên liên quan tới Excel nhưng WM/Laroux là virus macro trong Excel nguy hiểm nhất từng thấy. Mã virus thực tế gồm 2 macro: Auto\_Open và Check\_File, ẩn trong 1 datasheet có tên :”laroux”.
* Năm 1998, AUTOSTART 9805 có thể không phải là một virus, mà là một worm, Nó nhân rộng bằng cách sao chép chính nó, nhưng không tự gắn hay ký sinh vào một chương trình. Các biến thể ban đầu nhanh chóng có mặt tại Hồng Kông và Đài Loan vào 4/1998, và nhanh chóng có báo cáo về sự có mặt của nó tại ít nhất 4 lục địa.
* Năm 1999, MELISSA tác động trên diện rộng, mã độc worm lây nhiễm các hệ thống Intel và Microsoft thông qua ứng dụng email MS Outlook. Mã độc này được cung cấp thông qua một tệp tin đính kèm MS Word, được chuyển tiếp đến 50 địa chỉ Outlook đầu tiên của nạn nhân.
* Năm 2000, LOVELETTER một con mã độc worm được cho là đã tấn công hàng chục triệu máy tính Windows, mã độc này được đính kèm trong email gửi đến, có khả năng truy cập hệ điều hành, lưu trữ thứ cấp, hệ thống và dữ liệu người dùng của nạn nhân.
* Năm 2001, NIMDA là worm máy tính đặc biệt hiệu quả trong các phương pháp tấn công khác nhau, bao gồm email, chia sẻ trong mạng mở và các trang web bị xâm nhập. Lúc đầu, Nimda có liên quan tới A1 Qaeda trng các phương tiện truyền thông gắn với sự kiện 11/9 nhưng giả thuyết này chưa bao giờ được chứng minh.
* Năm 2002, KLEZ một mã độc worm đính kèm một virus đa hình. Mỗi lần thực thi trên một máy tính bị lây nhiễm, mã độc tự gửi chính nó đến các điịa chỉ được tìm thấy trong hệ thống. Đáng chú ý là kỹ thuật “giả mạo người gửi”, thay thế email được lựa chọn, kéo theo nhiều hiểu lầm và sai lệch.
* Năm 2003, SQL SLAMMER mã độc worm này khai thách một lỗ hổng trong Microsoft SQL Sever và lây nhiễm nhanh chóng hầu hết các nạn nhân khác chỉ trong 10 phút bằng cách tự sao chép gói tin mạng. Toàn bộ mạng Internet sẽ trở nên rất chậm trong hôm đó.
* Năm 2004, MYDOOM một trong những worm lây lan hàng loạt trong thập kỷ đầu tiên của thế kỷ 21. Phiên bản gốc đã được ghi nhận với sự lây lan nhanh chóng, đáng nhớ nhất là việc tiến hành triển khai các cuộc tấn công DdoS vào tập đoàn SCO và Microsoft.
* Năm 2005, COMMWARRIOR virus đầu tiên trên điện thoại di động có thể lây lan qua tin nhắn MMS và Bluetooth, nhắm vào các điện thoại thông minh Symbian Series 60. Ảnh hưởng của mã độc này không lớn nhưng lại tác động mạnh tới các chuyên gia Antivirus.
* Năm 2006, VB.NEI hay còn gọi Nyxem, Blackmail hoặc MyWif. Mã độc này được nhận nhiều sự chú ý vì nó sử dụng một bộ đếm cho phép các nhà nghiên cứu theo dõi số lượng các máy chủ bị lây nhiễm. VB.NEI cũng đáng chú ý vì đã xóa tập tin, phá hủy dữ liệu.
* Năm 2007, STORM đã bắt đầu lây nhiễm các máy tính trên khắp châu Âu và Hoa Kỳ, lan truyền thông qua một email tuyên bố về thảm họa thời tiết. Các máy tính bị lây nhiễm sẽ trở thành một phần của mạng botnet.
* Năm 2008, CONFICKER là một mạng botnet lây lan rộng, kéo dài. Mã độc sử djng các thuật toán để ngăn cản việc truy tìm một con trỏ được phát triển nhiều trong tương lai.
* Năm 2009, TDL3 sáng tạo và thích nghi là đặc điểm của rootkit này và bản kế thừ TDL4. Mã độc này đã đưa ra một thay đổi mới về ý tưởng như các mạng P2P và che giấu mã độc. TDL đã sử dụng hiệu quả một hệ thống ẩn tệp tin.
* Năm 2010, STUXNET worm trong lĩnh vực quân sự đầu tiên đã tạo một điểm nhấn lớ với việc làm ảnh hưởng đến các hệ thống ở mức thấp. Mã độc này nhằm vào các hệ thống điều khiển công nghiệp và đã được sử dụng để chống lại cơ sở hạt nhân của Iran.
* Năm 2011, KELIHOS có khả năng là một kế thừa của mã độc Storm, botnet này chủ yếu được sử dụng để thực hiện các chiến dịch thư rác và đánh cắp thông tin.
* Năm 2012, MEDRE thông tin bị đáh cắp bởi virus đíh kèm trong các tài liệu AutoCAD. Đội ngữ nghiên cứu của ESET đã phát hiện và phân tích mối đe dọa này để thấy rằng mã độc này được phát hiện để ăn cắp các bản thiết kế từ hầu hết các công ty tư nhân ở Peru.
* Năm 2013, HESPERBOT Trojan tiên tiến này nhằm mục tiêu vào người dùng ngân hàng trực tuyế với chiến lược lừa đảo, lấy thông tin liên quan đến các tổ chức. Kẻ tấn công có được thông tin đăng nhập thông qua việc dụ dỗ nạn nhân để thực thi các mã độc.
* Năm 2014, WINDIGO chiến dịch tin tặc đã chiếm quyền kiểm soát hơn 25.000 máy chủ Unix trên toàn thế giới, gửi hàng triệu thư rác mỗi ngày. Các thành phần của mã độc được thiết kế tinh vi để chiếm quyền điều khiển máy chủ, lây nhiễm các máy tính liên quan và ăn cắp thông tin.

## Cơ chế hoạt động của mã độc

* Mã độc có nhiều loại nhưng mục đích cuối cùng là phá hoại người dùng. Ở đây em sẽ nêu ra các loại mã độc phổ biến nhất là virus, worm và trojan horse [1].
  + 1. ***Cơ chế hoạt động của Virus***

Virus [3] không thể tồn tại độc lập nên cách hoạt động duy nhất là chúng lây lan qua các file trên máy tính người dung. Thường thì virus lây lan với mục đích là xóa, sửa file trên máy tính. Loại virus nguy hiểm nhất là loại virus đa hình chúng tự đính kèm vào các file exe, các file này vẫn hoạt động bình thường. Sau mỗi lần lây lan các đoạn mã virus được thay đổi theo các phương pháp khác nhau để che dấu đặc điểm nhận dạng của nó.

* + 1. ***Cơ chế hoạt động của Worm***

Worm [3] tồn tại một cách độc lập nên nó cách thức hoạt động của nó tuy là cũng tự nhân đôi, sao chép chính nó nhưng không cần lân lan vào một file nào. Worm hoạt động không cần tác động của người dung và thường lây lan qua mạng LAN hoặc Internet.

* + 1. ***Cơ chế hoạt động của Trojan Horse***

Trojan Horse [3] thường ẩn mình dưới dạng là một chương trình an toàn, vô hại với máy tính người dung. Chính vì vậy nó ngoài thực thực thi chức năngchương trình an toàn nó còn âm thầm thực thi các chức năng độc hại ẩn trong đó mà khi nó cần đến mới thực thi.

## Các kiểu dấu hiệu mã độc và kỹ thuật phát hiện tương ứng

Phương pháp này sử dụng một cơ sở dữ liệu có sẵn khi quét các chương trình quét chỉ việc quét các tập tin và so sánh với cơ sở dữ liệu có sẵm. Cơ sở dữ liệu mỗi hãng phát triển phần mềm diệt virus được tạo ra theo những cách khác nhau cũng với đó là thuật toán quét sao cho nhanh và hiệu quả nhất. Chúng đều được các hãng giữ kín và thường không tiết lộ ra ngoài.

* + 1. ***String – Chuỗi***

Phương pháp phát hiện theo chuỗi [4] là phương pháp đơn giản nhất để phát hiện mã độc. Nó sử dụng một chuỗi các byte hoặc kỹ tự chỉ có trong mã độc mà không có trong các chương trình bình thường khác. Các chuỗi sau khi đã được trích xuất ra được lưu trong một cơ sở dữ liệu, các chương trình phát hiện mã độc sẽ sử dụng cơ sở dữ liệu này để phát hiện các mã độc.

Phương pháp này có nhược điểm là đòi hỏi thuật toán quét phải tốt nếu không thời gian quét sẽ rất lâu nếu lượng tệp tin cần quét là lớn hoặc cơ sở dữ liệu lơn. Một nhược điểm nữa là việc tìm chuỗi nhận dạng này cần đối với mỗi hãng phát triển phần mềm diệt virus là khác nhau nên quá trình trao đổi dữ liệu giữa các hãng là không có.

Phương pháp này có ưu điểm là đối với các biến thể khác nhau của cùng một loại mã độc thì chuỗi nhận dạng vẫn giống nhau nếu việc tìm chuỗi nhận dạng tốt. Thường các loại mã độc ngay cả các mã độc tự biến đổi mã của mình vẫn giữa nguyên các chuôi nhận dạng của chúng.

Dưới đây là một chuỗi nhận dạng của virus Stoned:

*0400 B801 020E 07BB 0002 33C9 8BD1 419C*

Chuỗi này là chuỗi nhận dạng 16 byte thường được sử dụng với virus 16 bit. Đối với virus 32 bit thường sử dụng chuỗi 32 byte.

* + 1. ***Ký tự đại diện***

Ký tự đại diện [4] cũng giống như chuỗi nhận dạng tuy nhiên khi các chương trình phát hiện mã độc sử dụng cơ sở dữ để quét thì nó được phép bỏ qua một byte hoặc một dãy byte.

Ví dụ:

*0400 B801 020E 07BB ??02 %3 33C9 8BD1 419C*

Bắt đầu duyệt:

* Nếu khớp với 04 thì tiếp tục
* Khớp với 00 thì tiếp tục
* Tiếp tục cho đến khi gặp ?? thì bỏ qua byte này
* Khớp với 02 thì tiếp tục
* Gặp %3 33 thì nghĩa là tìm khớp 33 với 3 vị trí sau nếu khớp thì tiếp tục
* Duyệt đến hết chuỗi nếu khớp hết thì gửi lại kết quả

Ký tự đại điện thường được sử dụng đối với các byte bị mất cho phép quét một cách chính xác hơn. Đối với với mã độc được mã hóa, virus đa hình thời đầu có thể dễ dàng bị phát hiện bởi phương pháp này.

* + 1. ***Mã băm***

Mã băm [4] là thuật ngữ chung cho kỹ thuật tăng tốc tìm kiếm các thuật toán. Nó có thể được thực hiện trên các byte đầu tiên từ 16 bit và 32 bit của chuỗi quét. Điều này cho phép thêm byte để chứa ký tự đại diện. Các nhà nghiên cứu mã độc có thể kiểm soát mã băm tốt hơn bằng cách chọn lọc những byte bắt đầu của chuỗi chứa nó.

Việc lấy mã băm có thể là băm toàn bộ tệp tin hoặc một phần tệp tin như phần đầu hay phần cuối tệp tin. Mã độc thường chèn đoạn code thực thi của chúng vào đầu hoặc vào cuối tệp tin nên việc băm một phần sẽ giúp giảm thời gian băm và cơ sở dữ liệu băm có hiệu suất cao hơn.

Nhược điểm của mã băm là đối với các mã độc có khả năng tự biến đổi hay xáo trộn mã thì sẽ gặp khó khăn vì có thể cùng một loại mã độc nhưng sẽ có nhiều mã băm khác nhau.

* + 1. ***Khung mã độc sẵn có***

Phương pháp này được tạo bởi Eugene Kaspersky. Phương pháp khung mã độc có sẵn rất hữu ích trong việc phát hiện họ virus macro. Thay vì lựa chọn một chuỗi đơn giản hoặc một kiểm tra của tập các virus macro, chương trình quét phân tích các dòng báo cáo vĩ mô để lược bỏ các thông báo không cần thiết. Kết quả là xác định được các mã vĩ mô nằm trong virus macro sau đó so sánh với cơ sở dữ liệu có sẵn.

Phương pháp này dung để phát hiện rất tốt các biến thể của một họ virus.

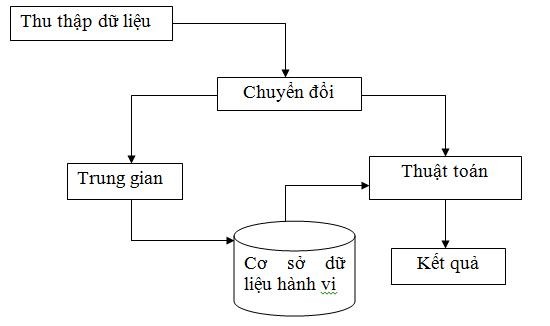
* + 1. ***Phương pháp dựa trên hành vi***

Phương pháp phát hiện dựa trên hành vi [4] khác với việc phát hiện đựa vào bề ngoài, cấu trúc tệp tin có sẵn là nó xác định các hành động thực hiện của mã độc hơn là việc xác 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 moeis 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 module 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 1.1 Mô hình chương trình quét hành vi

* + 1. ***Kỹ thuật lọc***

Kỹ thuật lọc [4] sử dụng trong các chương trình quét sau này. Các mã độc lây nhiễm thường là tập hợp con của các loại đối tượng đã biết. Do đó các chương trình quét có thể tiến hành quét một cách rất nhanh sử dụng kỹ thuật này. Ví dụ như chữ ký của một loại virus bị giới hạn để tự khởi động, trong hệ điều hành DOS thì chữ ký tệp tin exe có đặc điểm khác. Vì vậy một cờ được thêm vào chuỗi nhận dạng để xác định xem có chuỗi ký tự này trong đối tượng cần quét hay không. Do đó nó làm giảm thời gian quét của chương trình phải thực hiện.

Chương trình quét phục thuộc rất nhiều vào bộ lọc. Bộ lọc có tốt thì thuật toán quét mới hiệu quả, thời gian mới rút ngắn. Một bộ lọc có thể là : các loại thực thi, dấu nhận dạng mã độc, tên mã…Tuy nhiên có một số mã độc việc tìm kiếm bộ lọc rất phức tạp thậm chí là không thể tìm được điển hình là các virus đa hình hay mã độc tự biến đổi mã hay mã hóa.Những loại này được phát hiện tốt hơn với kỹ thuật phát hiện giải mã tĩnh.

* + 1. ***Phát hiện bằng việc giải mã tĩnh***

Kỹ thuật này sử dụng việc giải mã tĩnh [4] trong một mã độc cụ thể sử dụng tất cả các cách trong toàn bộ phần mã của nó. Như vậy tốc độ quét phục thuộc nhiều vào kích thước của phần mã chương trình được quét. Kỹ thuật này tương đối nhanh nếu kết hợp với kỹ thuật lọc. Tuy nhiên nó cũng dễ gấy ra các thông báo sai và không đảm bảo với những mã độc khi chạy mà không cần giải mã.

* + 1. ***Mã giả lập***

Là kỹ thuật rất mạnh mẽ trong việc phát hiện mã độc. Một máy ảo được mô phỏng các hệ thống CPU và bộ nhớ để bắt trước các thực thi mã. Do đó mã độc được mô phỏng trong các máy ảo quét và không có mã của mã độc thực tế nào thực hiện bởi bộ vi xử lý.

Công cụ mã giả lập:

* IDA pro [5] là bộ đa xử lý dung để disassembler và debug cũng cấp các tính năng.
* Ollydebug [6] là chương trình debug mạnh mẽ giúp cho ta có thể trace từng dòng code, theo dõi từng hành vi của mã độc.

Phương pháp đầu tiên của mã giả lập là sử dụng trình debug để theo dõi các mã sử dụng bộ vi xử lý. Tuy nhiên giải pháp này không đủ an toàn do mã các mã độc có thể nhảy ra ngoài nơi mô phỏng trong môi trường phân tích với việc sử dụng các kỹ thuật antidebug hay anti disassembly hoặc anti máy ảo.

Dưới đây là một ví dụ về các thanh ghi và cơ 16 bit định nghĩa với các cấu trúc trong ngôn ngữ C:

*Typedef struct*

*{ byte ah,al,bh,bl,ch,cl,dh,dl;*

*word si,di,sp,bp,cs,ds,es,ss,ip;*

*} Emulator\_Registers\_t;*

*typedef struct {*

*byte c,z,p,s,o,d,i,t,a;*

*} Emulator\_Flags\_t;*

## Quy trình phân tích mã độc

Đầu tiên ta sẽ tìm hiểu về các phương pháp phân tích mã độc từ đó sẽ đưa ra quy trình phân tích mã độc.

* + 1. ***Các phương pháp phân tích mã độc***

Phương pháp phân tích thì có 2 phương pháp chính :

* + - * Dynamic analysis [7] (phân tích hành vi mã độc).
      * Static analysis [7] (phân tích bằng cách xem mã dịch ngược).

Tuy nhiên không phải mã độc nào cũng phân tích theo hai phương pháp trên, bởi vì có những mã độc có thể chỉ cần phân tích động hoặc có những loại chỉ cần phân tích tĩnh. Vì vậy cần phải có cách thức phân tích riêng của mỗi người dựa trên kinh nghiệm và thực tế phân tích để tiến hành tránh mất thời gian cũng như công sức bỏ ra. Chính vì thế hai phương pháp trên chỉ mang tính tương đối.

* + - 1. *Static analysis*

Đối với phương pháp này, ta sẽ tiến hành xem code để xem thực sự mã độc này sẽ làm gì trên hệ thống. Không cần phải chạy mã độc đó, và hiểu rõ hơn thực sự về hoạt động của mã độc. Các công cụ thường được sử dụng trong quá trình phân tích tĩnh có thể kể đến các chương trình dịch ngược :

* + - * + Disassembler
        + Decompiler
        + IDA
        + OnlyDbg
        + Source Code Analyzer

Ưu điểm :

Static analysis là có thể phát hiện ra hoạt động, cách ứng xử của chương trình trong những điều kiện không tồn tại thực tế.

Static analysis sẽ cho chúng ta cái nhìn rất đúng nhất về một chương trình. Sở dĩ gọi là gần đúng bởi rất hiếm khi chúng ta có thể nắm được toàn bộ hoạt động của một chương trình, chỉ khi các chương trình đủ nhỏ.

Nhược điểm :

Phương pháp này đòi hỏi người phân tích phải am hiểu sâu về hệ thống và lập trình.

Phương pháp này đôi khi rất mất thời gian thậm chi có những loại mã độc mà thời gian phân tích của nó tính bằng tháng.

* + - 1. *Dynamic analysis*

Dynamic analysis thì là quan sát xem mã độc khi thực thi thì sẽ làm những gì, nó chạy ra sao, làm gì trên máy tính mình qua các công cụ monitor, cách này thì có nhược điểm với các dòng mã độc chạy lịch trình. Nghiên cứu hoạt động của chương trình bằng cách thực thi chương trình đó. Các công cụ được sử dụng trong trường hợp này có thể kể đến như :

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

*Ưu điểm :*

Dynamic analysis thì nhanh và thông tin chính xác. Tuy nhiên dynamic analysis có một khuyết điểm là: "Tất cả những thấy không phải là tất cả những gì chương trình có". 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 chạy theo thời gian, tức là ở thời điểm này mình cho nó chạy nó không có hoạt động gì, nhưng ở một thời điểm khác nói lại chạy.

Mỗi phương pháp đều có một đặc điểm riên nhưng khi phân tích mã độc 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.

* + 1. ***Các bước cơ bản phân tích mã độc***
       1. *Phân tích thông tin sơ bộ*

Đầu tiên nên nhìn nhận dấu hiệu trên máy bị nhiễm có những hiện tượng gì ? Máy chậm hay lưu lượng mạng tăng, từ đó kiểm tra xem có các processes lạ trên máy bị nhiễm không ? Có những bất thường xảy ra trên máy tính đó không?... Sau đó là thu thập mẫu nghi ngờ là mã độc.

Nhận diện hệ thống bị nhiễm mã độc:

* + - * + 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 processes, các dịch vụ, các drivers, các add-on trình duyệt, phiên bản hệ điều hành… của hệ thống. Mục đích thu thập thông tin này để sau này còn có thể xác định nguồn gốc lây lan và tại sao lại bị nhiễm thông qua đâu. Đồng thời cũng ra soát lại một lượt xem hệ thống mình có dính lỗi bảo mật nào không.

Dự đoán đặc điểm, phân loại mã độc, chia dòng, đặt tên. Bằng cách xem mã hex, xem properties của files, kích thước file… hoặc thông tin đơn giản khác để chúng ta tiến hành phân loại sơ bộ mã độc.

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

* + - 1. *Quan sát hành vi mã độc (Dynamic analysis)*

Thiết lập môi trường thử nghiệm: Sử dụng máy ảo, sandbox, vitural PC để cho mã độc chạy. Nếu mã độc hại có cơ chế phát hiện môi trường ảo có thể 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 lỗi nếu mã độc có khai thác lỗi... Môi trường thử nghiệm càng đầy đủ, càng "thật" thì sẽ quan sát được càng nhiều đặc điểm của mã độc. Nếu xây dựng được hệ thống máy như máy bị nhiễm thì là tốt nhất.

Thiết lập các công cụ giám sát: Vì không thể monitor virus bằng mắt nên trong môi trường thử nghiệm cần có các công cụ monitor 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 tốt hơn. Bắt đầu chạy mã độc và quan sát các thông tin :

* + - * + Khảo sát processes xem có processes lạ nào đáng ngờ không ?
        + Khảo sát các modules dlls 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 modules (files) đang chạy trên hệ thống xem có bị mã độc gắn vào không ?
        + Khảo sát các files, các folder ẩn để tìm các file nghi ngờ.
        + Khảo sát các driver, tìm rootkit làm ẩn tiến trình, ẩn key.
        + Khảo sát lưu lượng mạng (Dùng Dumetter) xem có tăng đột ngột không?
        + Kiểm tra xem các kết nối TCP/IP trên máy.

Chạy mã độc và ghi log, quan sát càng lâu kết quả càng chính xác. Thông thường mã độc 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.

Phân loại virus và ghi nhận các đặc điểm: Từ các thông tin trên chúng ta sẽ tiến hành phân loại virus: là worm hay trojan, backdoor, virus lây file... có rootkit hay không.

* + 1. ***Phân tích mã dịch ngược ( Static Analynis)***

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

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

* + - * + Xem mã độc được viết bằng ngôn ngữ gì ?
        + Sử dụng các packer để nén lại hay các protecter nào không ?
        + Khi biết mã độc được pack bằng trình nào thì có thể dùng trình đó unpack ra, tương tự cũng unprotecter file mã đ ộc hại.
        + Dùng các công phụ phù hợp với công cụ viết mã độc để decompile, disassemble, debugging.
        + Đọc và phân tích mã code assembly để tìm thêm hoạt động của mã độc.
    1. ***Trace Code Debug***

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. Có nghĩa là sẽ lần từng lệnh nhỏ của mã độc để xem chính xác là mã độc làm những gì. Mã độc thường có các kỹ thuật chống dịch ngược thì cũng sẽ có các kỹ thuận chống debug(anti debug) và người phân tích phải vượt qua nó (anti anti-debug).

* + 1. ***Tìm signature của mã độc và đưa vào cơ sở dữ liệu***

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

Từ những đặc điểm nhận dạng trên sẽ đưa vào cơ sở dữ liệu.

# CHƯƠNG 2.

# CÁC PHƯƠNG PHÁP TẠO MẪU MÃ ĐỘC

Để xây dựng một cơ sở dữ liệu mã độc trước tiên cần phải có kiến thức về việc phát hiện mã độc bằng cách nào ở chương một, các kỹ thuật thu thập mã và phát hiện mã độc. Chương này sẽ trình bày thêm về các dấu hiệu phát hiện mã độc, chuẩn trao đổi mã độc đang được sử dụng hiện nay.

## Nghiên cứu chuẩn trao đổi dữ liệu mã độc

Hiện nay do việc tìm và diệt mã độc càng ngày càng trở lên cấp thiết đối với không chỉ người dung mà cả đối với an ninh quốc gia nên việc chia sẻ cơ sở dữ liệu mã độc giữa các hang hay các tổ chức, quốc gia là cần thiết. Chính vì vậy một chuẩn trao đổi dữ liệu đã được ra đời.

Chuẩn trao đổi dữ liệu mã độc tạo ra bởi ICSG Malware Working Group. ICSG là viết tắt của Industry Connection Security Group [8], nó ra đời nhằm thúc đẩy việc hợp tác và chia sẻ thông tin trong ngành bảo mật.Mục tiêu của Malware Working Group là giải quyết các vấn đề về phần mềm độc hại mà ngành công nghệ thông tin hiện nay phải đối mặt.

Trọng tâm ban đầu có mục đích là thành lập cách thông minh hơn để chia sẻ các mẫu mã độc và các thông tin liên quan tới chúng trong ngành bảo mật máy tính đạt hiệu quả cao hơn.

Các nhóm làm việc giải quyết các vấn đề của việc đóng gói mã độc.

* + - Các tài liệu phục vụ tốt nhất trong việc sử dụng được đống gói bởi các nhà phát triển phần mềm.
    - Xác định các thuộc tính của việc đống gói tập trung vào các tính năng mà mã độc thường sử dụng.
    - Tạo một khóa đăng ký của việc đóng gói và một tập hợp các tên các việc đống gói đó.
    - Thiết lập một định dạng chia sẻ dữ liệu để chia sẻ thông tin đóng gói.
    - Phát triển và thực hiện “Hệ thống Taggant” – nhúng một mã băm mạnh vào từng đối tượng để đóng gói để phát hiện ra nguồn các tập tin đóng gói. Taggant là một dấu hiệu được bổ sung trong quá trình tạo ra sản phẩm. Một khi thư viện taggant được thực hiện các nhóm sẽ thúc đẩy và tạo điều kiện giám sát việc triển khai nó đối với cả bên phát triển Anti Virus và bên cung cấp mẫu đóng gói.

Để tham gia nhóm Malware Working Group thì các tổ chức cá nhân cần trở thành thành viên của ICSG. Chỉ có các thành viên ICSG mới có quyền tham gia trao đổi đưa ra các ý tưởng giải pháp trong việc đưa ra chuẩn. Tuy nhiên cũng có những chuyên gia được mời nhưng họ chỉ tham gia việc phát triển mà không có quyền quyết định.

Dưới đây là dạng của chuẩn trao đổi mã độc:

<malwareMetaData [xmlns="http://xml/metadataSharing.xsd](http://xml/metadataSharing.xsd)" xmlns:xsi="[http://www.](http://www/) w3.org/2001/XMLSchema- instance"xsi:schemaLocation="<http://xml/metadataSharing.xsd> file:metadataSharing.xsd" version="1.200000" id="10000">

<company>N/A</company>

<author>MMDEF Generation Script</author>

<comment>Test MMDEF v1.2 file generated using genMMDEF</comment>

<timestamp>2011-08-19T13:50:21.721000</timestamp>

<objects>

<file id="c7ae4ffe33fc841aea2e0113afa05fdf">

<md5>c7ae4ffe33fc841aea2e0113afa05fdf</md5>

<sha1>25daac9d19f18b5ac19769dcf7e5abc154768641</sha1>

<sha256>e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495 991b7852b855</sha256>

<sha512> cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36c e9ce47d0d13c5d85f2b0ff8318d2877eec2f63b931bd47417a81a538327a f927da3e

</sha512>

<size>1546790</size>

<filename>ProcessExplorer.zip</filename>

<MIMEType>application/zip</MIMEType>

</file>

<file id="d22ff2cc70fa2eec94aaa6c6f49e6eb0">

<md5>d22ff2cc70fa2eec94aaa6c6f49e6eb0</md5>

<sha1>2458a3d696698e2c4550b91e54ff63f4b964198d</sha1>

<sha256> 6ff22c87fb5ee105b33346dbb3f13f3049a292981e9df1eb8591e858ccf4 d782

</sha256>

<sha512> 34e18bf9679c71189383bcd89c9f723383715bbf63f051edd659c57e14d0 12987c33ba67fbbb0faeca962b3ec7b12b0aa24b3c134ddbb9f905aa2660 4718f375

</sha512>

<size>7005</size>

<crc32>1185414000</crc32>

<filename>Eula.txt</filename>

<filenameWithinInstaller>Eula.txt</filenameWithinInstaller>

<MIMEType>text/plain</MIMEType>

</file>

<file id="ae846553d77284da53abcd454b4eaedf">

<md5>ae846553d77284da53abcd454b4eaedf</md5>

<sha1>782333340d56a8f020a74bd2830e68f31310a5b7</sha1>

<sha256>

a8f4bd956816960691bc08bf94be342a6d62bf6d91c92f7e7506903ffda5 0b83

</sha256>

<sha512> bf1fd7b27234d5605731d21358bba01738098cb363a6deee79ed88699a39 27b3a09d9044767db24ff6ddd028fbe0f4a1572f9af4d4ab4996bfdefe2b 950a9b49

</sha512>

<size>72268</size>

<crc32>2807815698</crc32>

<filename>procexp.chm</filename>

<filenameWithinInsưtaller>procexp.chm</filenameWithinInstalle r>

</file>

<file id="4edc50d3a427566d6390ca76f389be80">

<md5>4edc50d3a427566d6390ca76f389be80</md5>

<sha1>9cb1bd5dc93124f526a1033b1b3f37cc0224a77e</sha1>

<sha256> e942d28c0e835b8384752731f1b430cb3fbd571381666ded7637a2db47fa fcc0

</sha256>

<sha512> 3ceb1bd07af9e470ff453ef3dd4b97f9228856cb78eb5cddb7b81796b4b8 30368e3ed2f0c6a9ce93009397e8158c68dba67e398f58df87137d8872cb 0bb3b53b

</sha512>

<size>3412856</size>

<crc32>1119775926</crc32>

<filename>procexp.exe</filename>

<filenameWithinInstaller>procexp.exe</filenameWithinInstalle r>

<MIMEType>application/octet-stream</MIMEType>

</file>

<softwarePackage id="procexp">

<vendor>Sysinternals</vendor>

<product>Process Explorer</product>

<version>14.11</version>

<language>English</language>

</softwarePackage>

</objects>

<relationships>

<relationship type="createdBy" id="1">

<source>

<ref>file[@id="c7ae4ffe33fc841aea2e0113afa05fdf"]</ref

>

</source>

<target>

<ref>file[@id="d22ff2cc70fa2eec94aaa6c6f49e6eb0"]</ref

>

<ref>file[@id="ae846553d77284da53abcd454b4eaedf"]</ref

>

<ref>file[@id="4edc50d3a427566d6390ca76f389be80"]</ref

>

</target>

<timestamp>2011-08-19T13:50:21.924000</timestamp>

</relationship>

<relationship type="partOfPackage" id="2">

<source>

<ref>softwarePackage[@id="procexp"]</ref>

</source>

<target>

<ref>file[@id="d22ff2cc70fa2eec94aaa6c6f49e6eb0"]</ref

>

<ref>file[@id="ae846553d77284da53abcd454b4eaedf"]</ref

>

<ref>file[@id="4edc50d3a427566d6390ca76f389be80"]</ref

>

<ref>file[@id="c7ae4ffe33fc841aea2e0113afa05fdf"]</ref

>

</target>

<timestamp>2011-08-19T15:50:21.924000</timestamp>

</relationship>

</relationships>

</malwareMetaData>

Chuẩn trao đổi được lưu dưới dạng file XML. XML viết tắt của Extensible Markup Language (ngôn ngữ đánh dấu mở rộng) là một ngôn ngữ đánh dấu sử dụng để tạo thẻ riêng cho người sử dụng. Nó được tạo bởi liên minh mạng toàn cầu nhằm khắc phục những hạn chế của HTML. File dữ liệu chuẩn trao đổi có thuộc tính cha là “malwareMetaData” và nó có những thuộc tính con: company, author, objects, relationships… và chúng lại chứa các thuộc tính con khác nữa như: name, md5, sha1, sha256 …Vì thời lượng và kiến thức nên chương trình xây dựng cơ sở dữ liệu mẫu mã độc của em sẽ thực hiện với một số các thuộc tính trên mà sẽ không dùng hết các thuộc tính.

## Nghiên cứu cơ sở dữ liệu mã độc Clam AV từ đó xây dựng cơ sở dữ liệu mẫu

* + 1. ***Clam Anti Virus***

Clam Anti Virus [9]- viết tắt là là CLAMAV - là một phần mềm diệt virus mã nguồn mở được thiết kế để phát hiện các loại mã độc như: trojan, virus,… và các mối đe dọa khác. Nó có chuẩn cho việc quét thư điện tử. Nó cung cấp một hiệu suất quét đa luồng cao, tiện ích dòng lệnh cho viêc quét file theo yêu cầu và một công cụ thông minh để cập nhật chữ ký tự động. Cốt lõi của CLAMAV là cung cấp nhiều cơ chế định dạng tập tin phát hiện, hỗ trợ giải nén tập tin, hỗ trợ lưu trữ và có nhiều dạng kiểu chữ ký để phát hiện các mối đe dọa. Cốt lõi thư viện của CLAMAV được sử dụng trong Immunet 3.0, được hỗ trợ bởi CLAMAV. Nó xử lý nhanh, đầy đủ các tính năng là giải pháp cho máy tinh sử dụng Windows.

* + 1. ***ClamAV Virus Databases***

CLAMAV Virus Databases - viết tắt là CVD – là nơi chứa các chữ ký số bao gồm cơ sở dữ liệu chữ ký trong các định dạng văn bản khác nhau. Các tiêu đề của các nơi chứa này là chuỗi ký tự có độ dài 512 byte với các trường riêng biệt :

*ClamAV-VDB:build time:version:number of signatures:functionality level required:MD5 checksum:digital signature:builder name:build time (sec)*

*sigtool –info* hiện thị thông tin chi tiết tệp tin trong CVD*: zolw@localhost:/usr/local/share/clamav$ sigtool -i main.cvd File: main.cvd*

*Build time: 09 Dec 2007 15:50 +0000*

*Version: 45*

*Signatures: 169676*

*Functionality level: 21 Builder: sven*

*MD5: b35429d8d5d60368eea9630062f7c75a*

*Digital signature: dxsusO/HWP3/GAA7VuZpxYwVsE9b+tCk+tPN6OyjVF/U8*

*JVh4vYmW8mZ62ZHYMlM903TMZFg5hZIxcjQB3SX0TapdF1SFNzoWjsyH53eXv MDYeaPVNe2ccXLfEegoda4xU2TezbGfbSEGoU1qolyQYLX674sNA2Ni6l6/ CEKYYh Verification OK.*

Project ClamAv phân phối dưới các tập tin của file CVD gồm: main.cdv và daily.cvd

* + 1. ***Debug thông tin từ libclamav***

Để tạo ra chữ ký có hiệu quả cho CLAMAV, điều quan trọng nhất là làm thế nào hiểu được công cụ xử lý tập tin đầu vào. Cách tốt nhất để xem nó hoạt dộng được hay không là có một cái nhìn về các thông tin debug libclamav. Chúng ta có thể làm việc này bằng cách gọi clamscan với tùy chọn –debug và –leavee-temps. Tùy chọn đầu tiên làm cho clamscan hiển thị các thông tin từ libclamav và tùy chọn thứ hai giúp tránh việc xóa tập tin tạm thời để có thể phân tích tiếp.

Như vậy không có một tập tin nào được tạo ra từ libclamav. Việc viết một chữ ký cho các tập tin giải nén ta có nhiều co hội mà các công cụ có thể phát hiện ra dữ liệu khi nó được nén bởi các bên đóng gói khác.

Phương pháp này nên áp dụng cho tất cả các tập tin muốn tạo chữ ký. Bằng cách debug cá thông tin ta có thể nhanh chóng xem các công cụ ghi nhận và thực hiện dữ liệu và những tập tin khác được tạo ra.Chữ ký được tạo ra cho các tập tin tạm thời ở cấp dưới thường chung chung hơn và việc phát hiện mã độc tương tự nhau trong với các hình thức khác nhau.

* + 1. ***Định dạng chữ ký của CLAMAV***
       1. *MD5*

Cách dễ nhất để tạo chữ ký cho ClamAV là sử dụng mã checksum MD5 [10], tuy nhiên phương pháp này chỉ có thể sử dụng cho mã độc tĩnh. Tạo mã MD5:

File *.hdb*: chứa các giá trị hash MD5 của các mẫu virus đã phát hiện được

Đây là ví dụ một mẫu trong file *main.hdb*

Giá trị hash MD5

Tên virus

2401851daa0343df8ff683f730fec39:92281:Dialer-85

Kích thước file

Để tạo một mẫu virus MD5 cho một file bất kì, ta dùng chương trình *sigtool* như sau:

s*igtool –md5 file*

Ví dụ*:*

*sigtool --md5 test.exe*

ta được kết quả:

*48c4533230e1ae1c118c741c0db19dfb:17387:test.exe*

* + - 1. *Tạo MD5 cho một phần PE file*

CLAMAV hỗ trợ việc tạo chữ ký MD5 cho một phần cụ thể của PE file [10]. Chữ ký này được lưu trữ trong file mdb với định dạng sau:

*PESectionSize:MD5:MalwareName*

File *.mdb*: chứa giá trị hash MD5 của PE section trong một file nhiễm virus.

Mẫu virus này có dạng:

Tên virus

Kích thước section

7168:a105e2cc8148158cd048360eb847c7d0:Trojan.Downloader-1421

Giá trị hash MD5

Cách dễ nhất để tạo mẫu virus PE section này là tách các sections của file ra từng file rồi biệt rồi dùng trình *sigtool*  với tùy chọn chọn là *–mdb*.

* + - 1. *Định dạng hexa*

File *.db*: chứa nội dung của mẫu virus biểu diễn dưới dạng chuỗi hex [10].

Mẫu virus này có dạng:

Biểu diễn hex của file

Tên virus

\_0017\_0001\_000=21b8004233c999cd218bd6b90300b440cd218b4c198b541bb80157cd

Mẫu virus này được tạo bằng cách dùng lệnh như sau:

*cat virusfile | sigtool –hex-dump*

* + - 1. *Ký tự đại diện*

ClamAv hỗ trợ các phần mở rộng dưới đây cho chữ ký hex:

?? : Phù hợp với bất kỳ byte nào

a?: Phù hợp với một nibble cao (bốn bít cao)

?a: phù hợp với nibble thấp ( bốn bít thấp)

\*: Phù hợp với bất kỳ số lượng byte

{}: Phù hợp với n byte

{-n}: phù hợp với nhỏ hơn hoặc bằng n byte

{n-}: phù hợp với lớn hơn hoặc bằng n byte

{n-m}: phù hợp n-m byte với n>m

(aa||bb||cc||…): phù hợp với aa hoặc bb hoặc cc ...

! (aa||bb||cc||…): phù hợp với bất kỳ byte nào trừ aa, bb,cc,…

HEXSIG[x-y] aa or aa[x-y]HEXSIG : phù hợp với aa gắn vào một chữ ký hex

(B): phù hợp với từ (bao gồm tập tin giới hạn)

(L): phù hợp với CR, CRLF hoặc tập tin giới hạn

Chữ ký phạm vi \* và {} hầu như tách một hex thành hai phần.

* + - * 1. Định dạng chữ ký cơ bản

Ví dụ về định dạng chữ ký :

*MalwareName=HexSignature*

CLAMAV sẽ quét tất cả tập tin để tìm chữ ký hex. Tất cả chữ ký kiểu này được lưu dưới dạng file .db.

* + - * 1. Định dạng chữ ký mở rộng

Định dạng chữ ký mở rộng cho phép bổ sung thông tin đặc điểm kỹ thuật như một loại tập tin mục tiêu, offset của virus hoặc phiên bản công cụ, giúp phát hiện tốt hơn. Định dạng như sau:

*MalwareName:TargetType:Offset:HexSignature[:MinFL:[MaxFL]]*

TargetType có giá trị là một trong các số sau đại diện cho kiểu file cụ thế:

0: bất kỳ file nào

1:file thực thi di động đối với cả 32 bit và 64 bit

2: file nằm trong OLE2: file ảnh, VBscript …

3: HTML

4: Tập tin mail

5: đồ họa

6: ELF

7: file text mã ASCII

8: không sử dụng

9: File Mach – O

Offset là một dấu hoặc số thập phân n có thể kết hợp với việc thay đổi đặc biệt:

\* : bất kỳ

n: offset tuyệt đối

EOF – n: kết thúc file trừ đi n byte

Chữ ký cho PE,ELF, Mach-O file hỗ trợ:

EP + n:entry point cộng n byte

EP – n: entry point trừ n byte

Sx + n: bắt đầu từ section x của dữ liệu cộng n byte

Sx – n: bắt đầu từ section x của dữ liệu trừ n byte

SL + n: kết thúc của section của dữ liệu cộng n byte

SL – n: kết thúc của section của dữ liệu trừ n byte

* + - 1. *Chữ ký biểu tượng cho PE file*

ClamAV 0.96 có một khoảng biểu tượng mờ để giúp phát hiện các tập tin thực thi mã độc đã ngụy trang bằng cách tìm kiếm các tập tin hình ảnh, office, văn bản, hay tương tự thế.

Biểu tượng phù hợp chỉ được thông qua chữ ký .ldb bằng cách sử dụng thuộc tính đặc biệt thẻ IconGroup1 hoặc IconGroup2. Định danh hai nhóm các biểu tượng được lưu trong tập tin cơ sở dữ liệu .ldb. Định dạng của tập tin .ldb là :

*ICONNAME:GROUP1:GROUP2:ICON\_HASH*

* ICON\_NAME llà một chuỗi định dannh duy nhất cho biểu tượng cụ thể
* Group1 là một chuỗi từ định danh nhóm đầu tiên của các biểu tượng (Icongroup1)
* GROUP2 là một chuỗi định danh nhóm thứ hai của các biểu tượng (IconGroup2)
* ICO\_HASH là một mã băm mò của hình ảnh biểu tượng
  + - 1. *Chữ ký cho các siêu dữ liệu thông tin phiên bản trong PE file*

Từ phiên bản CLAMAV 0.96, nó đã dễ dàng kết hợp một số thông tin xây dựng vào các PE file (gồm file thực thi và thư viện lien kết động). Bất kỳ khi nào tra cứu thuộc tính của một tập tin thực thi PE trong windows, chúng ta sẽ thấy một loạt các chi tiết đó.

Những thông tin được lưu trữ trong một khu vực đặc biệt của nguồn tập tin kèm theo tên của VS\_VERSION\_INFOMATION (hoặc versioninfo). Nó được chia làm hai phần. Phần đầu tiên là một loạt cá số và cờ cho ta thấy phiên bản tập tin. Lúc đầu nó được sử dụng với các trình cài đặt sau khi phân tích nó sẽ có thể xác định xem việc thực thi hoặc thư viện nhất định sẽ được nâng cấp, ghi đè hoặc phát triển lên. Phương pháp này không bao giờ thực sự được sử dụng.

Phần thứ hai là một danh sách đơn giản của khóa, chuỗi giá trị.Nó dành cho thông tin người dùng và hoàn toàn bỏ qua bởi hệ diều hành. Ví dụ khi nhìn vào ping.exe ta sẽ thấy công ty là “Microsoft Corpo-ration”, phần mô tả là “TCP/IP Ping command”,tên nội bộ là “ping.exe”…. Tùy thuộc vào phiên bản hệ điều hành, một số khóa sẽ có thể hiển thị thông tin đặc biệt trong hộp thoại thuộc tính, tuy tên nội bộ là như nhau.

Để phù hợp với một cặp khóa/giá trị thông tin phiên bản các offset đặc biệt của file gắn VI đã được đưa ra. Điều này tương tự như cá kiểu gắn khác ngoài trừ việc thay đổi để phù hợp với mẫu hex lặp lại của một offset riêng biệt, nó sẽ kiểm tra chính nó và mỗi cặp khóa/giá trị trong tập tin.Thẻ VI không cần và cũng không chấp nhận một +/- offset giống như các ví dụ EP +1. Đối với các chữ ký hex của chính nó, nó chỉ là UTF 16 kết xuất của khóa và giá trị. Chỉ ký tự đại diện ?? và (aa||bb) được cho phép trong chữ ký.Thông thường chúng ta không cần phải bận tâm tìm ra đáp án: bởi vì mỗi cặp khóa/giá trị cung với chữ ký VI dựa trên tương ứng được in bằng clamscan khi dùng tùy chọn –debug.

Ví dụ: *clamscan --debug freecell.exe*

*[...]*

*Recognized MS-EXE/DLL file in cli\_peheader*

*versioninfo\_cb: type: 10, name: 1, lang: 410, rva: 9608 cli\_peheader: parsing version info @ rva 9608 (1/1)*

*VersionInfo (d2de): ’CompanyName’=’Microsoft Corporation’ -VI:43006f006d00700061006e0079004e0061006d006500000000004d006900630072006f0073006f0066007400200043006f00720070006f0072006100740069006f006e000000*

*VersionInfo (d32a): ’FileDescription’=’Entertainment Pack*

*FreeCell Game’ - VI:460069006c0065004400650073006300720069007000740069006f006e000000000045006e007400650072007400610069006e006d0065006e00740020005000610063006b0020004600720065006500430065006 c006c002000470061006d0065000000*

*VersionInfo (d396): ’FileVersion’=’5.1.2600.0 (xpclient.010817*

*-1148)’ - VI:460069006c006500560065007200730069006f006e00000000 0035002e0031002e0032003600300030002e003000200028007800700063006c00690065006e0074002e003000310030003800310037002d00310031003400380029000000*

*VersionInfo (d3fa): ’InternalName’=’freecell’ - VI:49006e007400 650072006e0061006c004e0061006d00650000006600720065006500630065006c006c000000*

*VersionInfo (d4ba): ’OriginalFilename’=’freecell’ - VI:4f007200 6900670069006e0061006c00460069006c0065006e0061006d00650000006600720065006500630065006c006c000000*

*VersionInfo (d4f6): ’ProductName’=’Sistema operativo Microsoft*

*Windows’ - VI:500072006f0064007500630074004e0061006d0065000000000053006900730074065006d00610020006f0070006500720061007400690076006f0020004d006900630072006f0073006f0066007400ae002000570069006e0064006f0077007300ae000000*

*VersionInfo (d562): ’ProductVersion’=’5.1.2600.0’ - VI:50007200 6f006400750063007400560065007200730069006f006e00000035002e0031002e0032003600300030002e0030000000 [...]*

Mặc dù chữ ký VI ở trên được sử dụng trong chữ ký logic nhưng ta vẫn có thể thử dung chúng bình thường trong file .ndb:

*my\_test\_vi\_sig:1:VI:paste\_your\_hex\_sig\_here*

Nếu muốn giải mã một chữ ký VI cơ bản ta dùng :

*echo hex\_string | xxd -r -p | strings –el*

* + - 1. *Chữ ký dựa trên siêu dữ liệu*

CLAMAV 0.96 cho phép tạo chữ ký chung phù hợp với các tập tin được lưu

trữ trong các kiểu chứa khác nhau đáp ứng các điều kiện cụ thể. Định dạng như sau: *VirusName:ContainerType:ContainerSize:FileNameREGEX: FileSizeInContainer:FileSizeReal:IsEncrypted:FilePos: Res1:Res2[:MinFL[:MaxFL]]*

* VirusName: Tên của virus hiển thị khi khớp chữ ký
* ContainerType gồm các định dạng CL\_TYPE\_ZIP, CL\_TYPE\_RAR, CL\_TYPE\_ARJ,CL\_TYPE\_CAB,CL\_TYPE\_7Z, CL\_TYPE\_MAIL,CL\_TYPE\_(POSIX|OLD)\_TAR, CL\_TYPE\_CPIO\_(OLD|ODC|NEWC|CRC) hoặc \*
* ContainerSize: Kích thước nơi chứa tập tin
* FileNameREGEX: Mô tả tên file đích
* FileSizeInContainer: thường được nén kích thước,
* FileSizeReal: thường không nẽ kích thước, MAIL,TAR và CPIO thì

ngược lại

* IsEncrypted: 1nếu file mã hóa, 0, nếu không mã hoá, \* để bỏ qua
* FilePoss: Vị trí file trong nới chứa
* Res1: khi ContainerType là CL\_TYPE\_ZIP or CL\_TYPE\_RAR
* Res2: không sử dụng với CLAMAV 0.96
* Chữ ký được lưu trữ trong file .cdb.
  + - 1. *Chữ ký trên siêu dữ liệu ZIP/RAR*

Nó chỉ được sử dụng lưu trữ chữ ký đối với các định dạng Zip hoặc Rar và có định dạng:

*virname:encrypted:filename:normal size:csize:crc32:cmethod: fileno:max depth*

* Virname: Tên virus
* Encrypted: cờ mã hóa, 1- mã hóa , 0 – không mã hóa
* Filename: Tên file
* Normal size : kích thước file không nén (\* để bỏ qua)
* Compressed size: kích thước file khi nén(\* để bỏ qua)
* CRC32(\* để bỏ qua)
* Cmethod: phương thức nén (\* để bỏ qua)
* Fileno: vị trí file trong kho lưu trữ (\* để bỏ qua)
* Max depthSố lượng tối đa dữ liệu lưu trữ chồng nhau(\* để bỏ qua)

Cơ sở dữ liệu này được xuất dưới dạng file .zmd hoặc .rmd

* + - 1. *Danh sách trắng*

Danh sách trắng là một tập tin cụ thể sử dụng chữ ký MD5 và đặt trong một tập tin cớ sở dữ liệu có định dạng .fb

Để danh sách một chữ ký cụ thể từ cơ sở dữ liệu ta chỉ cần thêm tên của nó vào một tập tin cục bộ gọi là local.ign2 được lưu trữ trong thư mục cơ sở dữ liệu. Ta cũng có thể thực hiện theo tên chữ ký với MD5 của phần nhập toàn bộ cơ sở dữ liệu cho chữ ký .Ví dụ:

*Eicar-Test-Signature:bc356bae4c42f19a3de16e333ba3569c*

* + - 1. *Danh sách chữ ký theo tên*

CLAMAV sử dụng các tiền tố dưới đây làm tên chữ ký:

* Worm: Sâu mạng
* Trojan: Chương trình cửa hậu
* Adware: adware
* Flooder: flooder
* HTML: tập tin HTML
* Email: Thư điện tử
* IRC: trojan IRC
* JS: Mã độc Java Script
* PHP: Mã độc PHP
* ASP: Mã độc ASP
* VBS: Mã độc VBS
* BAT: Mã độc BAT
* W97M,W2000M:virus macro trên word
* X97M, X2000M: virus macro trên excel
* O97M,O200M: virus macro chung trên office
* DoS: phân mềm tấn công từ chối dịch vụ
* DOS: mã độc DOS trước đây
* Exploit: lỗ hổng bảo mật phổ biến
* VirTool: Công cụ tạo virus
* Dialer: dialer
* Joke: lừa đảo

Quy tắc đặc tên của CLAMAV:

* Luôn sử dụng một hậu tố zippwd vào tên mã độc cho loại ZMD
* Luôn sử dụng một hậu tố rarpwd vào tên mã độc cho loại RMD
* Chỉ sự dụng các ký tự chữ và số, dấu gạch ngang “-“, dấu chấm “.”, dấu gạch dưới “\_” trong tên mã độc không bao giờ sử dụng dấu đơn, kép, khoảng trống.
  + - 1. *File đặc biệt*
         1. HTML

CLAMAV có chứa một mã HTML đặc biệt bình thường, giúp phát hiện lỗ hổng HTML. Chạy sigtool –html chuẩn hóa trên một tập tin HTML sẽ tạo ra các tập tin dưới đây:

Nocmomment.html: tập tin bình thường, với trường hợp thấp hơn tất cả các nhận xét và khoảng trắng bị xóa.

Notags.html: giống như trên nhưng các thẻ HTML bị xóa.

Cần tạo ra một chữ ký đối với các tập tin được tạo ra. Để loại bỏ khả năng cảnh báo giả các mục tiếu cần được đặt giá trị là 3.

* + - * 1. Tập tin văn bản

Cũng giống như HTML các tập tin văn bản mã ASCII được bình thường hóa tất cả các khoảng trống và kiểm soát các ký tự bị xóa trước khi quét.

Câu lệnh : clamscan – leave-temp để có một tập tin bình thường và sau đó tạo ra chữ ký với đích có kiểu là 7

* + - * 1. Tập tin thực thi di động đã nén

Nếu tập tin được nén bởi UPX, FSG, Petite hoặc một trình đống gói PE file khác được hỗ trợ bởi libclamav, cần chạy clamscan với tùy chọn --debug --leave- temps. Ví dụ xuất ra từ trình nén file FSG:

*LibClamAV debug: UPX/FSG/MEW: empty section found - assuming compression LibClamAV debug: FSG: found old EP @119e0*

*LibClamAV debug: FSG: Unpacked and rebuilt executable saved in*

*/tmp/clamav-f592b20f9329ac1c91f0e12137bcce6c*

Tiếp theo tạo một chữ ký từ :

*/tmp/clamav-f592b20f9329ac1c91f0e12137bcce6c*

# CHƯƠNG 3.

# XÂY DỰNG CƠ SỞ DỮ LIỆU MẪU MÃ ĐỘC

Chương cuối của bản thực tập sẽ xây dựng chương trình tạo mẫu mã độc và từ đó xây dựng cơ sở dữ liệu mẫu mã độc.

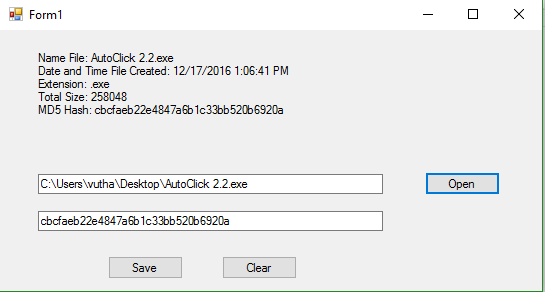
## 3.1 Xây dựng chương trình tạo mẫu mã độc

Xây dựng một chương trình tạo mẫu mã độc giúp cho việc lưu lại dữ liệu mã độc một cách khoa học, thuận tiện và theo quy tắc nhất định.

Chương trình được viết bằng ngôn ngữ C# dựa trên nền tảng .Net với Net Framework 4.5 và được biên dịch bằng Visual Studio 2012.

Cơ sở dữ liệu mã độc gồm các trường chính như sau :

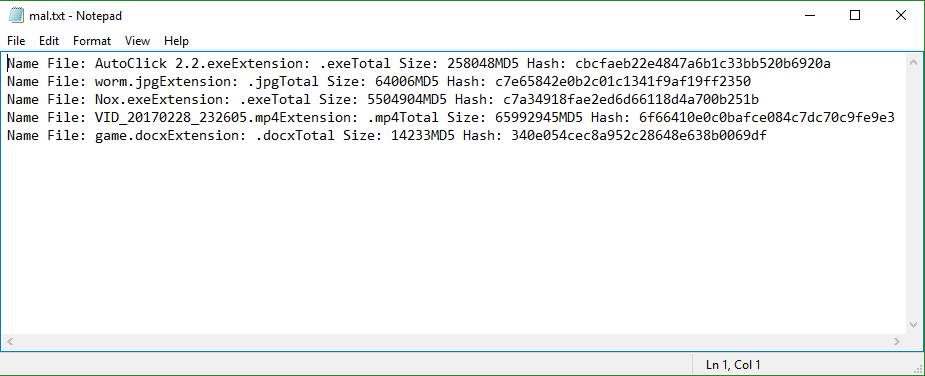
* + - MD5 : Mã băm MD5 trích xuất từ file mẫu
    - Date : Ngày giờ tạo
    - Filename : Tên file mẫu
    - Size: Kích thước của file mẫu
    - MINEType : Kiểu định dạng file mẫu : exe, rar, mp3 …



Hình 3.1 Giao diện chương trình khi tạo mẫu mã độc

Chức năng của chương trình:

Khi ấn “Save” thì file mẫu nhập vào sẽ được mã hóa thành chuỗi MD5 và tự động lưu vào file text để tạo lập cơ sở dữ liệu.



Hình 3.2 Dữ liệu mã độc trong file text

# KẾT LUẬN

Sau một thời gian tập trung nghiên cứu, do còn hạn chế về kiến thức quá trình thực tập cơ sở còn nhiều hạn chế. Kết quả đạt được là bản báo cáo này đã thực hiện được một số nội dung sau:

**Về phần lý thuyết:**

* + - Tìm hiểu tổng quan về mã độc hại, tình hình phát triển mã độc hại trên thế giới những năm gần đây.
    - Tìm hiểu được các cách thức phát hiện mã độc cơ bản.
    - Tìm hiểu về các phương pháp phân tích mã độc.
    - Tìm hiểu và xây dựng cơ sở dữ liệu mẫu mã độc theo chuẩn chung

**Về phần thực nghiệm:**

* + - Xây dựng chương trình tạo mẫu mã độc để phục vụ cho việc quản lý cơ sở dữ liệu mẫu mã độc.

**Hạn chế:**

Chưa xây dựng được cơ sở dữ liệu một cách đầy đủ các thành phần.

**Hướng phát triển**

Áp dụng thêm các phương pháp tạo mẫu mã độc khác để hoàn thành chương trình tạo cơ sở dữ liệu mẫu mã độc và áp dụng trong thực tế đi từ quy mô nhỏ tới lớn. Xây dựng một cơ sở dữ liệu mã độc đầy đủ hơn, truy xuất nhanh hơn và có thể áp dụng vào thực tế.

# TÀI LIỆU THAM KHẢO

[1]. https://vi.wikipedia.org/wiki/Phần\_mềm\_ác\_ý

[2]. <http://www.conmaz.net/>

[3]. Đặng Đức Dũng - Nghiên Cứu Cơ Chế Lây Nhiễm Và Cách Phòng Chống Mailware Trong Máy Tính – 2012

[4]. Addison and Wesley – *The Art of Computer Virus Research and Defense-*Feb.2005

[5]. <https://www.hex-rays.com/products/ida/>

[6]. [www.ollydbg.de](http://www.ollydbg.de/)

[7]. Michael Sikorski and Andrew Honig – *Practical Malware analysis* – 2012

[8]. http://standards.ieee.org/develop/indconn/icsg/

[9]. http://www.clamav.net

[10]. Creating-signatures-for-ClamAV  
[12]. https://stackoverflow.com/

# PHỤ LỤC

**Code chương trình tạo mẫu mã độc**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Windows.Forms;

namespace test2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

/\*\* Event Open file \*/

private void btnOpenFile\_Click(object sender, EventArgs e)

{

StringBuilder sb = new StringBuilder();

OpenFileDialog dialogOpen = new OpenFileDialog();

dialogOpen.Title = "Open MD5 File";

if (dialogOpen.ShowDialog() == DialogResult.OK)

{

txtLocationFile.Text = dialogOpen.FileName;

txtMD5File.Text = checkMD5(dialogOpen.FileName);

//

FileInfo oFileInfo = new FileInfo(dialogOpen.FileName);

sb.Append("Name File: " + oFileInfo.Name + "\n");

// DateTime dtCreationTime = oFileInfo.CreationTime;

// sb.Append("Date and Time File Created: " + dtCreationTime.ToString() + "\n");

sb.Append("Extension: " + oFileInfo.Extension + "\n");

sb.Append("Total Size: " + oFileInfo.Length.ToString() + "\n");

sb.Append("MD5 Hash: " + txtMD5File.Text);

lbInformationFile.Text = sb.ToString();

}

}

/\*\* Check md5 file method \*/

public string checkMD5(string FilePath)

{

//tạo đối tượng mã hóa MD5

MD5 MyMD5 = MD5.Create();

//đọc file

FileStream fs = new FileStream(FilePath, FileMode.Open);

//mã hóa mảng byte bằng MD5

byte[] HashCode = MyMD5.ComputeHash(fs);

//chuyển mảng byte thành chuỗi

StringBuilder SB = new StringBuilder();

for (int i = 0; i < HashCode.Length; i++)

SB.Append(HashCode[i].ToString("x2"));

return SB.ToString();

}

/\*\* Event Save file \*/

private void btnSave\_Click(object sender, EventArgs e)

{

AppendText(lbInformationFile.Text, "C://AdwCleaner/mal.txt");

}

public static void AppendText(string line, string pathFile)

{

using (StreamWriter sw = File.AppendText(pathFile))

{

sw.WriteLine(line);

sw.Close();

};

}

/\*\* Event Clear Form \*/

private void btnClear\_Click(object sender, EventArgs e)

{

txtLocationFile.Text = string.Empty;

txtMD5File.Text = string.Empty;

}

}

}

/\*\* Check md5 file method \*/

public string checkMD5(string FilePath)

{

//tạo đối tượng mã hóa MD5

MD5 MyMD5 = MD5.Create();

//đọc file

FileStream fs = new FileStream(FilePath, FileMode.Open);

//mã hóa mảng byte bằng MD5

byte[] HashCode = MyMD5.ComputeHash(fs);

//chuyển mảng byte thành chuỗi

StringBuilder SB = new StringBuilder();

for (int i = 0; i < HashCode.Length; i++)

SB.Append(HashCode[i].ToString("x2"));

return SB.ToString();

}

/\*\* Event Save file \*/

private void btnSave\_Click(object sender, EventArgs e)

{

AppendText(lbInformationFile.Text, "C://AdwCleaner/mal.txt");

}

public static void AppendText(string line, string pathFile)

{

using (StreamWriter sw = File.AppendText(pathFile))

{

sw.WriteLine(line);

sw.Close();

};

}

/\*\* Event Clear Form \*/

private void btnClear\_Click(object sender, EventArgs e)

{

txtLocationFile.Text = string.Empty;

txtMD5File.Text = string.Empty;

}

}

}

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

*Thái Nguyên, ngày … tháng … năm 2017*

**Giảng viên hướng dẫn**