**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**

**ĐỖ MINH HUY**

**VÕ QUỐC VƯƠNG**

# LUẬN VĂN TỐT NGHIỆP

# XÂY DỰNG CÔNG CỤ PHÁT HIỆN MÃ ĐỘC DỰA TRÊN CLAMAV

# A CLAMAV-BASED MALWARE DETECTION TOOL

**KỸ SƯ NGÀNH AN TOÀN THÔNG TIN**

**GIẢNG VIÊN HƯỚNG DẪN**

**TS.PHẠM VĂN HẬU**

**TP. HỒ CHÍ MINH, NĂM 2019**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**

**ĐỖ MINH HUY – 15520296**

**VÕ QUỐC VƯƠNG – 15521035**

**LUẬN VĂN TỐT NGHIỆP**

# XÂY DỰNG CÔNG CỤ PHÁT HIỆN MÃ ĐỘC DỰA TRÊN CLAMAV

# A CLAMAV-BASED MALWARE DETECTION TOOL

**KỸ SƯ NGÀNH AN TOÀN THÔNG TIN**

**GIẢNG VIÊN HƯỚNG DẪN**

**TS.PHẠM VĂN HẬU**

**TP. HỒ CHÍ MINH, NĂM 2019**

**DANH SÁCH HỘI ĐỒNG BẢO VỆ LUẬN VĂN**

Hội đồng chấm luận văn tốt nghiệp, thành lập theo Quyết định số …… ngày …….. của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

1. ……. – Chủ tịch.
2. ……. – Thư ký.
3. ……. – Ủy viên.
4. ……. – Ủy viên.

ĐẠI HỌC QUỐC GIA TP. HCM

**TRƯỜNG ĐẠI HỌC**

**CÔNG NGHỆ THÔNG TIN**

**CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**Độc lập – Tự do – Hạnh phúc**

TP. Hồ Chí Minh, ngày…..tháng 01 năm 2019

# NHẬN XÉT LUẬN VĂN TỐT NGHIỆP

# CỦA CÁN BỘ HƯỚNG DẪN

**Tên luận văn:**

**XÂY DỰNG CÔNG CỤ PHÁT HIỆN MÃ ĐỘC DỰA TRÊN CLAMAV**

**(A CLAMAV-BASED MALWARE DETECTION TOOL)**

| **Nhóm SV thực hiện:** | | **Cán bộ hướng dẫn:** |
|---|---|---|
| Đỗ Minh Huy | - 15520296 | Phạm Văn Hậu |
| Võ Quốc Vương | - 15521035 | |

**Đánh giá Luận văn:**

1. Về cuốn báo cáo

| | | | |
|---|---|---|---|
| Số trang | …….. | Số chương | …….. |
| Số bảng số liệu | …….. | Số hình vẽ | …….. |
| Số tài liệu tham khảo | …….. | Sản phẩm | …….. |

Một số nhận xét về hình thức cuốn báo cáo:

……...

.....

……...

……...

2. Về nội dung nghiên cứu:

.....

……...

.....

.....

3. Về chương trình ứng dụng:

……...

.....

……...

……...

4. Về thái độ làm việc của sinh viên:

……...

.....

……...

**Đánh giá chung:** Luận văn đạt/không đạt yêu cầu của một luận văn tốt nghiệp kỹ sư, xếp loại Giỏi/Khá/Trung bình.

……...

.....

……...

**Điểm từng sinh viên:**

Võ Quốc Vương : **……../10**

Đỗ Minh Huy : **……../10**

**Người nhận xét**

(Ký tên và ghi rõ họ tên)

ĐẠI HỌC QUỐC GIA TP. HCM
**TRƯỜNG ĐẠI HỌC**
**CÔNG NGHỆ THÔNG TIN**

**CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**
**Độc lập – Tự do – Hạnh phúc**

TP. Hồ Chí Minh, ngày…..tháng 01 năm 2019

# NHẬN XÉT LUẬN VĂN TỐT NGHIỆP
# CỦA CÁN BỘ PHẢN BIỆN

**Tên luận văn:**

**XÂY DỰNG CÔNG CỤ PHÁT HIỆN MÃ ĐỘC DỰA TRÊN CLAMAV**

**(A CLAMAV-BASED MALWARE DETECTION TOOL)**

**Nhóm SV thực hiện:** **Cán bộ hướng dẫn:**

Đỗ Minh Huy - 15520296

Võ Quốc Vương - 15521035

**Đánh giá Luận văn:**

5. Về cuốn báo cáo

| | | | |
|---|---|---|---|
| Số trang | …….. | Số chương | …….. |
| Số bảng số liệu | …….. | Số hình vẽ | …….. |
| Số tài liệu tham khảo | …….. | Sản phẩm | …….. |

Một số nhận xét về hình thức cuốn báo cáo:

……...

.....

……...

.....

6. Về nội dung nghiên cứu:

.....

……...

.....

……...

7. Về chương trình ứng dụng:

……...

.....

……...

.....

8. Về thái độ làm việc của sinh viên:

……...

.....

……...

.....

**Đánh giá chung:** Luận văn đạt/không đạt yêu cầu của một luận văn tốt nghiệp kỹ sư, xếp loại Giỏi/Khá/Trung bình.

……...

.....

……...

.....

**Điểm từng sinh viên:**

Đỗ Minh Huy : **……../10**

Võ Quốc Vương : **……../10**

**Người nhận xét**

(Ký tên và ghi rõ họ tên)

# LỜI CẢM ƠN

*Nhóm chúng em xin gửi lời cảm ơn chân thành đến các quý thầy cô trường Đại học Công Nghệ Thông Tin nói chung và quý thầy cô khoa Mạng máy tính và truyền thông nói riêng đã truyền dạy cho em những kiến thức quý báu trong suốt thời gian học tại đây và để chúng em hoàn thành khoá luận này.*

*Đặc biệt xin gửi lời cảm ơn sâu sắc đến thầy Tiến sĩ Phạm Văn Hậu Trưởng phòng thí nghiệm An toàn thông tin đã quan tâm, hỗ trợ, theo sát, trực tiếp hướng dẫn và đồng hành cùng chúng tôi trong suốt quá trình thực hiện để hoàn thành luận văn này với kết quả tốt nhất.*

*Nhóm em cũng gửi lời cảm ơn đến gia đình, bạn bè, và các anh trong Trung Tâm An Ninh Mạng CNSC là những người đã luôn động viên, khuyến khích và đưa ra những lời khuyên bổ ích để chúng tôi có thể hoàn thành luận văn này.*

*Xin chân thành cảm ơn!*

*Nhóm tác giả*

*Đỗ Minh Huy*

*Võ Quốc Vương*

ĐẠI HỌC QUỐC GIA TP. HCM
**TRƯỜNG ĐẠI HỌC**
**CÔNG NGHỆ THÔNG TIN**

**CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**
**Độc lập – Tự do – Hạnh phúc**

# ĐỀ CƯƠNG CHI TIẾT

**Tên đề tài:** Xây dựng công cụ phát hiện mã độc dựa trên ClamAV (A ClamAV-Based Malware Detection Tool).

**Cán bộ hướng dẫn: TS.Phạm Văn Hậu.**

**Sinh viên thực hiện đề tài:**

| | |
|---|---|
| Đỗ Minh Huy | 15520296 |
| Võ Quốc Vương | 15520937 |

**Thời gian thực hiện:** Từ ngày 02/09/2019 đến ngày 21/12/2019

**Nội dung đề tài:**

**Mục tiêu, đối tượng và phạm vi**:

**Mục tiêu:** Nhằm phát hiện mã độc gây ra, đề tài "Xây dựng công cụ phát hiện mã độc dựa trên ClamAV" đã phát triển công cụ phát hiện mã độc có tên là Avos UIT có khả năng phát hiện và cách ly mã độc ngoài ra còn có chức năng bảo vệ người dùng phục vụ khỏi các tác nhân xấu. Việc xây dựng một công cụ nhận diện mã độc mã nguồn mở với các tính nâng cao là nội dung mang tính cấp thiết và hữu ích, góp phần nâng cao kiến thức trong việc phát triển công cụ nhận diện mã độc.

**Đối tượng áp dụng:** Loại mã độc trên hệ điều hành Windows, bộ signature của ClamAV, các phương pháp phân tích mã độc, phương pháp phát hiện và loại bỏ mã độc.

**Phạm vi nghiên cứu:** Phát hiện mã độc của công cụ viết bằng c/c++ trên hệ điều hành Windows 10 32-bit. Các loại tệp tin thực thi nguy hiểm trên Windows như là: exe, if,msi,msp,com,scr…

**Nội dung, phương pháp**:

**Nội dung:**

- Tìm hiểu lập trình Win32 API.
- Tìm hiểu lập trình Driver.
- Tìm hiểu các phương pháp phát hiện và loại bỏ mã độc.
- Nghiên cứu và tìm hiểu về ClamAV, một phần mềm tự do nguồn mở cho việc phát hiện và loại bỏ các loại mã độc.
- Xây dựng phần mềm phát hiện và loại bỏ mã độc dựa trên Clamav.

**Phương pháp thực hiện:**

- Nghiên cứu tổng quan về các loại mã độc, cách thức lây lan phá hoại của chúng.
- Nghiên cứu các phương pháp phát hiện mã độc phổ biến hiện nay

**Kế hoạch thực hiện:**

| STT | THỜI GIAN | NỘI DUNG | CÔNG VIỆC |
|---|---|---|---|
| 1 | 01/09 - 30/09 | - Lập nhóm.<br>- Gặp giáo viên hướng dẫn, đề xuất đề tài, xây dựng ý tưởng cho luận văn | - Gặp giáo viên và trao đổi về ý tưởng thực hiện luận văn.<br>- Tìm các nghiên cứu, bài báo, dự án liên quan đến đề tài sẽ thực hiện.<br>- Tìm hiểu các chức năng cần triển khai. |
| 2 | 1/10 - 15/10 | Đăng ký đề tài | - Gặp giáo viên hướng dẫn đăng ký đề tài<br>- Trình bày hướng nghiên cứu cho luận văn với giáo viên, trao đổi các nội dung sẽ thực hiện trong đề tài. |
| 3 | 16/10 - 31/10 | - Tạo môi trường máy ảo và sửa các lỗi xảy ra trong quá trình chuẩn bị môi trường viết code.<br>- Lập flowchart cho các tính năng dự định sẽ làm. | - Đọc các tài liệu hướng dẫn trên google về các gói cần có để có thể biên dịch được code driver. |
| 4 | 1/11 | Báo cáo tiến độ | -Gặp giáo viên hướng dẫn<br>-Báo cáo tiến độ công việc |
| 5 | 2/11 - 30/11 | - Viết code tính năng Family Option trong phần mềm, phác thảo giao diện trên sketch và triển khai . | - Dùng ngôn ngữ c/c++ để code ra driver thực hiện các tính năng này, dùng c# để làm giao diện . |
| 6 | 01/12 - 14/12 | - Viết code cho tính năng firewall. | - Dùng ngôn ngữ c/c++ để code driver để capture, sửa đổi, loại bỏ các packets được gửi |

| | | | |
|---|---|---|---|
| | | - Viết code cho tính năng Shields và từng tính năng khác | đến windows network stack.<br>- Dùng ngôn ngữ c/c++ để viết các dịch vụ sử dụng 1 số API của thư viện WIN32 để thực hiện các tính năng đã nói ở trên. |
| 7 | 15/12 | Báo cáo tiến độ | -Gặp giáo viên hướng dẫn<br>-Báo cáo tiến độ công việc |
| 8 | 16/12 - 27/12 | -Xây dựng và hoàn thiện công cụ.<br>- Tổng hợp kết quả, viết báo cáo | -Xây dựng và hoàn thiện công cụ.<br>- Hoàn thiện các kết quả.<br>- Viết báo cáo luận văn tốt nghiệp. |
| 10 | 28/12 | Báo cáo tiến độ | -Gặp giao viên hướng dẫn<br>-Báo cáo tiến độ công việc |
| 11 | 29/12 - 30/12 | - Tiếp nhận các nhận xét báo cáo từ giáo viên hướng dẫn và chỉnh sửa. | - Bổ sung và hoàn thiện báo cáo.s |
| 12 | 31/12 - 6/1 | Phản biện đề tài | - Phản biện đề tài trước hội đồng.<br>- Khắc phục các hạn chế và hoàn thiện đề tài luận văn lần cuối trước khi bảo vệ. |
| 13 | 7/1 - 12/1 | Bảo vệ đề tài | Bảo vệ luận văn trước hội đồng. |

**Xác nhận của CBHD** **TP. HCM, ngày…..tháng…..năm 2019**

(Ký tên và ghi rõ họ tên) **Sinh viên 1** **Sinh viên 2**

(Ký tên và ghi rõ họ tên)

**MỤC LỤC**

## DANH MỤC HÌNH VẼ

## DANH MỤC BẢNG

## DANH MỤC TỪ VIẾT TẮT

| | |
|---|---|
| API | Application Program Interface |
| CVD | ClamAV Virus Database |
| ClamAV | Clam AntiVirus |
| SDK | Software Development Kit |
| GDI | Graphic Device Interface |
| GUI | Graphic User Interface |
| IRP | Interrupt Request Paquet |
| TCP | Transmission Control Protocol |
| URL | Uniform Resource Locator |
| UDP | User Datagram Protocol |
| MBR | Master Boot Record |
| WPF | Windows Presentation Foundation |

# TÓM TẮT LUẬN VĂN

Với mục tiêu phát hiện mã độc và bảo vệ người dùng máy tính tránh khỏi các nguy cơ tấn công bằng mã độc, virus máy tính….Công cụ phát triển trong luận văn này có các tính năng cơ bản nhất là kiểm tra, quét các tập tin thư mục ổ đĩa trong hệ thống máy tính của người dùng nhằm phát hiện các virus,  phần mềm độc hại trên máy tính, ngay lập tức cảnh báo người dùng , đưa chúng ra vùng cách ly để xử lý.

Các kỹ thuật phát hiện mã độc cơ bản bao gồm so sánh các dữ liệu với Clam AntiVirus viết tắt là ClamAV đã nhận dạng trước đó nhằm kịp thời phát hiện các virus độc hại, phát hiện và kiểm tra các hoạt động bất thường trên máy tính nhằm tìm ra các phần mềm độc hại hoạt động ẩn trên máy tính của người dùng.

Nhóm nghiên cứu hy vọng công cụ này sẽ giúp ích được cho các nghiên cứu sau này.

# Chương 1. MỞ ĐẦU

## 1.1. Lý do chọn đề tài

Mã độc từ lâu đã là mối nguy hiểm luôn rình rập người dùng máy tính. Đối với các doanh nghiệp, tổ chức, mối nguy này càng lớn hơn khi các thông tin quan trọng, nhạy cảm của cả tô chức có thê bị mất mát, lộ lọt bất cứ lúc nào bởi mã độc nằm trong một máy tính nào đó trong mạng.

Mã độc máy tính phát triển theo sự phát triển của công nghệ phần mềm cũng như phần cứng máy tính. Hệ điều hành thay đổi, virus cũng thay đổi để có thể ăn bám, ký sinh trên hệ điều hành mới. Sự phát triển của Internet làm bùng nổ sự lây lan của virus trên phạm vi toàn cầu với hậu quả cực kỳ nghiêm trọng. Các hệ thống máy tính thường xuyên phải đối phó với các đợt tấn công, dẫn đến nguy cơ bị đình trệ, tắc nghẽn và tê liệt.

Việc sử dụng virus lấy cắp tài khoản ngân hàng hay thông tin cá nhân quan trọng, mở cửa sau cho tin tặc đột nhập chiếm quyền điều khiển và các hành động khác nhằm có lợi cho người phát tán trở nên phổ biến.

Ở Việt Nam hiện này, có rất nhiều phần mềm phát hiện mã độc. Nhưng đa số, các phần mềm đó đều là mã nguồn đóng và thương mại. Có rất ít phần mềm phát hiện mã độc mã nguồn mở.

## 1.2. Mục đích nghiên cứu

Nhằm phát hiện mã độc gây ra, đề tài "Xây dựng công cụ phát hiện mã độc dựa trên ClamAV" đã nghiên cứu và phát triển phần mềm phát hiện mã độc Avos UIT có khả năng phát hiện và cách ly mã độc, bảo vệ người dùng phục vụ cho mục đích giáo dục và học tập.

Việc xây dựng một công cụ nhận diện mã độc mã nguồn mở với các tính nâng cao là nội dung mang tính cấp thiết và hữu ích, góp phần nâng cao kiến thức trong việc phát triển công cụ nhận diện mã độc.

## 1.3. Đối tượng và phạm vi nghiên cứu

### 1.3.1. Đối tượng nghiên cứu

- Nghiên cứu các loại mã độc trên Windows.
- Nghiên cứu bộ signature của ClamAV.
- Nghiên cứu các phương pháp phân tích mã độc.
- Nghiên cứu các phương pháp phát hiện và loại bỏ mã độc.
- Nghiên cứu phần mềm mã nguồn mở ClamAV.

### 1.3.2. Phạm vi nghiên cứu

- Hệ điều hành Windows 32 bit.
- Các loại tệp tin thực thi trên Windows: exe, if,msi,msp,com,scr.

## 1.4. Nội dung và phương pháp nghiên cứu

### 1.4.1. Nội dung nghiên cứu

- Tìm hiểu lập trình Win32 API.
- Tìm hiểu lập trình Driver.
- Tìm hiểu các phương pháp phát hiện và loại bỏ mã độc.
- Nghiên cứu và tìm hiểu về ClamAV, một phần mềm tự do nguồn mở cho việc phát hiện và lọai bỏ các loại mã độc.
- Xây dựng phần mềm phát hiện và loại bỏ mã độc dựa trên Clamav.

### 1.4.2. Phương pháp nghiên cứu

- Nghiên cứu tổng quan về các loại mã độc, cách thức lây lan phá hoại của chúng.
- Nghiên cứu các phương pháp phát hiện mã độc phổ biến hiện nay.

## 1.5. Bố cục luận văn

Luận văn sẽ trình bày tổng quan về tình hình nghiên cứu thuộc lĩnh vực phân tích  mã độc cũng như mối nguy hiểm của mã độc có thể gây ảnh hưởng nghiêm trọng đến sự an toàn của người dùng. Sau khi có cái nhìn cơ bản về những vấn đề đó, luận văn sẽ trình bày cách công cụ tiếp cận và phát hiện, nhận diện sự tồn tại của mã độc tồn tại trong máy tính người dùng dựa trên ClamAV.

Trong chương 1, luận văn sẽ giới thiệu tổng quan về đề tài, trình bày lý do nhóm chọn đề tài, đối tượng và phạm vi nghiên cứu, nội dung và phương pháp nghiên cứu.

Ở chương 2, luận văn sẽ trình bày và giới thiệu các phần mềm anti-virus mã nguồn mở đã được nghiên cứu và phát triển trước đó. Những ưu điểm và nhược điểm của các phần mềm này. Từ đó chỉ ra những vấn đề mà đề tài tập trung nghiên cứu và giải quyết. Hơn nữa là trình bày tổng quan một số kiến thức, công nghệ được áp dụng trong suốt quá trình làm luận văn này.

Ở chương 3 này, luận văn sẽ giải thích chi tiết cách thiết kế công cụ, mô hình hệ thống, chi tiết hoạt động của các chức năng trong công cụ.

Trong chương 4, luận văn trình bày cách thiết lập môi trường chạy ứng dụng, cấu hình máy tính và  mô tả các kết quả thực nghiệm các tính năng.

Trong chương 5, luận văn trình bày những kết quả đạt được, những đóng góp mới và những đề xuất mới. Kiến nghị về những hướng nghiên cứu tiếp theo.

# Chương 2. TỔNG QUAN VÀ KIẾN THỨC NỀN TẢNG

## 2.1. TỔNG QUAN

### 2.1.1. Những nghiên cứu liên quan

#### 2.1.1.1. ClamWin

ClamWin là một giải pháp diệt virus miễn phí và có mã nguồn mở (FOSS) cho Windows, đóng vai trò là bộ quét ClamAV front-end và mã nguồn mở.

Đây là phần mềm diệt virus sử dụng cho hệ điều hành windows 8/7/Vista /XP/Me/200/2003. Phần mềm diệt virus ClamWin được sử dụng bởi 600,000 người dùng trên thế giới.

Đây là phần mềm mã nguồn mở. Phần mềm diệt virus ClamWin có đặc điểm như là khả năng phát hiện cao với nhiêu loại virus, phần mềm gián điệp, tự động đặt lịch quét, tự động tải dữ liệu các mẫu virus, thực hiện lệnh quét trên từng file với việc click chuột phải chọn files và thực hiện quét, tích hợp với Outlook để quét email có gắn file đính kèm. ClamWin cũng có bao gồm khả năng vừa thực hiện quét hệ thống theo yêu cầu lần quét theo lập trình tại những mốc thời gian định trước , cũng có thể chạy "memory scan/ quét bộ nhớ", sẽ chỉ quét những chương trình hiện đang chạy trên bộ nhớ hệ thống.

Do ClamWin được xây dựng để kết thúc trở thành công cụ mã nguồn mở nên có một số hạn chế hiển nhiên. Một ví dụ là sự thiếu sót rõ ràng của tính năng bảo vệ theo thời gian thực , vốn đặc biệt quan trọng để bảo vệ trước những đợt tấn công zero-day (kiểu tấn công mà khả năng bị tác động vẫn chưa được lưu tâm và vá lỗi).

#### 2.1.1.2. D32 Anti virus

D32 Anti-virus là chương trình phát hiện và diệt virus dành các máy tính hệ điều hành 32 bit. D32 D32 Anti-virus là phần mềm diệt virus đầu tiên của Việt Nam. Tiền thân của phần mềm này là D2 (Diagnose and Destroy Viruses) anti-virus chạy trên hệ điều hành MS-DOS rất phổ biến ở Việt Nam từ thập niên 1990. Kể từ

lúc phát hành (tháng 2 năm 2001) đến nay, D32 đã có nhiều cải tiến quan trọng; cùng với BKAV, D32 trở thành phần mềm diệt virus trên nhiều hệ thống công nghệ thông tin ở Việt Nam.

D32 Anti-virus có tốc độ tìm và diệt virus rất nhanh, phần mềm này cũng có chế độ bảo thời gian thực mạnh mẽ. Trong quá trình quét có thể phát hiện và ngăn chặn phần mềm độc hại trước khi cài đặt. Nếu phát hiện ra tập tin đáng ghi ngờ thì phần mềm sẽ nhắc nhở bạn và tự động gửi các tập tin bị nhiễm bệnh phải kiểm dịch. Giao diện của D32 Anti-virus giúp người dùng có thể  biết được đầy đủ thông tin liên quan đến phần mềm này. Từ thông tin về tình trạng máy tính, đến phiên bản cập nhật mới nhất phần mềm và thời gian về lần quét gần đây nhất, cũng như số lượng các mối đe dọa được tìm thấy.

Các tính năng đáng chú ý là : có chức năng tự động bảo vệ, tự động diệt và cách ly, khôi phục các tập tin qua trọng mặc dù bị nhiễm virus đã được các ly bằng tiện ích phục hồi, có chức năng bảo vệ thời gian thực.

#### 2.1.1.3. Moon secure Antivirus

Là một antivirus mã nguồn mở, sử dụng ClamAV engine. Được thiết kế và phát triển để phát hiện virus, Trojans, Spyware cũng giống như những antivirus khác. Hỗ trợ phát hiện và xử lý virus trong các tập tin nhiễm độc một cách nhanh chóng. Hiển thị thành biểu tượng trên thanh hệ thống giúp truy xuất nhanh chóng. Hiển thị báo cáo cụ thể trên cửa sổ chính của chương trình. Antivitus này còn có tính  năng nổi bật như là tính năng tự động bảo vệ, sử dụng tài nguyên thấp, quét các ổ đĩa di động và các ổ đĩa cố định, có thể phát hiện virus, trojan và phần mềm gián điệp, thực hiện scan và chống rootkit. Tính năng tự động update.

#### 2.1.1.4. Armadito Antivirus

Đây là một chương trình chống vi-rút mã nguồn mở cho các máy chủ và PC bảo vệ các hệ thống khỏi mọi vi-rút và phần mềm độc hại. Phần mềm có thể được sử dụng từ xa từ bất kỳ vị trí nào thông qua giao diện trực quan và nó sẽ cung cấp quyền truy cập vào rất nhiều tính năng. Phần mềm có các tính năng như

là: Armadito Antivirus cung cấp bảo vệ thời gian thực trên Windows. Phần mềm này bao gồm khả năng cách ly các khu vực cụ thể và nó cung cấp thông báo truyền tải và sự kiện. Ngoài ra Armadito Antivirus đi kèm với giao diện thân thiện với người dùng, cung cấp một thiết lập an toàn và chạy các chức năng cho phép bạn kiểm tra trạng thái của máy và máy chủ của bạn ngay lập tức. Bảng điều khiển cung cấp quyền truy cập để quét theo yêu cầu, bảo vệ thời gian thực, tạp chí phát hiện mối đe dọa, hỗ trợ kỹ thuật và nhiều tính năng hơn. Ngoài ra có thể kiểm tra tất cả các tính năng có trong phần mềm này bằng cách truy cập trang web chính thức của Armadito Antivirus.

### 2.1.2. Các vấn đề còn tồn đọng của phần mềm diệt virus mã nguồn mở

Mặc dù những phần mềm antivirus trên đã có đầy đủ tính năng cần thiết của một phần mềm antivirus. Thế nhưng, đa phần những phần mềm này đều hoạt dộng trên nền Window XP (32 bit).

ClamWin hỗ trợ hệ điều hành Microsoft Windows 8, Microsoft Windows 7, Microsoft Windows Vista, Microsoft Windows XP, Microsoft Windows Me, Microsoft Windows 2000, Microsoft Windows 98, Microsoft Windows Server 2012, 2008 and 2003. Một nhược điểm của ClamWin nữa  đó là phần mềm này sẽ không tự động quét virus cho một file nào đó. Để quét virus, chúng ta phải thực hiện quá trình này.

D32 Antivirus chỉ hổ trợ hệ điều hành Windows XP 32 bit. Bởi vì là phần mềm chống virus tương đối cũ nên cũng còn nhiều thiếu sót.

Ngoài ra tất cả các tính năng chuyên về việc phát hiện mã độc mà chưa có các tính  năng bảo vệ người dùng, đặc biệt là trẻ em.

### 2.1.3. Hướng giải quyết vấn đề

Chính vì vậy, nhóm đã tập trung vào tiến hành xây một công cụ mã nguồn mở dựa trên thư viện của ClamAV. Về cơ bản có đầy đủ các tính năng như là một antivirus. Sau đó, phát triển để chạy trên nền tảng đó là Windows 10 cho cả bản 32

bit và 64 bit. Nhóm còn tích hợp một số chức năng cơ bản như firewall, kiểm soát trẻ em, xây dựng giao diện tối ưu dễ sử dụng hơn.

## 2.2. KIẾN THỨC NỀN TẢNG

Hầu hết các chức năng trong công cụ được lập trình bằng c/c++. Các chức năng này sử dụng 1 số API trong bộ thư viện Windows.h để thực hiện chức năng của nó. Chỉ riêng ở tính năng real-time. Nhóm sử dụng một MiniFilter driver để theo dõi các hoạt động tạo tệp tin, mở tệp tin, hay xóa tệp tin của một ứng dụng nào nó khi thực thi lên, dựa vào đó để thực hiện các hoạt động quét liên quan. Sau đó, tất cả các chức năng này sẽ trao đổi dữ liệu với ClamAV. Quá trình tiếp theo, ClamAV sẽ nhận dữ liệu liệu, phân tích và trả kết quả về cho các chức năng . Nếu như ứng dụng an toàn, các chức năng này sẽ không làm gì cả. Ngược lại, nếu ClamAV trả về là không an toàn, thì chức năng sẽ gửi qua giao diện để thông báo đến người dùng. Quá trình trao đổi dữ liệu giữa chức năng và ClamAV, giữa chức năng và giao diện đều được trao đổi qua socket.

### 2.2.1. Tổng quan về ClamAV

#### 2.2.1.1. Giới thiệu ClamAV

ClamAV là một bộ công cụ chống virus mã nguồn mở cho UNIX, được thiết kế một cách đặc biệt cho việc quét thư điện tử trên gateway. Nó cung cấp một vài tiên ích bao gồm một daemon đa luồng linh động, một bộ quét dòng lệnh và một công cụ tự động cập nhật cơ sở dữ liệu. ClamAV có chứa cơ sở dữ liệu lớn định nghĩa về các loại virus và được cập nhật hàng ngày thông qua kết nối Internet. Cốt lõi của ClamAV là một thư viện chia sẻ các danh sách: mã nguồn virus, malware, toolkit, rootkit,… cho phép ta dễ dàng nhận biết và phân biệt các nguy hại đang tồn tại trên máy thông qua các danh sách trong cơ sở dữ liệu.

#### 2.2.1.2. Các tính năng

Hỗ trợ quét theo dòng lệnh

- Quét theo lịch đặt sẵn.

- Quét nhanh, daemon đa luồng với sự hỗ trợ cho việc quét truy cập.
- Sử dụng giao diện milter cho sendmail.
- Nâng cao cơ sở dữ liệu cập nhật với sự hỗ trợ các bản cập và chữ ký số.
- Truy cập quét (Linux và FreeBSD ).
- Cơ sở dữ liệu virus được cập nhật nhiều lần trong ngày.
- Tích hợp hỗ trợ cho các định dạng lưu trữ khác nhau, bao gồm cả Zip, RAR, TAR, GZIP, BZIP2, OLE2, nội các, CHM, BinHex, SIS và các định dạng khác.
- Tích hợp hỗ trợ cho thực thi ELF và các tập tin thực thi Portable nén với UPX, FSG, NsPack, wwpack32, MEW, Upack và obfuscated với SUE, Y0da Cryptor và các định dạng khác.
- Tích hợp hỗ trợ cho các định dạng tài liệu phổ biến bao gồm MS Office và MacOffice hình ảnh, HTML, RTF và PDF.

Bên cạnh những tính năng ưu điểm như trên, ClamAV còn có một hạn chế: ClamAV không hỗ trợ cơ chế bảo vệ thời gian thực nên khi nào chúng ta có nghi ngờ về sự xuất hiện của virus, malware, toolkit,… thì có thể khởi động ClamAV lên để quét. Hoặc có thể dựa vào cơ chế đặt lịch quét mà ClamAV đã hỗ trợ trên.

Đó là lý do nhóm đã chọn xây dựng công cụ phát hiện mã độc dựa trên ClamAV.

#### 2.2.1.3. LibClamAV

Libclamav cung cấp một phương pháp dễ dàng và hiệu quả để thêm một sự ngăn chặn virus vào trong phần mềm. thư viện này là an toàn và trong suốt, có thể phát hiện và quét trong các tệp tin nén, thư điện tử, tệp tin MS Office, file thực thi và những đinh dạng đặc biệt khác. Libclamav đươc cấp bản quyền theo giấy phép GNU GPL v2, điều này có nghĩa là chúng ta không được phép xây dựng các phần mềm nguồn đóng, thương mại dựa vào nó. Tất cả các phần mềm sử dụng libclamav phải tuân theo GPL.

#### 2.2.1.4. Các hàm API

- **Header file**

Mỗi chương trình sử dụng libclamav phải thêm vào tệp tiêu đề (header file) clamav.h:

```
#include <clamav.h>
```

- **Bước khởi tạo**

Trước khi sử dụng libclamav, chúng ta nên gọi *cl_init()* để khởi tạo nó, tiếp theo để tạo một engine quét mới gọi *cl_engine_new()*. Để giải phóng các tài nguyên được cấp phát bởi engine sử dụng *cl_engine_free()*. Nguyên mẫu các hàm này:

```
Int cl_init(unsigned int options);
struct cl_engine *cl_engine_new(void);
int cl_engine_free(struct cl_engine *engine);
```

*cl_init()* và *cl_engine_free()* trả về *CL_SUCCESS* khi thành công hoặc một mã khác khi lỗi. cl_engine_new() trả về một con trỏ hoặc NULL nếu không đủ bộ nhớ.

- **Tải về cơ sở dữ liệu virus**

Tập lệnh dưới đây cung cấp một giao diện để tải về cơ sở dữ liệu virus:

```
const char *cl_retdbdir(void);
int cl_load(const char *path, struct cl_engine *engine,
unsigned int *signo, unsigned int options);
```

*cl_retdbdir(void)* trả về đường dẫn mặc định tới thư mục chứa cơ sở dữ liệu ClamAV. *cl_load()* tải về một file cơ sở dữ liệu đơn hoặc tất cả từ một thư mục cho trước ( khi *path* chỉ tới một thư mục). tham số thứ hai được sử dụng để chuyển con trỏ tới engine cái mà sẽ được cấp phát trước với *cl_engine_new()*. Một vài dấu hiệu đã được tải sẽ được thêm tới *signo*.

Tham số cuối cùng có thể truyền những cờ sau:

| Tên cờ | Chức năng |
|---|---|
| CL_DB_STDOPT | Đây là một biệt hiêu cho một tập các tùy chọn quét được đề nghị. |
| CL_DB_PHISHING | Tải về các dấu hiệu lừa đảo (phishing). |
| CL_DB_PHISHING_URLS | Khởi tạo mô-đun phát hiện phishing và tải các tệp .wdb và .pdb, |
| CL_DB_PUA | Tải về dấu hiệu cho các ứng dụng không mong muốn tiềm ẩn (PUA). |
| CL_DB_OFFICIAL_ONLY | Chỉ tải về các dấu hiệu chính thức từ cơ sở dữ liệu được ký số. |
| CL_DB_BYTECODE | Tải về bytecode |

- **Xử lý lỗi**

Sử dụng *cl_strerror()* để chuyển đổi mã lỗi thành các thông điệp có thể đọc được, hàm này trả về một chuỗi được cấp phát tĩnh:

```
if(ret != CL_SUCCESS) {
printf("cl_load() error: %s\n", cl_strerror(ret));
cl_engine_free(engine);
return 1;
}
```

- **Cấu trúc engine**

Khi tất cả cơ sở dữ liệu đã được tải, tiếp theo cần chuẩn bị engine phát hiện bằng cách *gọi cl_engine_compile()*. Trong trường hợp thất bại chúng ta vẫn cần giải phóng bộ nhớ được cấp cho engine với *cl_engine_free():*

```
int cl_engine_compile(struct cl_engine *engine);
```

- **Các hàm quét dữ liệu**

Có thể quét một file hoặc bộ mô tả sử dụng:

```
int cl_scanfile(const char *filename, const char **virname,
unsigned long int *scanned, const struct cl_engine *engine,
unsigned int options);
int cl_scandesc(int desc, const char **virname, unsigned
long int *scanned, const struct cl_engine *engine,
unsigned int options);
```

**- Định dạng CVD**

CVD (ClamAV virus database) là một tệp tarball được ký số chứa đựng một hoặc nhiều cơ sở dữ liệu. phần đầu là một chuỗi dài 512 byte với các trường được phân tách bằng dấu hai chấm:

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

sigtool –info hiển thị thông tin chi tiết về file CVD:

```
File: daily.cvd
Build time: 10 Mar 2008 10:45 +0000
Version: 6191
Signatures: 59084
Functionality level: 26
Builder: ccordes
MD5: 6e6e29dae36b4b7315932c921e568330
Digital signature:
```

zz9irc9irupR3z7yX6J+OR6XdFPUat4HIM9ERn3kAcOWpcMFxq
Fs4toG5WJsHda0Jj92IUusZ7wAgYjpai1Nr+jFfXHsJxv0dBkS5/XWMntj0T1ctNg
qmiF
+RLU6V0VeTl4Oej3Aya0cVpd9K4XXevEO2eTTvzWNCAq0ZzWNdjc

### 2.2.2. Tổng quan về Win32 API

#### 2.2.2.1. Giới thiệu Window API

Windows API (WinAPI, hay nhiều nơi vẫn dùng tên cũ là Win32 API) là các hàm thư viện và các định nghĩa khác (struct, enum,…) được Windows cung cấp cho người lập trình, để viết các ứng dụng trên nền Windows.

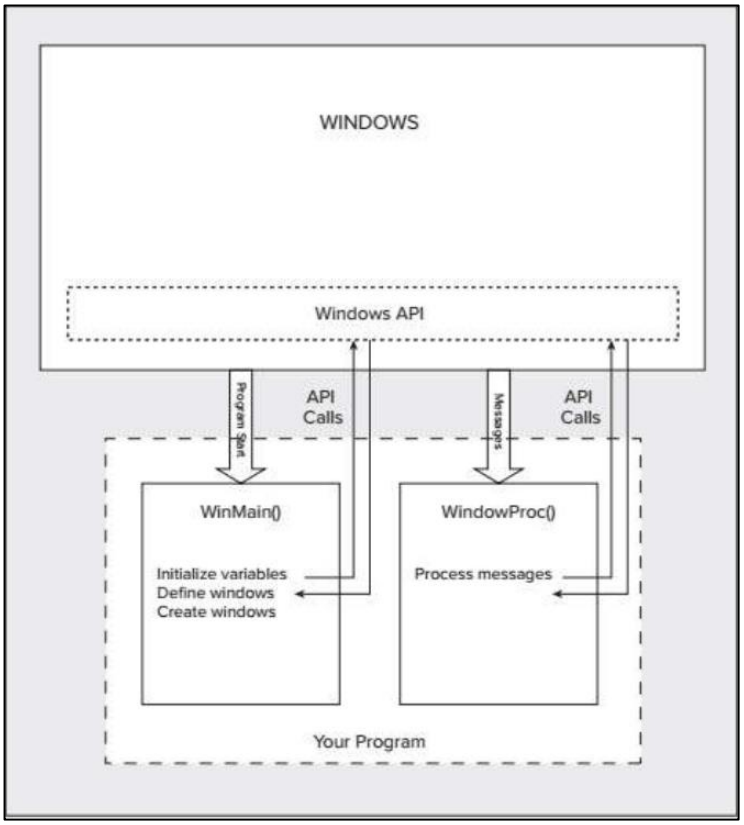
Windows API là giao diện lập trình nằm ngay trên nền Windows, cung cấp các hàm thao tác trực tiếp với hệ điều hành và phần cứng máy tính. Các ứng dụng Windows sẽ thông qua Windows API để thao tác với máy tính.Chính về thế nên nhóm đã chọn Win32 API để lập trình các dịch vụ có trong công cụ của mình.

Windows API là các hàm được sử dụng để tạo các ứng dụng Windows. Windows SDK bao gồm header file, library (Windows API) , sample, documentation và các tool (Windows SDK đã được tích hợp trong Visual Studio ). Widows API có thể chia ra thành các loại sau : Base dịch vụ Security Graphics User Interface Multimedia Windows Shell Networking .

- Base dịch vụ cung cấp các resource cơ bản trên Windows. Chúng bao gồm file system, devices, processes, threads, registry hoặc là xử lý error.
- Security cung cấp các function, interface, object và các programming element cho việc authentication, Cryptography, Security
- Graphics bao gồm các GDI (Graphic Device Interface),GDI+, DirectX hoặc OpenGL.
- User Interface cung cấp các function để tạo ra các window, các control.

- Multimedia cung cấp các tool để làm việc với video, sound, và các thiết bị đầu vào.

#### 2.2.2.2. Cấu trúc chương trình trong Windows




Hình 2.1: Chương trình ứng dụng Windows

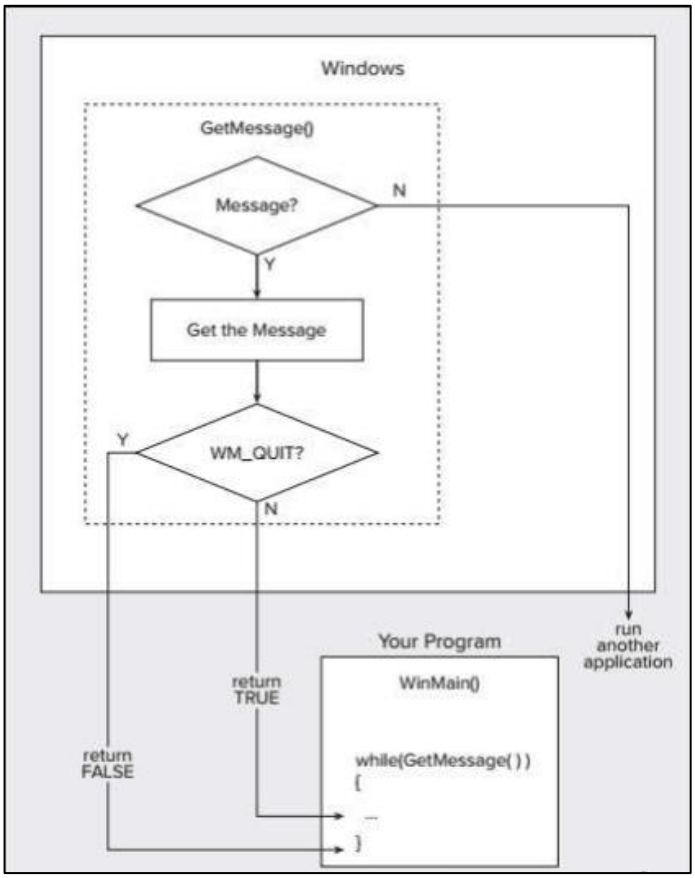
Mọi chương trình ứng dụng trong Windows bắt buộc phải có 2 hàm:

**WinMain():**

- Khởi tạo giá trị biên.
- Định Nghĩa của sổ.
- Hiển thị cửa sổ ứng dụng lên màn hình.

**Window procedure (WindowProc()).**

- Xử lý các thông diệp.




Hình 2.2: Xử lý thông diệp

**Khai báo hàm Winmain()**

```
MSG msg;
 while (GetMessage(&msg,NULL,0,0))
{ TranslateMessage (&msg);
DispatchMessage(&msg); }
```

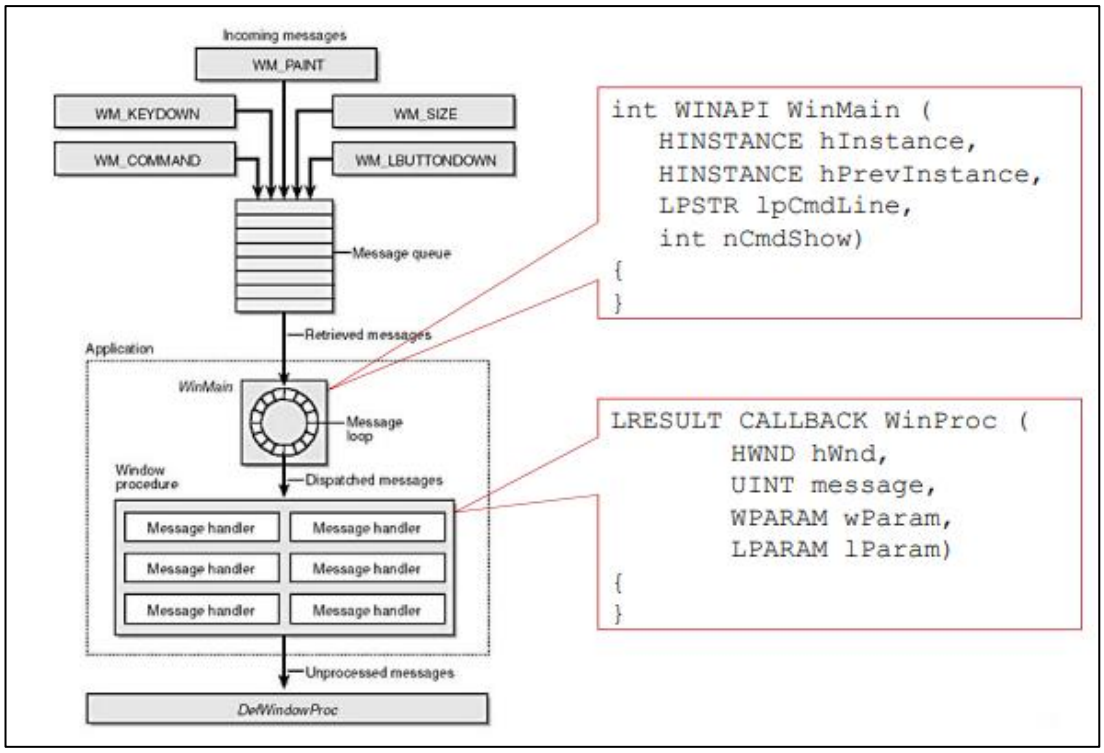
Hàm GetMessage() trả về 0 khi thông điệp chứa định danh WM_QUIT báo chương trình kết thúc.

**Cấu trúc của 1 gọi tin MSG**

```
typedef struct tagMSG {
 HWND hwnd; // Cửa sổ nhận thông điệp
UINT message; // Định danh thông điệp
```

```
WPARAM wParam; // Nội dung thông điệp
LPARAM lParam; // Nội dung thông điệp
DWORD time; // Thời điểm thông điệp được đưa lên
POINT pt; // Vị trí con trỏ khi thông điệp được // đưa lên
} MSG, *PMSG, *LPMSG;
```

**Hoạt động của chương trình**




Hình 2.3: Sơ đồ hoạt động của chương trình WIN32 API

**Message:**

- Hàm WinMain():

Tạo ra 1 vòng lặp thông điệp(message loop). Nó là vòng lặp vô hạn, chạy trong suốt vòng đời của ứng dụng. Message loop là 1 cấu trúc đợi và phát các sự kiện hoặc các message trong chương trình. HĐH Windows giao tiếp sử dụng các message.

- Message là giá trị số nguyên chỉ ra một sự kiện cụ thể.

Khi chúng ta click vào button, thay đổi kích thước cửa sổ hoặc đóng ứng dụng,…v.v. Thì sẽ có rất nhiều message được tạo ra. Các message này có thể không được xử lí đồng thời, mà các message này sẽ được đưa vào 1 hàng đợi thông điệp (message queue) và đợi để xử lí lần lượt từng message một.

- Hàm GetMessage() :

Được sử dụng để lấy các bản tin từ message queue

- Hàm TranslateMessage()

Chuyển virtual-key message thành character message. ( HĐH Windows tạo ra các Virtual-key message khi người dùng ấn các phím trên key-board (nhưng không phải là giá trị character). Ứng dụng muốn lấy được message này thì cần phải có hàm để translate virtual-key message thành character message).

- Hàm DispatchMessage() :

Dùng để phát message tới Window produce.

| Tên thông điệp | Chức năng |
|---|---|
| WM_CHAR | Khi nhập 1 kí tự từ bàn phím |
| WM_COMMAND | Khi lựa chọn các item trong popup menu |
| WM_CREAT | Khi windows được tạo |
| WM_DESTROY | Khi windows hủy bỏ |
| WM_RBUTTONDOWN | Khi click chuột phải |
| WM_LBUTTONDOWN | Khi click chuột trái |
| WM_MOUSEMOVE | Khi di chuyển con trỏ chuột |
| WM_PAINT | Khi windows được vẽ lại |
| WM_QUIT | Khi close windows |

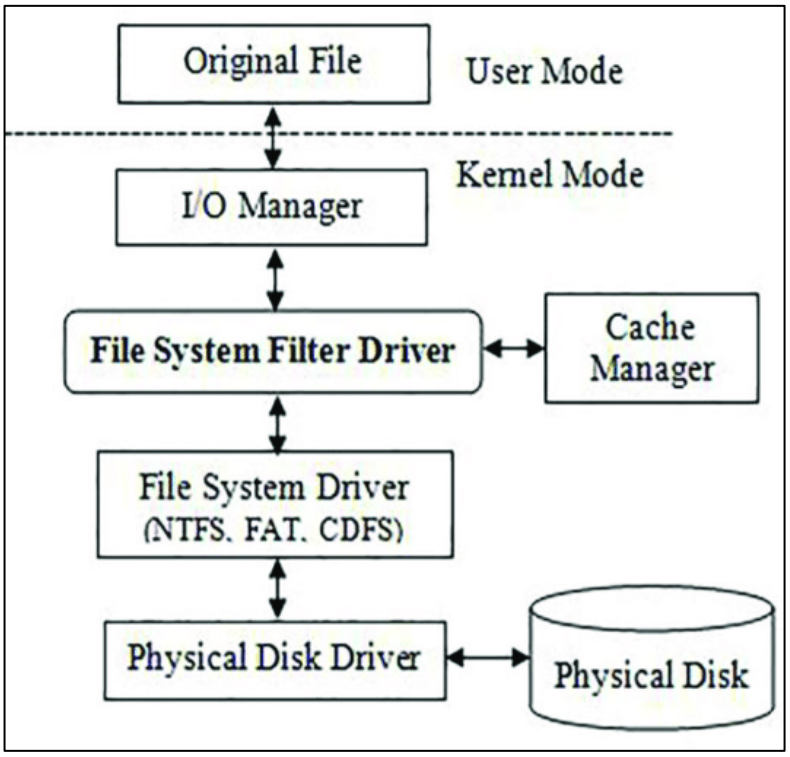
Bảng 2.1: Bảng các tin nhắn phát ra từ Window produce

### 2.2.3. Theo dõi tệp tin hệ thống

#### 2.2.3.1. Giới thiệu về trình theo dõi tệp tin

Là một công cụ theo dõi và ghi nhật ký các hoạt động xảy ra trên trình quản lý I/O. Minispy chứa 2 thành phần chính đó là User-mode và Kernel mode. Thành phần Kernel-mode đăng ký các hàm Callback tương ứng với các I/O, sau đó chuyển hoạt động xuống cho Filter Manager. Các hàm Callback này sẽ giúp Minispy ghi Log lại các hoạt động xảy ra trên hệ thống. Khi người dùng yêu cầu ghi lại thông tin này, thông tin sẽ được chuyển đến User-mode tiến hành xuất ra màn hình hay là ghi nhật ký xuống file.

Một số đặc trưng của mô hình I/O.




Hình 2.4: Sơ đồ hoạt động Minifilter

Mô hình quản lý I/O đưa ra một giao diện thống nhất cho tất cả driver ở mức độ kernel, bao gồm ở mức thấp nhất , mức trung gian và hệ thống tệp tin driver. Tất cả các yêu cầu I/O đến các drivers đều được thực hiện qua 1 gói tin yêu cầu có tên IRPs-I/O (request packets).

Mô hình I/O chia thành 2 lớp sau:

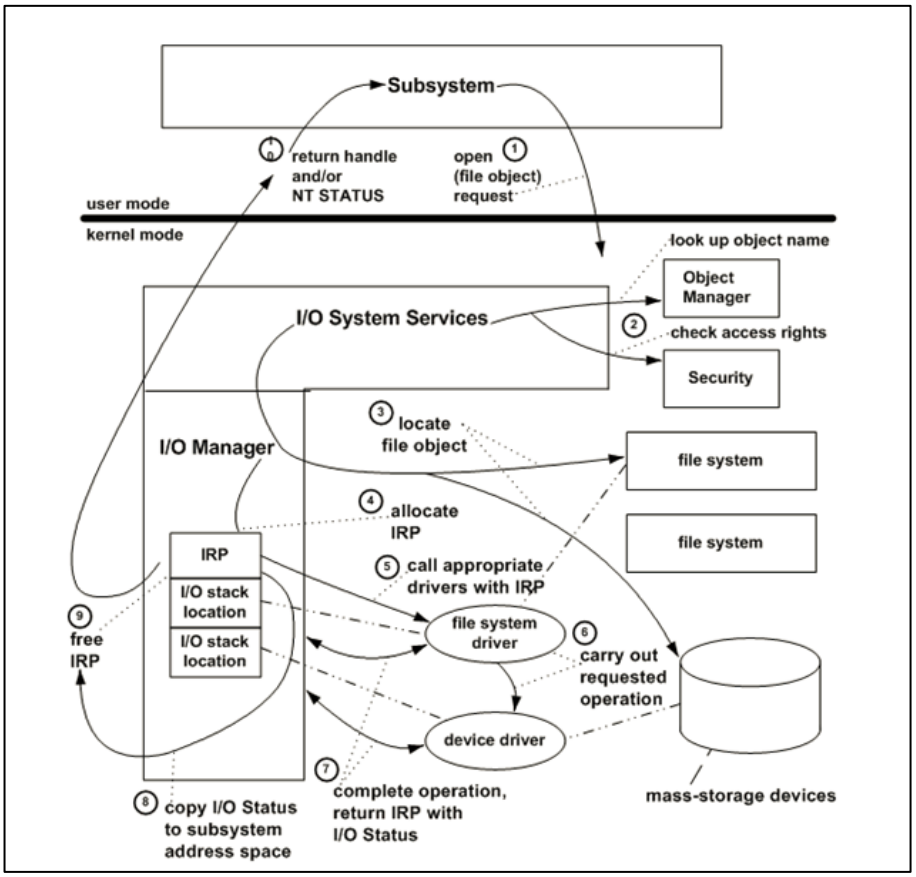
- I/O System: Chịu trách nhiệm giao tiếp với phần cứng:

Là một tập hợp các thành phần mức kernel quản lý các drivers thiết bị chạy trong hệ thống và thông tin liên lạc giữa các ứng dụng và drivers thiết bị. I/O System cùng với drivers thiết bị cho phép ứng dụng giao tiếp với drivers và thực hiện các yêu cầu từ thiết bị. Các I/O System có trách nhiệm chuyển tiếp yêu cầu từ ứng dụng đến drivers thiết bị chịu trách nhiệm để thực hiện. I/O System cũng được phân chia thành các lớp. Điều này có nghĩa là mỗi thiết bị có thể tương tác với nhiều drivers thiết bị bằng việc xếp chồng các drivers thiết bị. Điều này cho phép tạo ra một drivers chung, sử dụng để quản lý dữ liệu trên thiết bị. Chính vì điều này mà chúng ta có thể dễ dàng thêm các driver bọ lọc đề giám sát hoặc sửa đổi thông tin liên lạc giữa các driver và các ứng dụng.

- I/O Subsystem: Chịu trách nhiệm với các thực thi có liên quan đến đồ họa và xử lý các thông tin nhập vào  của người dùng. I/O Subsystem (hay Win32 Subsystem) là thành phần chịu trách nhiệm về mọi khía cạnh Giao diện người dùng. Bao gồm từ những engine đồ họa ở mức thấp, giao diện đồ họa thiết bị (GDI – graphics device interface)  và các thành phần ở mức User như cửa sổ, menu,… hay xử lý thông tin người dùng đưa vào.

#### 2.2.3.2. Chu trình của một yếu cầu I/O

Hình minh họa dưới đây cho chúng ta một cái nhìn tổng quan về những gì sẽ xảy ra khi một Subsystem mở một đối tượng tệp tin, đại diện cho tệp tin dữ liệu trong ứng dụng.




Hình 2.5: Chu trình hoạt động I/O

Các bước thực hiện:

1. Subsystem gọi một dịch vụ của I/O System để mở một tệp tin đã có tên.

2. Thành phần quản lý I/O gọi đến thành phần quản lý đối tượng (Object Manager) để tìm kiếm các tệp tin có tên và giúp giải quyết bất kỳ liên kết nào cho đối tượng tệp tin (file object). Nó cũng gọi đến các tham chiếu kiểm tra xem Subsystem có chính xác quyền để mở đối tượng tệp tin không.

3. Nếu Volume – khu vực lưu trữ truy cập đơn lẻ – chưa được gắn vào, thành phần quản lý I/O sẽ tạm ngừng yêu cầu mở tệp tin. Thay vào đó là việc gọi đến các hệ thống tệp tin khác cho đến khi xác định được rằng tệp tin cần mở được lưu trữ trong một thiết bị lưu trữ nào đó. Khi Volume được gắn, thành phần quản lý I/O sẽ tiếp tục lại yêu cầu.

4. Thành phần quản lý I/O cấp phát vùng nhớ và khởi tạo một IRP cho yêu cầu mở tệp tin. Đối với drivers, yêu cầu mở tương đương với yêu cầu khởi tạo.

5. Thành phần quản lý I/O gọi driver của hệ thống tệp tin thông qua các IRP. Các driver của hệ thống tệp tin truy cập I/O Stack Location để xác định hoạt động phải thực hiện và kiểm tra các thông số.

6. Các driver xử lý các IRP và hoàn thiện hoạt động I/O được yêu cầu. Thành phần quản lý I/O và các thành phần khác của hệ điều hành cung cấp việc gọi đến các thủ tục hỗ trợ ở kernel mode.

7. Các driver sẽ trả IRP về cho thành phần quản lý I/O để thiết đặt trạng thái vào/ra của yêu cầu là thành công hay lí do thất bại.

8. Thành phần quản lý I/O lấy được trạng thái vào/ra từ IRP, nên nó có thể trả về thông tin trạng thái thông qua Subsystem được bảo vệ tới lời gọi ban đầu.

9. Thành phần quản lý I/O sẽ giải phóng IRP khi đã hoàn thành.

10. Nếu thành công, thành phần quản lý I/O sẽ trả về một handle cho đối tượng tệp tin. Còn nếu có lỗi, nó sẽ trả về trạng thái thích hợp cho Subsystem.

Sau khi mở thành công một đối tượng tệp tin, Subsystem sẽ sử dụng handle được trả về để xác định đối tượng tệp tin trong yêu cầu tiếp theo cho các hoạt động như đọc, ghi, hay yêu cầu điều khiển của thiết bị vào/ra. Điều này được thực hiện bằng việc Subsystem gọi đến dịch vụ của I/O System. Thành phần quản lý I/O sẽ định tuyến các yêu cầu này tương tự các IRP đến các driver thích hợp gọi 1 dịch vụ của I/O System để mở 1 tệp tin đã có tên.

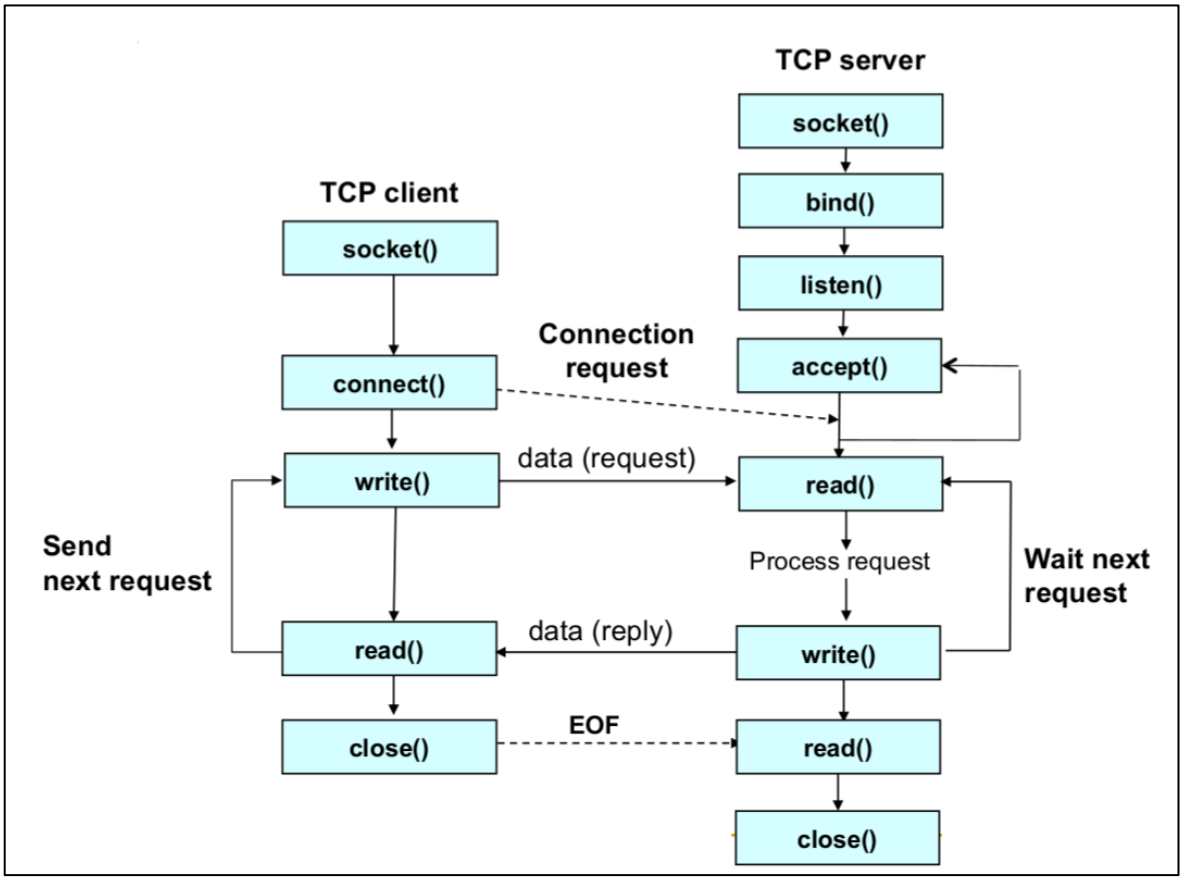
| Tên mã IRP_MJ | Chức năng |
|---|---|
| IRP_MJ_CREATE | Hệ điều hành gửi yêu cầu IRP_MJ_CREATE để mở 1 handle cho đối tượng tệp hoặc đối tượng thiết bị. Ví dụ: khi trình điều khiển gọi *ZwCreateFile*, hệ điều hành sẽ gửi yêu cầu IRP_MJ_CREATE để thực hiện thao tác mở thực tế |
| IRP_MJ_READ | Yêu cầu IRP_MJ_READ được gửi bởi Trình quản lý I / O hoặc bởi trình điều khiển hệ thống tệp. Yêu cầu này có thể được gửi, ví dụ, khi một ứng dụng chế độ người dùng đã gọi hàm Microsoft Win32 như *ReadFile.* |
| IRP_MJ_WRITE | Bất cứ lúc nào sau khi hoàn thành thành công một yêu cầu tạo. Có thể, một ứng dụng chế độ người dùng hoặc thành phần Win32 có handle cho đối tượng tệp đại diện nào đó đã yêu cầu truyền dữ liệu đến thiết bị. Có thể, trình điều khiển cấp cao hơn đã tạo và thiết lập IRP ghi. |
| IRP_MJ_SET_INFORMATION | Hệ điều hành sẽ gửi một yêu cầu IRP_MJ_SET_INFORMATION để đặt siêu dữ liệu về một tệp hoặc xử lý tệp. Ví dụ: khi trình điều khiển gọi ZwInIn informationFile, hệ điều hành sẽ gửi yêu cầu JRP_MJ_SET_INFORMATION. |
| IRP_MJ_SYSTEM_CONTROL | Thành phần Kernel-mode WMI có thể gửi yêu cầu IRP_MJ_SYSTIM_CONTROL sau khi đăng ký thành công của trình điều khiển với tư cách là nhà cung cấp dữ liệu WMI. Các IRP WMI thường được gửi khi người User-mode yêu cầu dữ liệu WMI. |

Bảng 2.2: Các chức năng các mã IRP

### 2.2.4. Lập trình Socket

#### 2.2.4.1. Giới thiệu tổng quan về lập trình Socket

Socket là một giao diện lập trình (API – Application Program Interface) ứng dụng mạng thông qua giao diện này có thể lập trình điều khiển việc truyền thông giữa 2 máy sử dụng các giao thức mức thấp như TCP,UDP… , Có thể tưởng tượng nó như một thiết bị truyền thông 2 chiều tương tự như tệp tin, chúng ta gửi/nhận dữ liệu giữa 2 máy, tương tự như việc đọc ghi trên tệp tin.




Hình 2.6: Sơ đồ hoạt động Socket

#### 2.2.4.2. Các bước để tạo lên 1 chương trình phía server

- Tạo socket với hàm socket (int family, int type, int protocol) các tham số trong đó theo thứ tự là họ giao thức, kiểu socket, kiểu giao thức.
- Gán địa chỉ cho socket bind (int sockfd, const struct sockaddr *sockaddr, socklen_t addrlen) các tham số lần lượt là mô tả socket vừa tạo, con trỏ chỏ đến địa chỉ socket, độ lớn địa chỉ.
- Chỉ định socket lắng nghe kết nối listen (int sockfd, int backlog) trong đó sockfd là mô tả socket vừa tạo, backlog là số lượng tối đa các kết nối đang chờ.
- Chờ/chấp nhận kết nối accept (int sockfd, struct sockaddr *cliaddr, socklen_t *addrlen) lần lượt có các tham số là mô tả socket vừa tạo, con trỏ tới cấu trúc địa chỉ socket của tiến trình kết nối đến, độ lớn cấu trúc địa chỉ.
- Thiết lập kết nối với máy chủ TCP connect (int sockfd, const struct sockaddr *servaddr, socklen_t addrlen).

#### 2.2.4.3. Các bước để tạo lên 1 chương trình phía client

- Tạo socket với hàm socket (int family, int type, int protocol) các tham số trong đó theo thứ tự là họ giao thức, kiểu socket, kiểu giao thức.
- Connect tới địa chỉ server với hàm connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr).
- Đọc dữ liệu từ server và ghi vào biến buffer read( sock , buffer, 1024).

# Chương 3. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## 3.1. Sơ đồ các chức năng có trong công cụ

### 3.1.1. Chức năng Scanning

#### 3.1.1.1. Mô tả các tùy chọn.

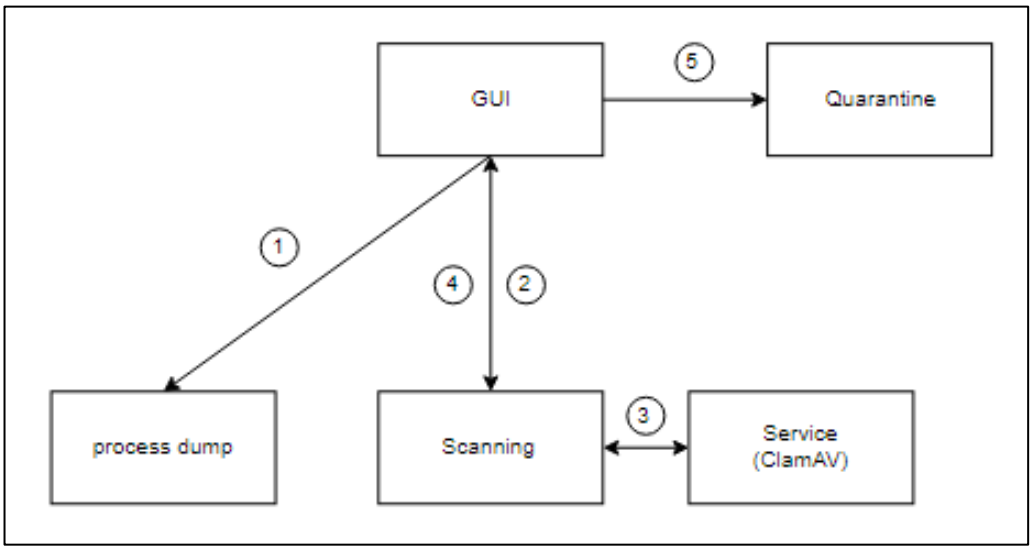
Chức năng scanning bao gồm ba tùy chọn nhỏ và tổng quan chức năng của chúng:

- Memory scan:
  - Sử dụng công cụ **process dump (một công cụ dump memory)** để dump toàn bộ process đang chạy trên hệ thống, bao gồm cả rootkit. Chính vì thế dung lượng ổ đĩa tạm thời sẽ tăng lên. Sau đó, chương trình sẽ quét toàn bộ các file được dump ra.
  - Sau khi chương trình quét xong, chương trình sẽ xóa các file dump đó và dung lượng ổ đĩa trở về ban đầu.
- Full scan:
  - Ban đầu, chương trình sẽ quét toàn bộ các file có trong hệ thống, bao gồm luôn các file ẩn, các file trong USB nếu có.
  - Sau đó, chương trình sẽ quét memory như đã đề cập bên trên.
  - Trong lúc quét, nếu phát hiện file pdf có chứa javascript hay ole chứa vba sẽ cách ly hoặc xóa nếu trong settings được bật.
  - Kế tiếp, chương trình sẽ quét MBR.
- Custom scan
  - Chương trình sẽ quét các file trong một folder chỉ định theo yêu cầu của người dùng.

#### 3.1.1.2. Sơ đồ hoạt động của Memory Scan

Nhóm sử dụng công cụ **process dump** để dump các tiến trình thành các tệp tin thực thi các đuôi mở rộng là .exe hay các thư viện động .dll trong một thư mục.

Kế tiếp, các tệp tin đó sẽ được gửi sang dịch vụ được kết nối với signature ClamAV để kiểm tra, nếu có tệp tin nào được dịch vụ trả về kết quả là mã độc thì sẽ hiện thị kết quả lên giao diện người dùng và cách ly.



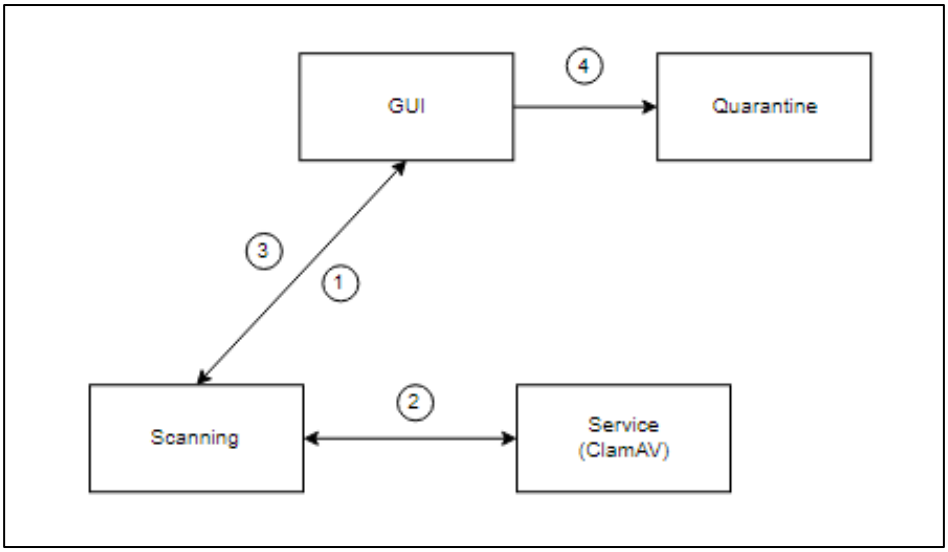

Hình 3.1: Sơ đồ chức năng Memory scan

Trong sơ đồ hoạt động trên, có tổng cộng năm bước và các thành phần trong sơ đồ giao tiếp bằng socket:

1. Từ giao diện, người dung chọn mục **memory scan** và bấm **Start** để thực thi. Các tệp tin được dump sẽ nằm trong một thư mục nhất định.
2. Kế tiếp, công cụ sẽ tự động lấy lần lượt đường dẫn các tệp tin được dump ra.
3. Sau đó, công cụ sẽ gửi các đường dẫn đó lần lượt cho Dịch vụ được kết nối cho ClamAV để kiểm tra và trả về kết quả.
4. Nếu có tệp tin nào là mã độc thì sẽ hiện thị kết quả lên giao diện cho người dung biết.
5. Các mã độc sẽ bị phát hiện được hiển thị trên giao diện, lúc này công cụ sẽ tự động đưa chúng vào vùng cách ly .

#### 3.1.1.3. Sơ đồ hoạt động của Custom Scan

Đối với chức năng này, sau khi người dùng chọn thư mục muốn quét. Công cụ sẽ lần lượt lấy các đường dẫn của các tệp tin trong thư mục đó rồi gửi cho dịch vụ được kết nối với signature ClamAV để kiểm tra và trả kết quả.




Hình 3.2: Sơ đồ chức năng Custom scan

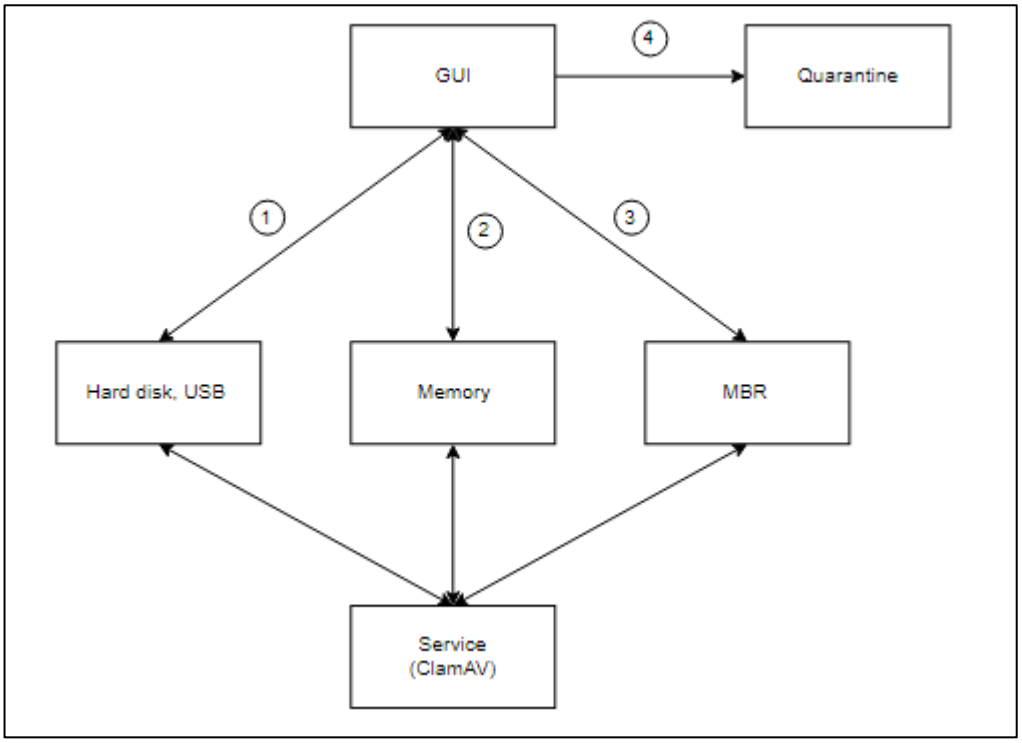
Trong sơ đồ hoạt động trên, có tổng cộng bốn bước và các thành phần trong sơ đồ giao tiếp bằng socket:

1. Từ giao diện, người dùng chọn mục **Custom scan** và bấm **Start**. Lúc này, một hộp thoại chọn thư mục sẽ hiện ra để người dùng chọn lựa. Sau đó, công cụ sẽ tiến hành lần lượt lấy các đường dẫn của các tệp tin có trong thư mục đó.
2. Kế tiếp, công cụ sẽ gửi các đường dẫn đó cho dịch vụ được kết nối với signature ClamAV để kiểm tra và trả kết quả.
3. Nếu có tệp tin nào là mã độc thì sẽ hiện kết quả trên giao diện thông báo cho người dùng biết.
4. Các mã độc sẽ bị phát hiện được hiển thị trên giao diện, lúc này công cụ sẽ tự động đưa chúng vào vùng cách ly .

#### 3.1.1.4. Sơ đồ hoạt động của Full Scan

Với chức năng này, công cụ sẽ quét toàn bộ hệ thống bao gồm:

- Các tệp tin có trong các ổ đĩa, USB.
- Memory như đã đề cập ở trên.
- MBR.




Hình 3.3: Sơ đồ chức năng Full Scan

Trong sơ đồ hoạt động trên, có tổng cộng bốn bước và các thành phần trong sơ đồ giao tiếp bằng socket:

1. Đầu tiên, công cụ sẽ lần lượt lấy đường dẫn của các tệp tin có trong hệ thống rồi gửi cho dịch vụ được kết nối signature ClamAV để kiểm tra và trả về kết quả
2. Tiếp theo, công cụ sẽ quét các tiến trình trong hệ thống như đã đề cập ở trên
3. Kế tiếp, công cụ sẽ tiến hành dump MBR và quét.
4. Các mã độc sẽ bị phát hiện được hiển thị trên giao diện, lúc này công cụ sẽ tự động đưa chúng vào vùng cách ly .

### 3.1.2. Chức năng Shields

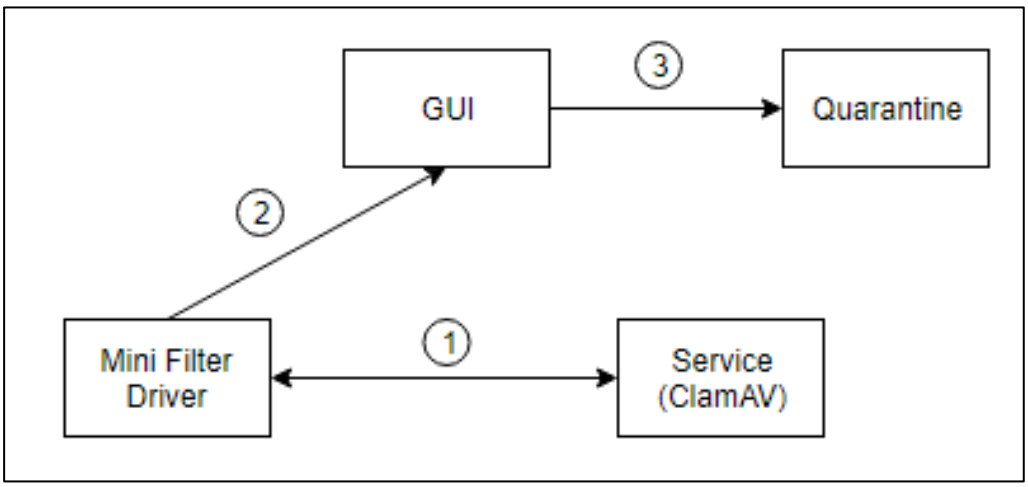
Thành phần máy tính sẽ được bảo vệ theo thời gian thực, bao gồm bốn thành phần chính:

- File Shield: Bảo vệ các tệp tin khỏi mã độc lây nhiễm theo thời gian thực, khi người  dung click vào một mã độc bất kì sẽ được ngăn chặn.
- Fraud Shield: Bảo vệ người dung tránh truy cập vào các website lừa đảo thông qua proxy.
- USB Shield: Khi usb được cắm vào máy tính thì sẽ lập tức quét ngay
- Family Shield: Bảo vệ trẻ em tránh truy cập web đen.

#### 3.1.2.1. Sơ đồ chức năng File Shield

File Shield: công cụ sẽ phát hiện và cách ly mã độc theo thời gian thực khi người dùng chạy mã độc trên máy tính. Ở đây, nhóm sử dụng MiniFilter Driver để lọc các tiến trình chạy và các tệp tin được mở theo thời gian thực.

Sơ đồ luồng thực thi của File Shield:




Hình 3.4: Sơ đồ chức năng File Shield

Trong sơ đồ hoạt động trên, có tổng cộng ba bước và các thành phần trong sơ đồ giao tiếp bằng socket:

1. Minifilter driver sẽ lọc các tiến trình được sinh ra hay các tệp tin được mở theo thời gian thực rồi gửi cho dịch vụ được kết nối với signature ClamAV để kiểm tra và trả về kết quả
2. Nếu kết quả trả là mã độc thì sẽ hiện thị popup cho người dung biết.
3. Lúc này mã độc sẽ được đưa vào vùng cách ly tự động.

#### 3.1.2.2. Sơ đồ chức năng Fraud Shield

Fraud Shield: Công cụ sẽ ngăn chặn máy tính truy cập vào các trang web lừa đảo dựa trên danh sách có sẵn, người dùng có thể thêm vào danh sách đó bằng tay. Ở đây, nhóm sử dụng web proxy để ngăn chặn máy tính truy cập các web chỉ định.

Với việc sử dụng web proxy thì ngay cả https vẫn sẽ bắt được, đảm bảo kết quả việc chặn triệt để.

Sơ đồ luồng thực thi của Fraud Shield:


Hình 3.5: Sơ đồ chức năng Fraud Shield

Trong sơ đồ hoạt động trên, có tổng cộng hai bước:

1. Đầu tiên, người dùng tiến hành truy cập web thông qua Browser.
2. Sau đó, web proxy kiểm tra xem địa chỉ truy cập đó có nằm trong danh sách bị cấm không, nếu có thì sẽ từ chối và hiển thị thông báo ngược lại cho người qua thông qua Browser, còn nếu không thì mọi thứ diễn ra bình thường.

#### 3.1.2.3. Sơ đồ chức năng Family Shield

Family Shield: Công cụ sẽ ngăn chặn trẻ em tiếp xúc với những nội dung không mong muốn như web đồi trụy, chơi game, các từ khóa khi lướt web. Với chức năng, nhóm tiếp tục sử dụng web proxy để lọc từ khóa lướt web bằng việc

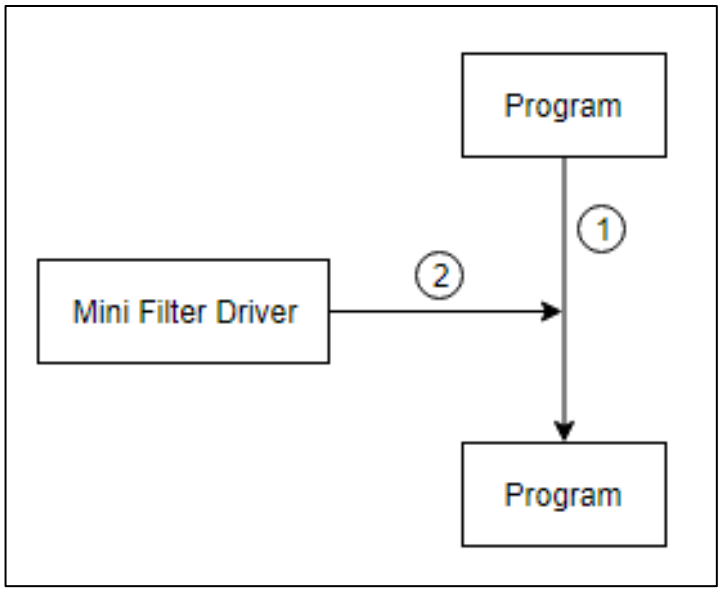
phân tích url của các công cụ tìm kiếm như google, yahoo,… thông qua regex. Còn với chức năng chặn một chương trình nhất định không cho trẻ em sử dụng thì nhóm dùng Mini Filter Driver để chặn.

Sơ đồ luồng thực thi của Family Shields cho cả chặn web và tiến trình:


Hình 3.6: Sơ đồ chức năng Family Shied

Trong sơ đồ hoạt động trên, đối với chặn web gồm hai bước:

1. Đầu tiên, người dùng tiến hành truy cập web thông qua Browser.
2. Sau đó, web proxy kiểm tra xem địa chỉ truy cập đó có nằm trong danh sách bị cấm không, nếu có thì sẽ từ chối và hiển thị thông báo ngược lại cho người qua thông qua Browser, còn nếu không thì mọi thứ diễn ra bình thường.




Hình 3.7: Sơ đồ chức năng chặn tiến trình

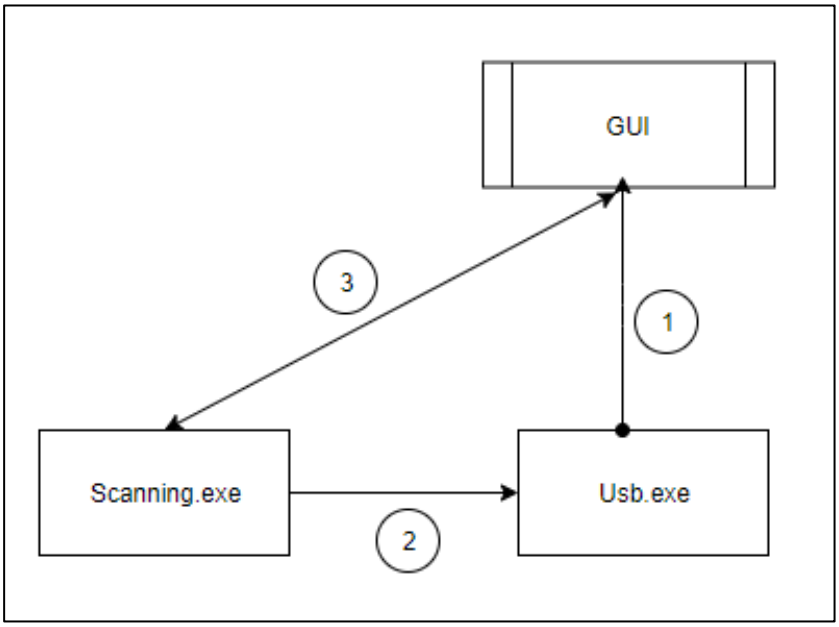
Đối với chặn tiến trình cũng có hai bước:

1. Đầu tiên, người dung mở chương trình cần sử dụng
2. Kế tiếp, MiniFilter driver sẽ kiểm tra tiến trình sắp chạy có nằm trong danh sách bị cấm không, nếu có sẽ kill lập tức, còn không thì mọi thứ diễn ra bình thường.

#### 3.1.2.4. Sơ đồ chức năng USB

USB Shield: Công cụ sẽ phát hiện USB được cắm vào máy tính theo thời gian thực và quét các tệp tin trong USB. Với chức năng, nhóm sử dụng các Win32 API có sẵn trong Windows để phát hiện ra các sự kiện khi USB được cắm vào máy tính.

Sơ đồ luồng thực thi của USB :




Hình 3.8: Sơ đồ chức năng USB

Trong sơ đồ hoạt động trên, có tổng cộng ba bước:

1. Đầu tiên, người dùng tiến hành cắm usb vào máy.
2. Giao diện sẽ tự động phát hiện ra sự kiện usb được gắm vào. Sau đó gửi lệnh scan.

3. Mô-đun chức năng sẽ scan ổ đĩa USB. Sau đó trả kết quả về cho giao diện người dùng.

### 3.1.3. Chức năng Firewall

Firewall là tường lửa giúp người dung ngăn ngừa truy cập vào các web lừa đảo, mạo danh. Chức năng này chứa danh sách địa chỉ web cho chức năng Fraud Shield trong Shields. Trong danh sách địa chỉ web người dùng có thể thêm bằng tay hoặc xóa và sửa theo ý muốn. Với chức năng này, nhóm sử dụng web proxy như đã trình bày ở trên. Danh sách địa chỉ web sẽ được cập nhật theo thời gian thực dựa trên signature ClamAV.

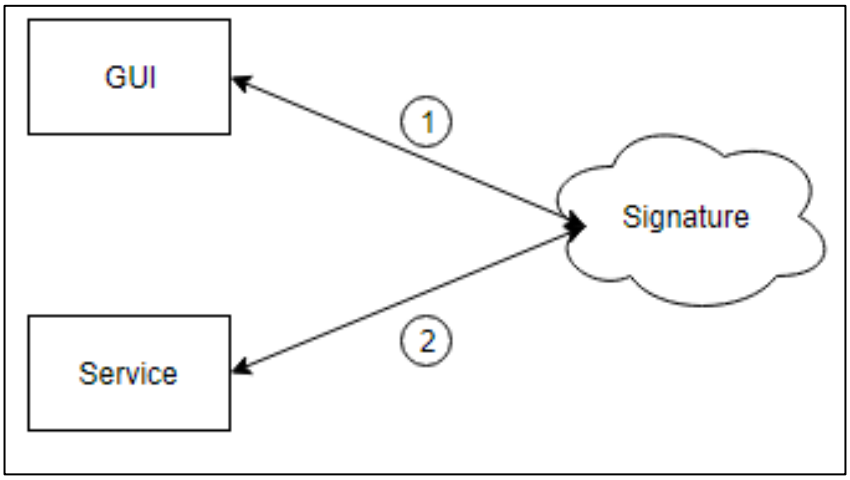
### 3.1.4. Chức năng Update

Đối với chức năng này, công cụ sẽ luôn được cập nhật theo thời gian thực thông qua dịch vụ, với các chu kì do người dùng lựa chọn. Ngoài ra, người dùng cũng có thể cập nhật bằng tay bằng cách nhấn vào Button Start trên giao diện. Địa chỉ cập nhật signature chính là trang chủ của ClamAV.

Tổng cộng có 3 file được update:

- Main.cvd
- Daily.cvd
- Bytecode.cvd

Sơ đồ luồng thực thi của Update:




Hình 3.9: Sơ đồ chức năng Update

Trong sơ đồ hoạt động trên gồm hai luồng:

1. Từ giao diện, người dùng có thể cập nhật bằng tay bằng Button Start.
2. Đối với dịch vụ, sau mỗi chu kì, công cụ sẽ được cập nhật.

### 3.1.5. Chức năng Family Options

Family Options chứa danh sách các địa chỉ web, từ khóa lướt web và các chương trình cấm chạy cho chức năng Family Shield trong Shields. Trong danh sách web người dùng có thể thêm bằng tay hoặc xóa và sửa theo ý muốn. Với chức năng này, nhóm sử dụng web proxy như đã trình bày ở trên và Minifilter driver.

Danh sách sẽ được thêm bởi người dung. Sơ đồ luồng thực thi đã được trình bày trong phần Family Shield.

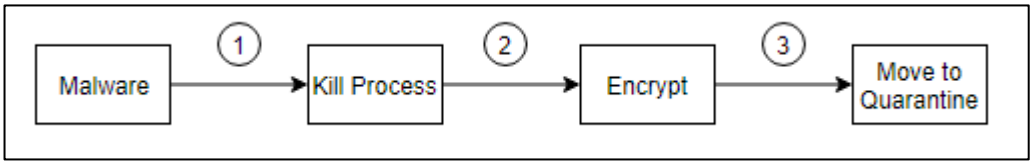
Chứa các thành phần bảo vệ trẻ em khỏi những nội dung không mong muốn, bao gồm:

- Target: cấm chạy những chương trình trên máy tính được chỉ định.
- Keyword: cấm lướt web với những từ khóa được chỉ định.
- URL: cấm truy cập vào những web đen.

### 3.1.6. Chức năng Quarantine

Thành phần chứa các mã độc hoặc các chương trình được người dùng cho vào để cách ly.

Đây là khu vực cách ly của mã độc khi được phát hiện bởi Shields và Scanning. Ngoài ra, người dung nếu cảm thấy nghi ngờ một tệp tin thì có thể di chuyển chúng vào đây bằng tay như thao tác thêm mới trong Family Options hay Firewall.




Hình 3.10: Sơ đồ chức năng Quarantine

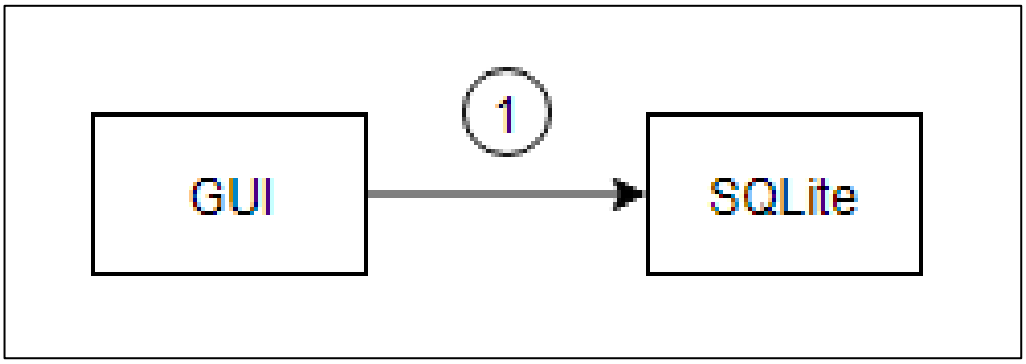
Trong sơ đồ hoạt động trên gồm ba bước:

1. Mã độc được phát hiện sẽ được kill tiến trình nếu có.
2. Kế tiếp, mã độc sẽ được encrypt bằng mã hóa XOR với một chuỗi ngẫu nhiên được sinh ra bởi công cụ
3. Sau đó, mã độc sẽ được di chuyển vào vùng cách ly, thực chất là một thư mục ẩn.

### 3.1.7. Chức năng Setting

Thành phần dùng để cấu hình cách hoạt động cho các chức năng. Sau khi điều chỉnh xong, công cụ cần được khởi động lại để chạy theo cấu hình mới. Cơ sở dữ liệu nhóm sử dụng là SQLite vì tính tiện dụng, nhỏ gọn của nó. Mọi thao tác cấu hình sẽ được thực hiện trực tiếp thông qua giao diện.

Sơ đồ thực thi của của Settings như sau:




Hình 3.11: Sơ đồ chức năng Setting

Trong sơ đồ hoạt động trên gồm một bước:

1. Từ giao diện, người dùng tiến hành cấu hình. Khi người dùng bấm vào Button Save thì sẽ công cụ sẽ lưu vào cơ sở dữ liệu SQLite .

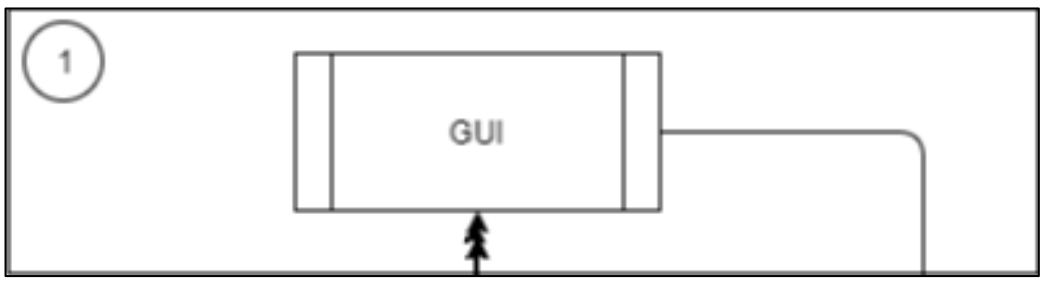
### 3.1.8. Chức năng Report

Thành phần sẽ chứa các kết quả, event của chương trình.

Chức năng này đơn giản chỉ là lưu lại nhật ký các sự kiện của công cụ sinh ra như sự kiện bắt được mã độc theo thời gian thực, kết quả của quá trình quét toàn bộ hệ thống, chặn web cấm, chặn chương trình cấm,… Và các sự kiện này cũng được lưu vào cơ sở dữ liệu SQLite.

## 3.2. Tổng kế mô hình hệ thống

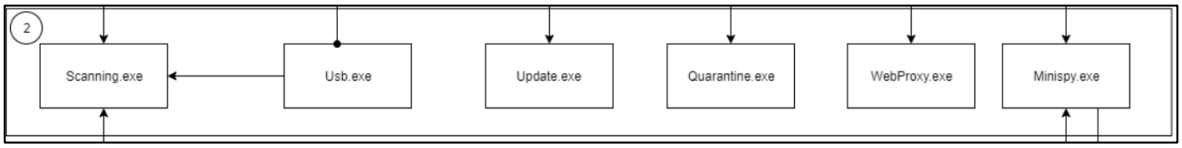
### 3.2.1. Giao diện người dùng




Hình 3.12: Giao diện người dùng

Đây là thành phần giao diện của công cụ. Giao diện có chức năng là mọi kết quả, thông báo sẽ được xuất thông qua đây. Tại đây người dùng có thể tương tác trực tiếp với công cụ, thực hiện các thao tác như quét toàn bộ máy, quét bộ nhớ, cách ly mã độc... . GUI ngoài việc giao tiếp với người dùng thì còn giao tiếp với Mô-đun chương trình thông qua socket.

### 3.2.2. Các Mô-đun chương trình

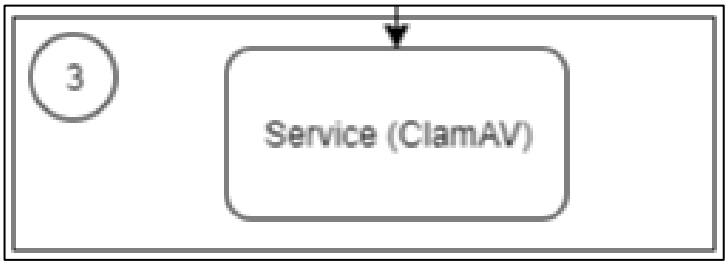



Hình 3.13: Các Mô-đun chương trình

Phần mềm chứa các Mô-đun chương trình để thực hiện các chức năng liên quan. Thành phần này bao gồm các Mô-đun sau:

- Scanning: Lần lượt lấy đường dẫn các tệp tin rồi gửi cho dịch vụ (ClamAV) qua socket, rồi truyền kết quả lên GUI qua socket.
- USB: Phát hiện USB được cắm vào máy tính theo thời gian thực, thông báo được gửi lên GUI qua socket hiện thông báo cho người dùng biết. Sau đó, tên kí tự ở đĩa USB được gửi cho Scanning.exe qua tham số để lấy các đường dẫn các tệp tin trong USB.
- Update: Cập nhật signature từ trang chủ ClamAV theo yêu cầu người dung từ GUI hoặc theo chu kì.
- Quarantine: Giao tiếp với GUI thông qua tham số. Đầu tiên sẽ kill tiến trình mã độc nếu có, sau đó mã hóa bằng thuật toán XOR với chuỗi ngẫu nhiên, kế tiếp mã độc được di chuyển vào vùng cách ly.
- Web Proxy: Lọc các địa chỉ web theo chỉ địa và các từ khóa lướt web.

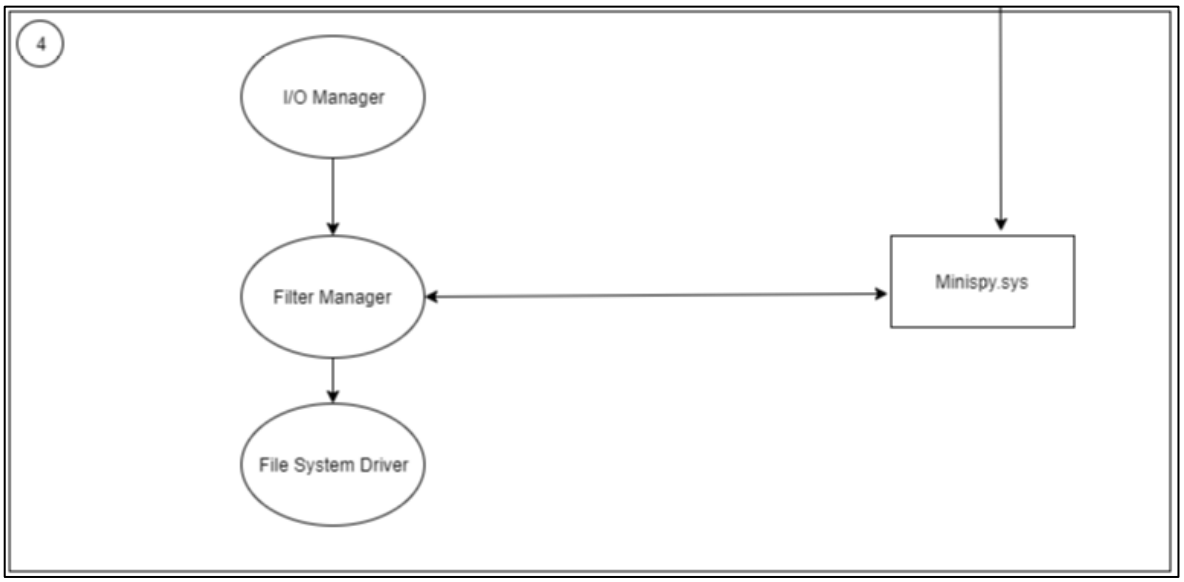
### 3.2.3. Dịch vụ




Hình 3.14: Mô hình thành phần Dịch vụ

Các dịch vụ chạy nền. Nơi tiếp nhận các dữ liệu mà các Mô-đun chương trình gửi xuống. Các dịch vụ này sẽ kết nối hai chiều với gui để có thể nhận lệnh cũng như đưa kết quả lên gui. Thành phần này bao gồm ClamAV.

### 3.2.4. Driver



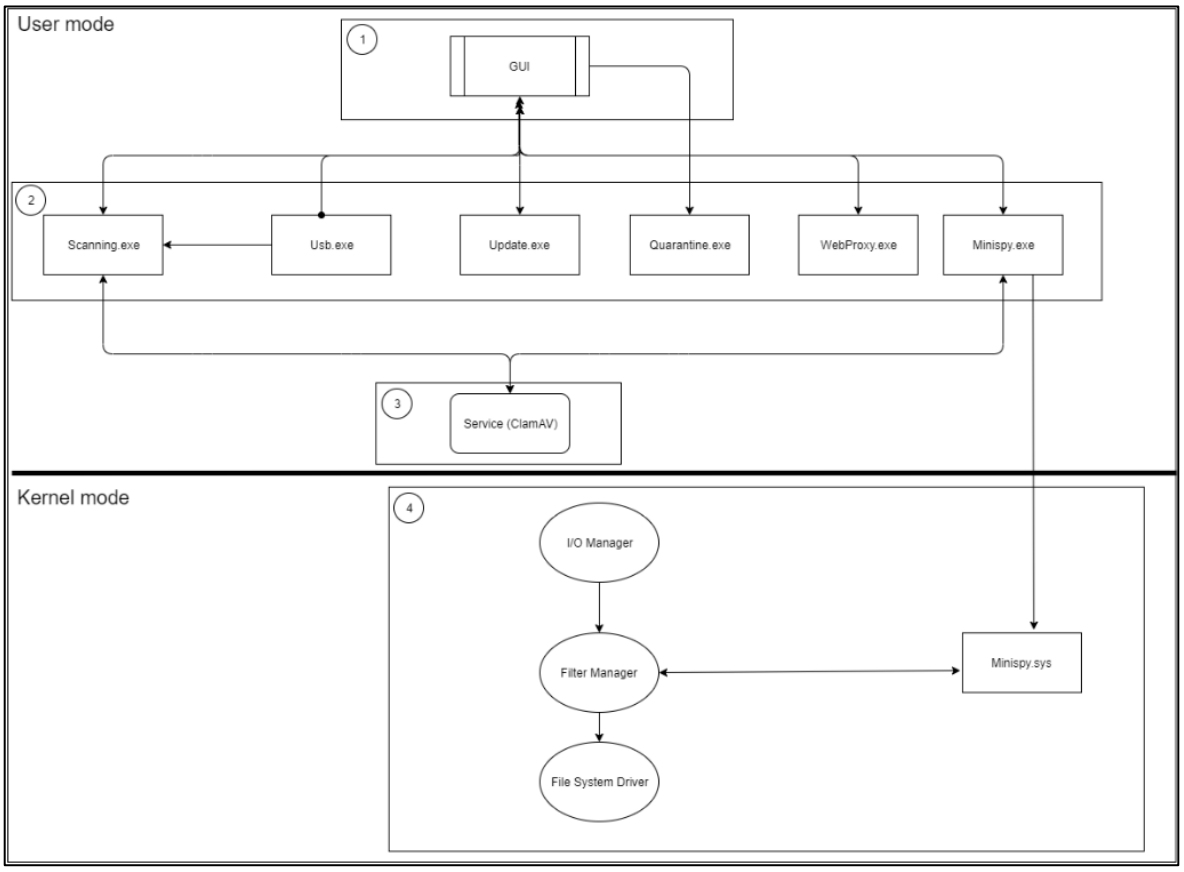

Hình 3.15: Mô hình thành phần Driver

Thành phần driver được cái vào hệ điều hành, phục việc cho việc theo dõi tệp tin hệ thống. Chúng sẽ lọc các tiến trìnhvà tập tin, khi có phát hiện tệp tin nào được mở bởi người dùng . Thành phần này bao gồm tệp tin sau:

- Minispy.inf: Tệp tin cài đặt driver.
- Minispy.exe: Chạy ở User-mode. Các đặt các tùy chọn cần thiết.
- Minispy.sys: Chạy ở Kernel-mode. Được cài thẳng vào hệ điều hành.

## 3.3. Tổng kết mô hình hệ thống

Tổng kết mô tả chi tiết thiết kế công cụ phát hiện mã độc. Kiến trúc của công cụ được chia làm các thành phần lớn: Giao diện người dùng, các Mô-đun chương trình, dịch vụ, driver.




Hình 3.16: Sơ đồ hệ thống tổng quát

Tất cả các thành phần đã được mô tả chi tiết ở trên.

# Chương 4. HIỆN THỰC HÓA HỆ THỐNG

## 4.1. Hiện thực hệ thống

### 4.1.1. Môi trường triển khai

Môi trường mà nhóm đề tài phát triển công cụ được thể hiện ở bảng sau:

| | |
|---|---|
| Ngôn nữ lập trình | C/C++, C# |
| Thư viện hổ trợ | Window 32 API |
| Hệ điều hành | Windows 10 (10586) |
| Công cụ lập trình | Visual studio 2015 |

Bảng 4.1: Bảng môi trường hiện thực

| | |
|---|---|
| RAM | 2 GB |
| Ổ cứng | 60 GB |
| Số CPU | 1 |

Bảng 4.2: Cấu hình máy hiện thực

### 4.1.2. Giao diện (GUI)

Ở đây, nhóm sử dụng WPF để thiết kế giao diện người dùng. WPF(Windows Presentation Foundation) do Microsoft phát triển, là công nghệ kế tiếp Windows Form dùng để xây dựng các ứng dụng dành cho máy trạm chạy hệ điều hành Windows. WPF được giới thiệu từ năm 2006 trong.NET Framework 3.0 (dưới tên gọi Avalon), công nghệ này nhận được sự quan tâm của cộng đồng lập trình viên bởi nhiều điểm đổi mới trong lập trình ứng dụng và khả năng xây dựng giao diện thân thiện, sinh động.

### 4.1.3. Hiện thực chức năng File Shield

Hàm main của chương trình File Shield

```
int _cdecl
```

```
main (
   _In_ int argc,
   _In_reads_(argc) char *argiáo viên[]
   )
{
printf( "Connecting to filter's port...\n" );

   hResult = FilterConnectCommunicationPort( MINISPY_PORT_NAME,
                                             0,
                                             NULL,
                                             0,
                                             NULL,
                                             &port );

   if (IS_ERROR( hResult )) {

      printf( "Could not connect to filter: 0x%08x\n", hResult );
      DisplayError( hResult );
      goto Main_Exit;
   }
}
```

Hàm FilterConnectCommunicationPort(): Mở 1 connection đến server port đã được tạo bởi file system minifilter.

```
int _cdecl
main (
   _In_ int argc,
   _In_reads_(argc) char *argiáo viên[]
```

```
   )
{
        .......
        pintf( "Creating logging thread...\n" );
        thread = CreateThread( NULL,
                    0,
                      RetrieveLogRecords,
                      (LPVOID)&context,
                      0,
                      &threadId);

         if (!thread) {
      result = GetLastError();
      printf( "Could not create logging thread: %d\n", result );
      DisplayError( result );
      goto Main_Exit;
        .........
   }
```

CreateThread() Tạo ra 1 thread để lắng nghe và đọc các log records mà hàm RetrieveLogRecords() trả về từ MiniSpy.sys.

```
DWORD
WINAPI
RetrieveLogRecords(
   _In_ LPVOID lpParameter
   )
{     ........
```

```
    hResult = FilterSendMessage( context->Port,
                                 &commandMessage,
                                 sizeof( COMMAND_MESSAGE ),
                                 buffer,
                                 sizeof(alignedBuffer),
                                 &bytesReturned );

      .........
  }
}
```

Trong hàm RetrieveLogRecords có sử dụng 1 hàm tên là FilterSendMessage() hàm này có mục đích là gửi 1 message đến kernel-mode minifilter thông qua port đã được khai báo trong tham số context->Port, Tin nhắn này sẽ được truyền xuống các chương trình callback của minifilter. Tuy vào input của tham số &commandMessage mà kernel sẽ gọi tới callback nào để thực hiện. Sau khi thực hiện xong thì biến buffer sẽ nhận dữ liệu trả về của minifilter. Buffer này là 1 mảng các LOG_RECORD structures được định nghĩa bởi Microsoft.

```
typedef struct _LOG_RECORD {

  ULONG Length;
  ULONG SequenceNumber;

  ULONG RecordType;
  ULONG Reserved;

  RECORD_DATA Data;
  WCHAR Name[];
```

```
} LOG_RECORD, *PLOG_RECORD;
```

Cấu trúc data được trả về từ kernel.

```
if (
		RecordData->CallbackMajorId == IRP_MJ_CREATE ||
		RecordData->CallbackMajorId == IRP_MJ_READ ||
		RecordData->CallbackMajorId == IRP_MJ_WRITE ||
		RecordData->CallbackMajorId == IRP_MJ_SET_INFORMATION
) {
		if (wcslen(Name) > 25) {
			if (RecordData->ProcessId != 4) {
				memset(Name, 0, sizeof Name);
				char content[2048];
				memset(content, 0, 2048);
				int i = 23;
				while (Name[i])
				{
					content[i - 21] = Name[i];
					i++;
				}
				const char* temp1 = content;
				int flag = check_extension(temp1);

				if (flag==1)
				{
					Send_Full_Path_Name_Process(temp1);
				}
```

```
                return;
            }
        }
```

Dựa theo các record mà kernnel trả về mà công cụ xác định xem file nào được tạo ra trong hệ thống. Từ đó, chương trình sẽ dùng hàm Send_Full_Path_Name_Process () để gửi thông tin file đó cho ClamAV.

### 4.1.4. Hiện thực chức năng USB

Chức năng usb được viết bằng các sử dụng API trong bộ thu viện <windows.h>

Khai báo tên thư viện

```
#include <windows.h>
```

```
char getRemovableDisk() {
    char drive = '0';

    char szLogicalDrives[MAX_PATH];
    DWORD dwResult = GetLogicalDriveStrings(MAX_PATH, szLogicalDrives);

    string currentDrives = "";

    //cout << dwResult << endl;
    for (int i = 0; i<dwResult; i++)
    {
        if (szLogicalDrives[i]>64 && szLogicalDrives[i]< 90)
        {
            currentDrives.append(1, szLogicalDrives[i]);
```

```
                if (allDrives.find(szLogicalDrives[i]) > 100)
                {
                        drive = szLogicalDrives[i];
                }
        }
    }

    allDrives = currentDrives;

    return drive;
}
```

GetLogicalDriveStrings() : Hàm này sẽ trả về các drives hợp lệ có trong hệ thống.

```
    int main(void) {

        char driveLetter = getRemovableDisk();
        while (1) {
                driveLetter = getRemovableDisk();
                if (driveLetter != '0') {
                        SendData(tem);
                }

                Sleep(1000);
        }
        return 0;
    }
```

Hàm main sử dụng vòng lặp while để khi người dùng cắm usb vào chương trình sẽ phát hiện và gửi tới GUI.

### 4.1.5. Hiện thực chức năng Scanning

Với chức năng này, nhóm sử dụng C++ để hiện thực, và để kết nối với các thành phần khác dùng socket.

```
HANDLE hArcData;
int RHCode, PFCode;
wchar_t CmtBuf[16384];
struct RARHeaderData HeaderData;
struct RAROpenArchiveDataEx OpenArchiveData;

memset(&HeaderData, 0, sizeof(HeaderData));
memset(&OpenArchiveData, 0, sizeof(OpenArchiveData));

OpenArchiveData.ArcName = pathfile;
OpenArchiveData.CmtBufW = CmtBuf;
OpenArchiveData.CmtBufSize = sizeof(CmtBuf) / sizeof(CmtBuf[0]);
OpenArchiveData.OpenMode = RAR_OM_EXTRACT;
OpenArchiveData.UserData = 0;
hArcData = RAROpenArchiveEx(&OpenArchiveData);

if (OpenArchiveData.OpenResult != 0)
	return 1;

while ((RHCode = RARReadHeader(hArcData, &HeaderData)) == 0)
{
	PFCode = RARProcessFile(hArcData, RAR_EXTRACT, destpath, NULL);
	if (PFCode != 0)
		return 1;
```

```
}

RARCloseArchive(hArcData);
return 0;
```

Vì thư viện ClamAV không thể truy cập vào tệp tin nén định dạng .rar. Nên nhóm sử dụng thư viện Winrar để giải nén các tệp tin. Đầu tiên, nhóm sử dụng *RAROpenArchiveEx* và truyền vào đó là địa chỉ một struct. Trong struct đó chứa tham số tới tệp tin cần giải nén. Kế tiếp, nhóm dùng *RARReadHeader* để đọc header tệp tin nén. Và trong vòng lặp while, nhóm dùng *RARProcessFile* để giải nén lần lượt các tệp tin trong đó. Và cuối cùng, để lưu kết quả phải dùng *RARCloseArchive* để đóng kết nối tới tệp tin nén đó.

```
WIN32_FIND_DATA wfda;
	HANDLE hFind = NULL;
	char path[2048];
	sprintf(path, "%s\\*.*", dir);
	if ((hFind = FindFirstFile(path, &wfda)) == INVALID_HANDLE_VALUE)
		return;

	do
	{
		if (strcmp(wfda.cFileName, ".") != 0 && strcmp(wfda.cFileName, "..") != 0)
		{
			sprintf(path, "%s\\%s", dir, wfda.cFileName);
			if (wfda.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
			{
```

```
                    DeleteFileInFolder(path);
                    _rmdir(path);
                }
                else
                    DeleteFile(path);
            }
        } while (FindNextFile(hFind, &wfda));
    FindClose(hFind);
```

Kế tiếp làm hàm xóa các tệp tin đã được giải nén trước đó sau khi gửi các đường dẫn qua socket cho dịch vụ ConnectClamAV. *FindFirstFile* dùng để tìm kiếm tệp tin đầu tiên có trong thư mục được giải nén. Nếu kết quả trả về khác INVALID_HANDLE_VALUE thì sẽ tìm kiếm các tệp tin kế tiếp trong thư mục đó. *DeleteFileInFolder* và *_rmdir* lần lượt được dùng để xóa tệp tin. Và cuối cùng không thể thiếu là kết thúc quá trình tìm kiếm là dùng *FindClose.*

```
    WSADATA wsaData;
    struct addrinfo* result = NULL,
    *ptr = NULL,
    hints;
    int len;
    int iResult;
    SOCKET ConnectSocket = INVALID_SOCKET;
    //char                    sendbuf[]                    =
    "C:\\Users\\vuong\\Desktop\\test\\Win32.Polip.a_infected.exe";

    // Initialize Winsock
    iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);
    if (iResult != 0) {
        printf("WSAStartup failed with error: %d\n", iResult);
```

```
        return 1;
}

ZeroMemory(&hints, sizeof(hints));
hints.ai_family = AF_UNSPEC;
hints.ai_socktype = SOCK_STREAM;
hints.ai_protocol = IPPROTO_TCP;

// Resolve the server address and port
iResult = getaddrinfo("127.0.0.1", DEFAULT_PORT_GUI, &hints,
&result);
if (iResult != 0) {
        printf("getaddrinfo failed with error: %d\n", iResult);
        WSACleanup();
        return 1;
}

// Attempt to connect to an address until one succeeds
for (ptr = result; ptr != NULL; ptr = ptr->ai_next) {
        // Create a SOCKET for connecting to server
        ConnectSocket = socket(ptr->ai_family, ptr->ai_socktype,
        ptr->ai_protocol);
        if (ConnectSocket == INVALID_SOCKET) {
                printf("socket failed with error: %ld\n", WSAGetLastError());
                WSACleanup();
                return 1;
        }

        // Connect to server.
```

```
            iResult = connect(ConnectSocket, ptr->ai_addr, (int)ptr->ai_addrlen);
            if (iResult == SOCKET_ERROR) {
                    closesocket(ConnectSocket);
                    ConnectSocket = INVALID_SOCKET;
                    continue;
            }
            break;
    }

    freeaddrinfo(result);

    if (ConnectSocket == INVALID_SOCKET) {
            printf("Unable to connect to server!\n");
            WSACleanup();
            return 1;
    }

    iResult = send(ConnectSocket, sendbuf, (int)strlen(sendbuf), 0);
    if (iResult == SOCKET_ERROR) {
            printf("send failed with error: %d\n", WSAGetLastError());
            closesocket(ConnectSocket);
            WSACleanup();
    }
    printf("Bytes Sent: %ld\n", iResult);
    // shutdown the connection since no more data will be sent
    iResult = shutdown(ConnectSocket, SD_SEND);
    if (iResult == SOCKET_ERROR) {
            printf("shutdown failed with error: %d\n", WSAGetLastError());
            closesocket(ConnectSocket);
```

```
        WSACleanup();
    }
    closesocket(ConnectSocket);
    WSACleanup();
```

Hàm này có chức năng gửi đường dẫn tệp tin qua dịch vụ *ConnectClamAV***.** Đầu tiên, nhóm sẽ khởi tạo Winsock bằng hàm *WSAStartup***.** Kế tiếp là phân giải địa chỉ và port máy chủ bằng *getaddrinfo***,** nếu thất bị sẽ dùng *WSACleanup* để dọn dẹp và thoát khỏi chương trình. Sau đó, chương trình sẽ cố gắng kết nối tới địa chỉ và port máy chủ trên cho tới khi thành công thông qua vòng lặp. Và trong mỗi vòng lặp sử dụng *socket* để tạo socket, *connect* để kết nối tới máy chủ qua socket được tạo trên. Nếu kết nối thành công, chương trình dùng *send* để gửi dữ liệu cho máy chủ. Bước cuối cùng không thể quên là tắt socket thông qua *closesocket* và dọn dẹp bằng *WSACleanup***.**

```
    char VolumeName[MAX_PATH + 1];
    char DriveLetter[] = "A:\\";

    vector <string> list_disk;
    for (char count = 'A'; count <= 'Z'; ++count)
    {
        DriveLetter[0] = count;

        if (GetVolumeInformation(DriveLetter, VolumeName, MAX_PATH
    + 1, NULL, NULL, NULL, NULL, 0) != FALSE)
        {
            list_disk.push_back(DriveLetter);
        }

    }
```

```
return list_disk;
```

Hàm này sẽ lấy tất cả các ổ đĩa có trong hệ thống bằng vòng lặp for từ A tới Z. Trong mỗi vòng lặp dùng *GetVolumeInformation* để lấy thông tin. Nếu thành công sẽ lưu nó vào danh sách dùng vector để lưu

### 4.1.6. Hiện thực chức năng Update

Với chức năng này, nhóm sử dụng C++ để triển khai, và để giao tiếp với các thành phần khác dùng socket

```
#include "Percent.h"

int main()

{

	Percent percent;

	URLDownloadToFile(NULL,   "http://database.clamav.net/main.cvd",
"D:\\main.cvd", 0, &percent);

	URLDownloadToFile(NULL,   "http://database.clamav.net/daily.cvd",
"D:\\daily.cvd", 0, &percent);

	URLDownloadToFile(NULL,
"http://database.clamav.net/bytecode.cvd", "D:\\bytecode.cvd", 0, &percent);

	return 0;

}
```

Chức năng này sẽ download signature từ trang chủ về thông qua hàm *URLDownloadToFile***.** Ngoài ra, để có thể hiện thị phần tram download thì ta tạo thêm một tệp tin đặt là Percent.h kế thừa từ interface *IBindStatusCallback*

### 4.1.7. Hiện thực chức năng Quarantine

Chức năng này nhóm sử dụng C++ để triển khai và để giao tiếp với các thành phần khác dùng tham số

```
//snapshot toan bo process dang chay
HANDLE hSnapshot = CreateToolhelp32Snapshot(TH32CS_SNAPALL, NULL);
//tao 1 con tro cho toi process
PROCESSENTRY32 pEntry;
pEntry.dwSize = sizeof(pEntry);
//lay process dau tien truyen vao pEntry
BOOL hRes = Process32First(hSnapshot, &pEntry);
while (hRes)
        {
        //so sanh ten duoc truyen vao pEntry va filename
          if (strcmp(pEntry.szExeFile, filename) == 0)
                        {
// neu dung thi truy cap vao process de lay handle process do
HANDLE hProcess = OpenProcess(PROCESS_TERMINATE, 0, (DWORD)pEntry.th32ProcessID);
        if (hProcess != NULL)
                {
                                        // kill process voi ext code la 9
                        TerminateProcess(hProcess, 9);
                        CloseHandle(hProcess);
```

```
                }
        }
//tim process tiep theo
        hRes = Process32Next(hSnapshot, &pEntry);
      }
CloseHandle(hSnapshot);
```

Hàm này có tác dụng kill tiến trình mã độc nếu được phát hiện. Đầu tiên, sử dụng CreateToolhelp32Snapshot để snapshot toàn bộ các tiến trình. Sau đó lần lượt query vào đó. Sử dụng Process32First để đảm bảo lấy được tiến trình đầu tiên. Nếu thành công sẽ đưa vào vòng lặp while để tìm tiến trình mã độc để kill thông qua TerminateProcess .

```
        string temp = string(path) + string(filename);
                ifstream plainFile(temp.c_str(), ios::in || ios::ate | ios::binary);
                char* memblock;
                if (plainFile.is_open())
                {
                        ifstream::pos_type size = plainFile.tellg();
                        memblock = new char[size];
                        plainFile.seekg(0, ios::beg);
                        plainFile.read(memblock, size);
                        plainFile.close();

                string temp1 = ".\\quarantine" + string("\\en_") + string(filename);
                        ofstream encryptFile(temp1.c_str(), ios::out | ios::binary);

                        for (int i = 0; i < size; i++)
```

```
                    memblock[i] ^= atoi(pass);


                encryptFile.write(memblock, size);
                DeleteFileA(temp.c_str());
                delete[] memblock;
                plainFile.close();
                encryptFile.close();
    }
```

Hàm trên có tác dụng mã hóa mã độc. Đầu tiên, chương trình lấy đường dẫn mã độc cần mã hóa, rồi đọc nó lên mảng động trong bộ nhớ heap. Sau đó xor từng byte với chuỗi ngẫu nhiên. Kế tiếp, tệp tin sẽ được ghi từ bộ nhớ vào ổ đĩa và cuối cùng là xóa bộ nhớ động.

### 4.1.8. Hiện thực chức năng Web Proxy

Đối với chức năng này, nhóm sử dụng c# để triển khai thông qua package Titanium để tạo một local proxy trong máy tính.

```
        public void ClassInit()
        {
            _proxyServer = new ProxyServer();
            _proxyServer.CertificateManager.TrustRootCertificate(true);
var explicitEndPoint = new ExplicitProxyEndPoint(System.Net.IPAddress.Any,
18882, true);
        _proxyServer.AddEndPoint(explicitEndPoint);
        _proxyServer.Start();
        _proxyServer.SetAsSystemHttpProxy(explicitEndPoint);
        _proxyServer.SetAsSystemHttpsProxy(explicitEndPoint);
       _proxyServer.BeforeRequest += OnRequestCaptureTrafficEventHandler;
```

```
_proxyServer.BeforeResponse += OnResponseCaptureTrafficEventHandler;
    }
```

Đầu tiên, chương trình sẽ khởi tạo đối tượng proxy. Lúc này, ta cần override hàm hàm quan trọng *OnRequestCaptureTrafficEventHandler* và *OnResponseCaptureTrafficEventHandler*.

```
            await Task.Run(() => {
                MatchCollection matchCollection =
    r.Matches(e.HttpClient.Request.Url);
                if (matchCollection.Count > 0)
                {
                    string temp = matchCollection[0].Value;
                    temp = temp.Substring(3, temp.Length - 6);
                    temp.Replace('+', ' ');
                    //s.Substring(s.Length - 5 ,4);
                    Console.OutputEncoding = Encoding.UTF8;
                    Console.WriteLine(Uri.UnescapeDataString(temp));
                }
                if
    (e.HttpClient.Request.RequestUri.AbsoluteUri.Contains("vlxx.tv"))
                {
                    e.Ok("<!DOCTYPE html>" +
                        "<html><body><h1>" +
                        "Website Blocked" +
                        "</h1>" +
                        "<p>Blocked :)</p>" +
                        "</body>" +
                        "</html>");
```

```
        }
    });
```

Với *OnRequestCaptureTrafficEventHandler* thì proxy sẽ lọc, sửa đổi luồng traffic đi ra của máy tính qua từng packet. Vì packet đi qua proxy đã và chưa mã hóa đối với https thì việc xem địa chỉ web là hoàn toàn khả thi, còn đối với http thì bình thường.

### 4.1.9. Hiện thực chức năng Dịch vụ (ClamAV)

Đối với chức năng này, nhóm sử dụng thư viện ClamAV tên là *libclamav* và dùng C++ để triển khai. Để giao tiếp với các thành phần khác, chương trình sẽ dùng socket.

```
struct cl_engine* engine;
unsigned int sigs = 0;
int ret;

if ((ret = cl_init(CL_INIT_DEFAULT)) != CL_SUCCESS)
{
   cout << cl_strerror(ret);
   return 1;
}

if (!(engine = cl_engine_new()))
{
   cout << "Can't create new engine";
   return 1;
}

ret = cl_load(cl_retdbdir(), engine, &sigs, CL_DB_STDOPT);
```

```
    if (ret != CL_SUCCESS)
    {
        cout << cl_strerror(ret);
        cl_engine_free(engine);
        return 1;
    }

    if ((ret = cl_engine_compile(engine)) != CL_SUCCESS)
    {
        cout << cl_strerror(ret);
        cl_engine_free(engine);
        return 1;
    }

    struct cl_stat dbstat;
    memset(&dbstat, 0, sizeof(struct cl_stat));
    cl_statinidir(cl_retdbdir(), &dbstat);
    if (cl_statchkdir(&dbstat) == 1)
    {
        cout << "Out up date";
        cl_statfree(&dbstat);
        cl_statinidir(cl_retdbdir(), &dbstat);
    }

    cl_scan_options scanoptions;
    scanoptions.general                                          =
static_cast<uint32_t>(CL_SCAN_GENERAL_ALLMATCHES                 |
CL_SCAN_GENERAL_COLLECT_METADATA                                 |
CL_SCAN_GENERAL_HEURISTICS                                       |
```

```
        CL_SCAN_GENERAL_HEURISTIC_PRECEDENCE);
            scanoptions.parse                                                =
        static_cast<uint32_t>(CL_SCAN_PARSE_ARCHIVE                          |
        CL_SCAN_PARSE_ELF          |          CL_SCAN_PARSE_PDF          |
        CL_SCAN_PARSE_SWF          |          CL_SCAN_PARSE_HWP3         |
        CL_SCAN_PARSE_XMLDOCS      |      CL_SCAN_PARSE_MAIL             |
        CL_SCAN_PARSE_OLE2         |          CL_SCAN_PARSE_HTML         |
        CL_SCAN_PARSE_PE);
            scanoptions.heuristic                                            =
        static_cast<uint32_t>(CL_SCAN_HEURISTIC_MACROS                       |
        CL_SCAN_HEURISTIC_ENCRYPTED_ARCHIVE);
            scanoptions.mail                                                 =
        static_cast<uint32_t>(CL_SCAN_MAIL_PARTIAL_MESSAGE);
            scanoptions.dev = 0;
            char filename[] = "D:\\Download\\Compressed\\mbr.bin";
            const char* virname;
            if ((ret = cl_scanfile(filename, &virname, NULL, engine, &scanoptions))
        == CL_VIRUS)
                cout << "Detect: " << virname << endl;
            else if (ret != CL_CLEAN)
                cout << cl_strerror(ret) << endl;
            else
                cout << "Clean";


            cl_engine_free(engine);
            system("pause");
    return 0;
```

Đầu tiên, để chạy được chương trình thì cần tạo một thư mục tên là *db* trong đó chứa signature ClamAV, ta có thể thực hiện hành động này thông qua giao diện tương tác. Khi chương trình chạy, đầu tiên nó sẽ dùng *cl_init* để khởi tạo engine của ClamAV. Sau đó, chương trình sẽ tạo mới engine bằng *cl_engine_new*. Sau khi tạo mới thành công, chương trình sẽ load signature vào bộ nhớ bang *cl_load*, và để lấy được đường dẫn signature dùng *cl_retdbdir*. Nếu thất bại thì sẽ xuất ra lỗi và giải phóng engine thông qua *cl_engine_free*. Còn thành công thì sẽ compile engine thông qua *cl_engine_compile*. Kế tiếp, chương trình sẽ kiểm tra xem signature có lỗi thời hay không, nếu không thì sẽ tiến hành quét tệp tin thông qua *cl_scanfile*.

## 4.2. Yêu cầu hệ thống

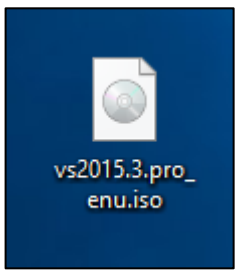
- Windows 10 64 bit ( Phiên bản 10586)
- CPU 2 core
- RAM 2 GB
- Ổ cứng 60GB
- Máy có kết nối internet

## 4.3. Cài đặt môi trường thử nghiệm

- Cài đặt OpenSSL 1.1.1d 32 bit .


Hình 4.1: Tệp tin cài đặt OpenSLL

- Cài đặt Visual Studio Community 2015 3.

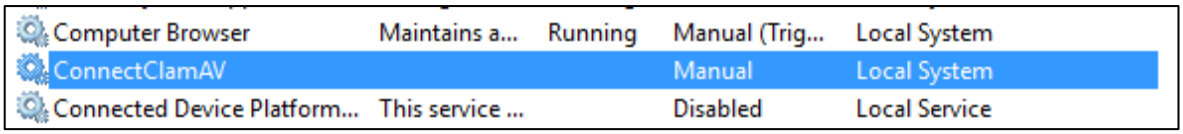



Hình 4.2: Tệp tin cài đặt Visual 2015

- Tiến hành cài đặt dịch vụ ConnectClamAV bằng lệnh SC CREATE "ConnectClamAV" binpath= "đường dẫn tới tệp tin cần cài đặt là dịch vụ"

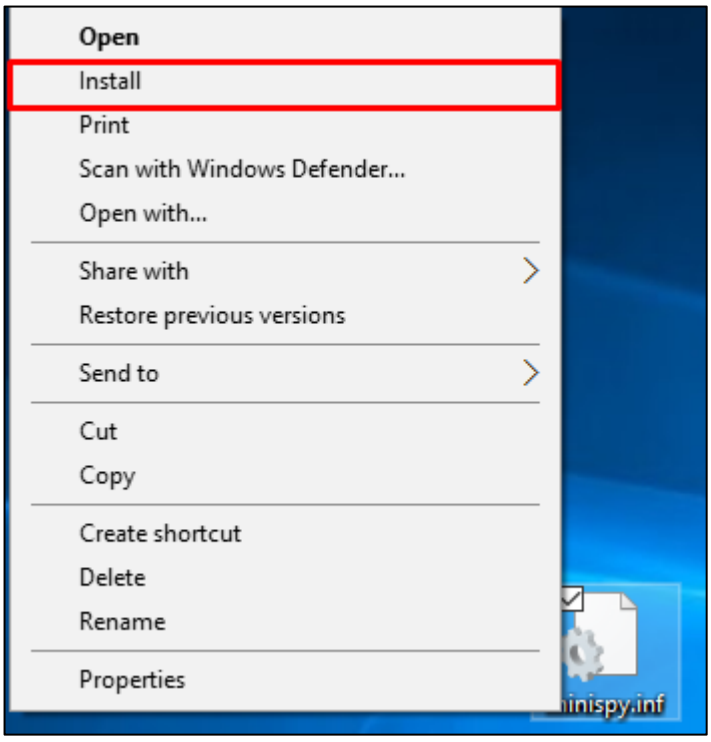

Hình 4.3: Câu lệnh để kết nối ClamAV

Kết quả khi cài đặt thành công:



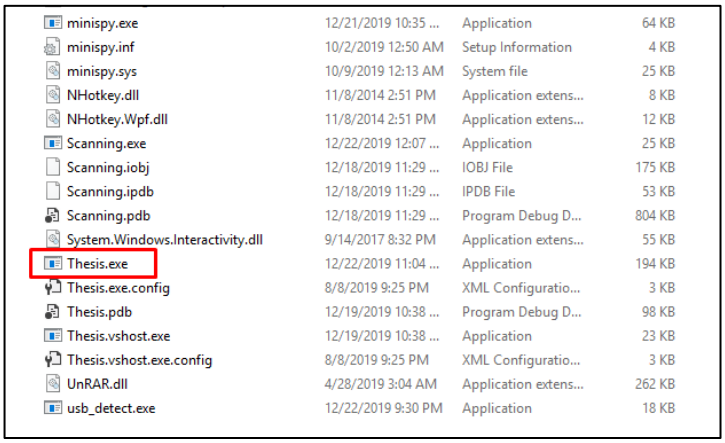

Hình 4.4: Dịch vụ đã được cài đặt vào hệ thống

- Cài đặt  driver minispy:




Hình 4.5: Cài đặt driver

- Build lần lượt các Mô-đun Scanning, USB, Update, Quarantine, WebProxy rồi để cùng thư mục với GUI .
- Khởi động giao diện bằng cách nhấn hai lần vào tệp tin Thesis.exe




Hình 4.6: Tệp tin thực thi công cụ

## 4.4. Các kết quả hiện thực và kiểm thử tính năng

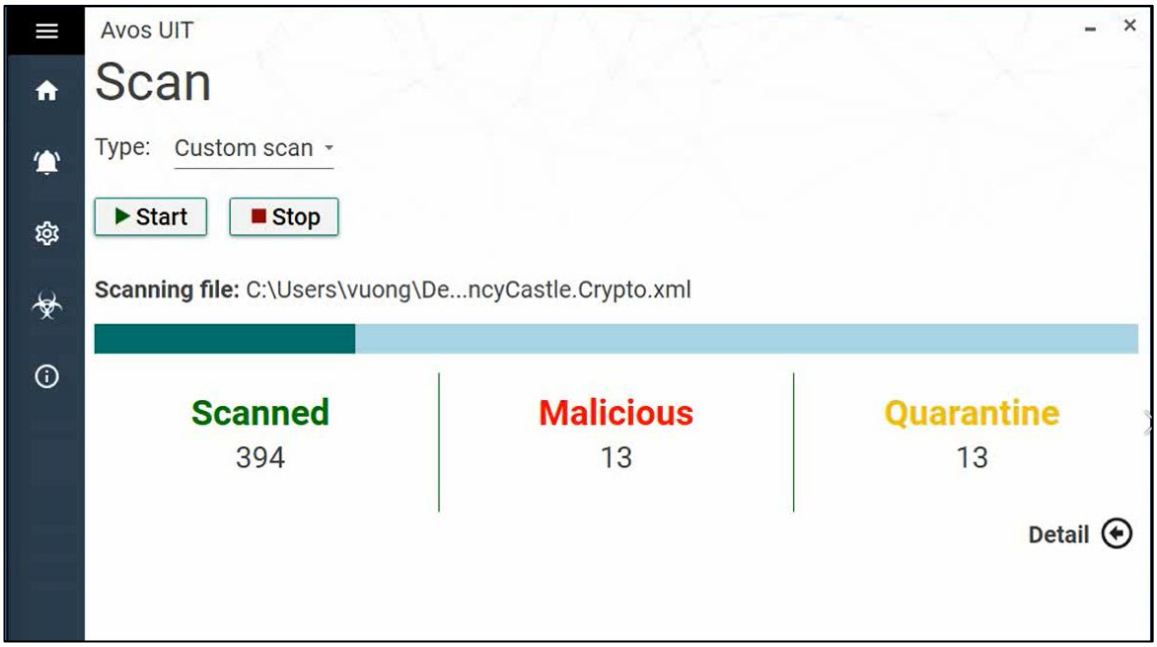
### 4.4.1. Môi trường thử nghiệm công cụ

- Windows 10 (Phiên bản 10586)
- CPU 2 core
- RAM 4GB
- Ổ cứng 60GB
- Các thư viện cần thiết (OpenSSL, thư viện Visual Studio)

### 4.4.2. Kết quả kiểm thử chức năng Scanning

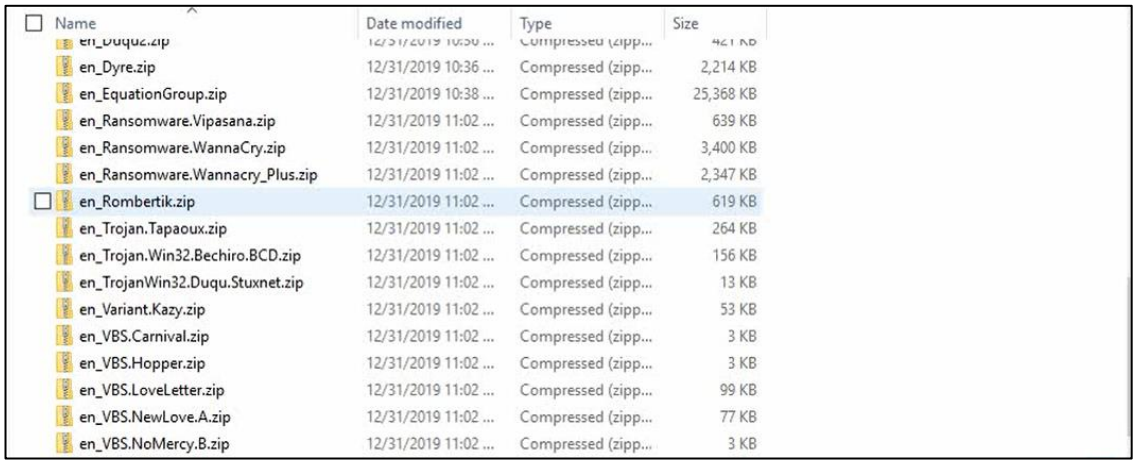
Kịch bản kiểm thử: Ở chức năng này, công cụ sẽ phát hiện mã độc bằng các scan hệ thống. Người dùng vào mục scan. Chọn tùy chọn custom scan .Chọn thư mục muốn scan. Nhấn Button để bắt đầu quá trình scan.

Kết quả: Công cụ đã phát hiện mã độc, di chuyển mã độc vào thư mục cách ly.

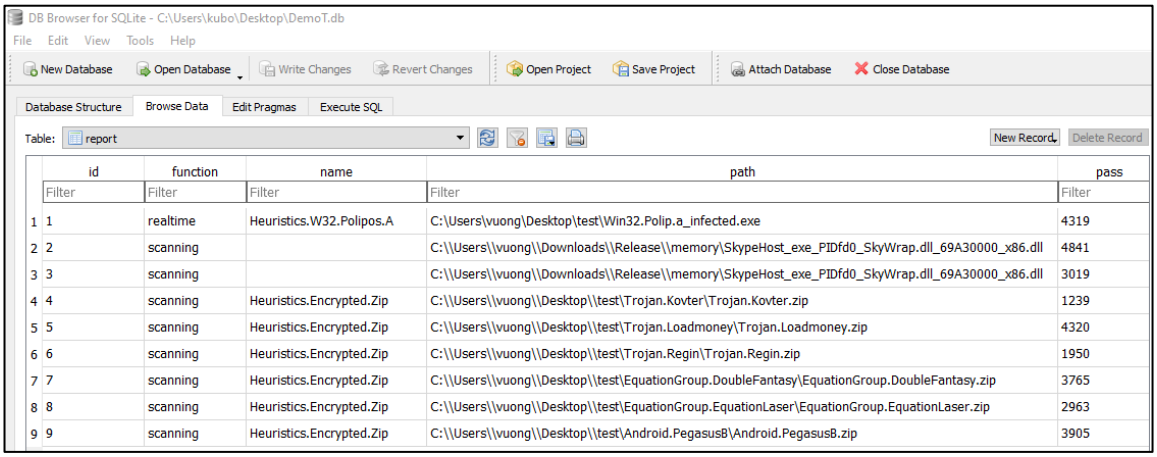



Hình 4.7: Giao diện chức năng Scan


Hình 4.8: Kết quả sau khi scan




Hình 4.9: Kết quả ở mục cách ly

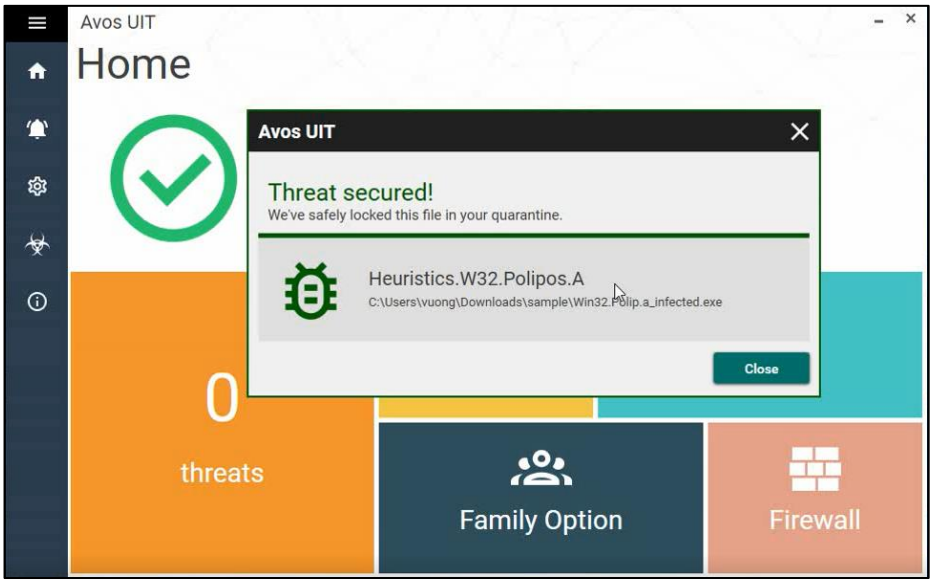



Hình 4.10: Kết quả được lưu vào cơ sở dữ liệu

### 4.4.3. Kết quả kiểm thử chức năng Realtime

Kịch bản kiểm thử: Ở chức năng này, công cụ sẽ phát hiện mã độc theo thời gian thực. Khi người dùng mở tệp tin mã độc, công cụ sẽ hiện thông báo. Sau đó cách ly mã độc này vào thư mục cách ly.

Kết quả: Công cụ đã xuất hiện thông báo và di chuyển mã độc vào thư mục cách ly.

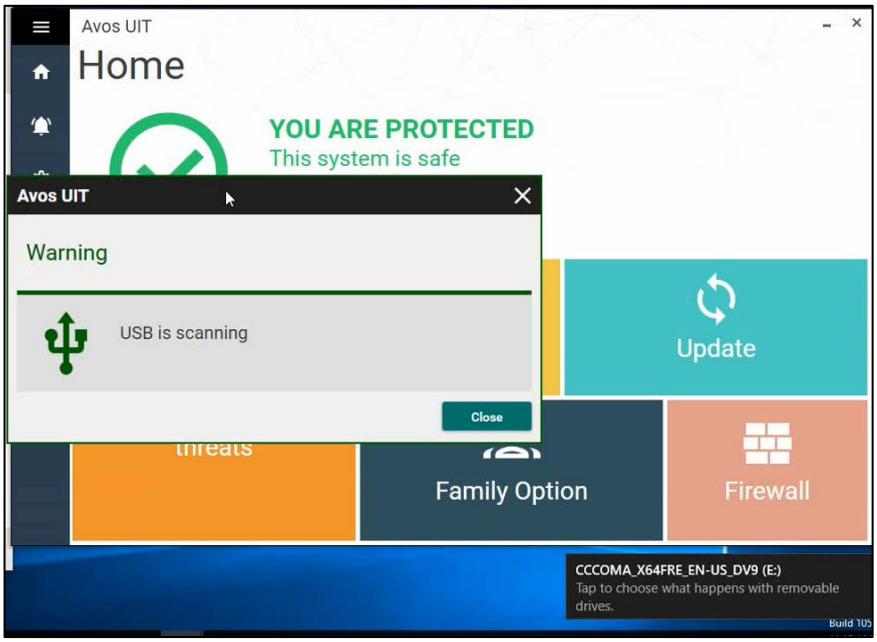



Hình 4.11: Kết quả chức năng Realtime

### 4.4.4. Kết quả kiểm thử chức năng USB

Kịch bản: Công cụ sẽ quét usb mỗi khi được cắm vào máy. Công cụ sẽ hiện kết quả trong mục scanning.

Kết quả: Công cụ phát hiện mã độc tồn tại trong usb.



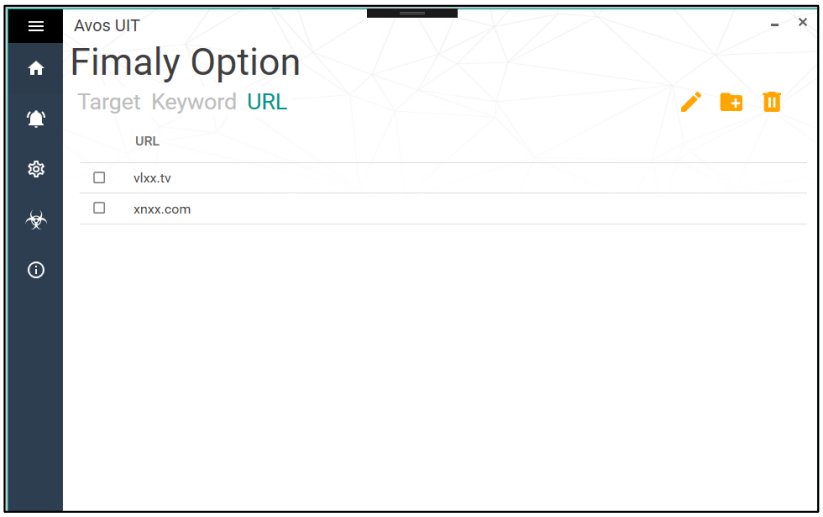

Hình 4.12: Hiện thực chức năng USB


Hình 4.13: Kết quả hiện thực chức năng USB

### 4.4.5. Kết quả thử nghiệm chức năng Family Options

Kịch bản: Vào mục Family Option, tại tab URL thêm các địa chỉ mà người dùng muốn hạn chế truy cập thông qua biểu tượng thêm mới. Sau đó công cụ sẽ chặn không cho truy cập những địa chỉ trên. Tại tab Target, thì bước thêm mới cũng như tab URL, lúc này công cụ sẽ cấm người dùng thực thi các chương trình chỉ định như: Chrome, Zalo,…

Kết quả: Công cụ đã chặn thành công các địa chỉ đã được thêm vào.



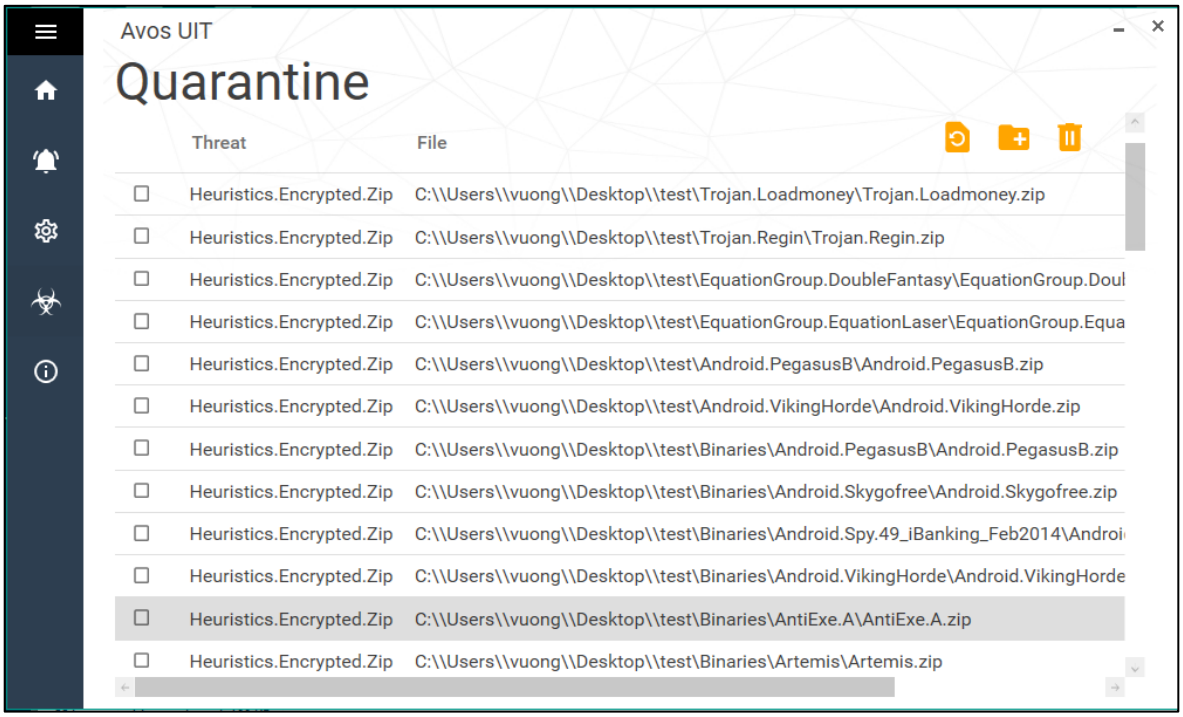

Hình 4.14: Hiện thực chức năng Family Options


Hình 4.15: Kết quả hiện thực chức năng Family Optiops

### 4.4.6. Kết quả thử nghiệm chức năng Quarantine

Kịch bản: Kiểm tra danh sách các tệp tin mã độc được cách ly.

Kết quả: Công cụ đã cách ly thành công.

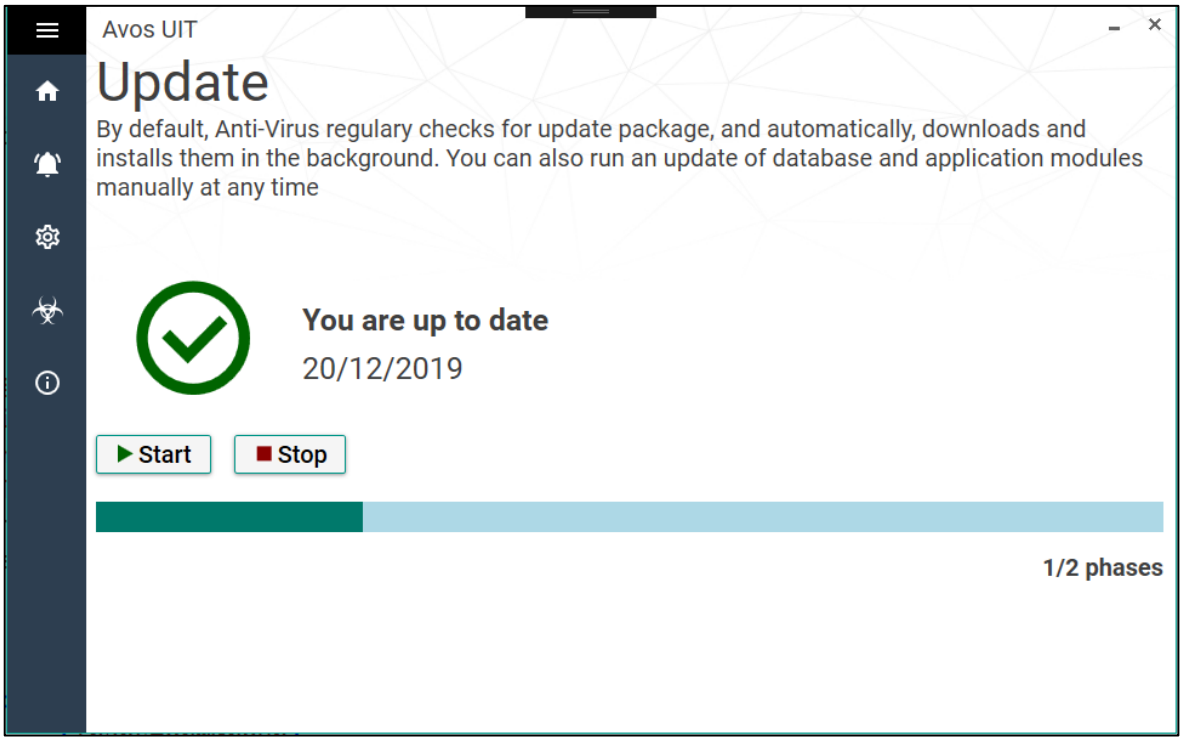



Hình 4.16: Kết quả chức năng Quarantine

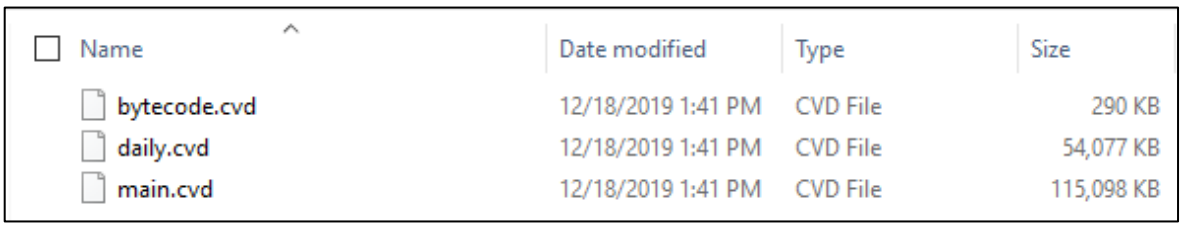
### 4.4.7. Kết quả kiểm thử chức năng Update

Kịch bản: Vào thư mục update của công cụ. Nhấn Button start để bắt đầu update.

Kết quả: Công cụ update thành công.




Hình 4.17: Hiện thực chức năng Update

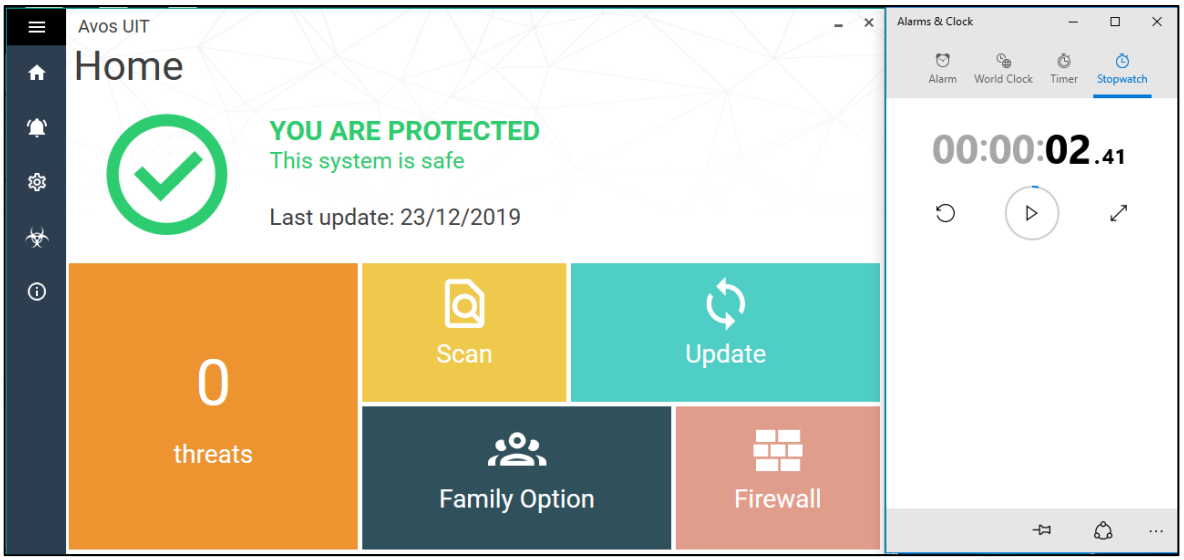



Hình 4.18: Kết quả hiện thực chức năng Update

## 4.5. Đánh giá hiệu suất công cụ.

### 4.5.1. Đánh giá hiệu suất khi công cụ được thực thi lên.

Nhóm đã tiến hành do thời gian khi công cụ được tải lên, dưới đây là hình ảnh minh chứng:




Hình 4.19: Thời gian khi mới thực thi công cụ

| | CPU | RAM | DISK |
|---|---|---|---|
| Scanning | 0% | 0.8 MB | 0 MB/s |
| Minispy | 0% | 0.5 MB | 0 MB/s |
| WebProxy | 0% | 3.8 MB | 0 MB/s |
| USB scan | 0% | 0.8 MB | 0 MB/s |
| GUI | 1.8% | 39.5 MB | 0 MB/s |

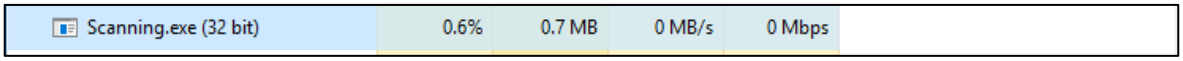
Bảng 4.3: Bảng hiệu suất các chức năng ở trạng thái bình thường

### 4.5.2. Đánh giá chức năng Scan

| | CPU | RAM | DISK |
|---|---|---|---|
| Scanning | 0.6 % | 0.7 MB | 0 MB/s |

Bảng 4.4: Bảng hiệu suất chức năng Scan ở trạng thái Scanning

Minh chứng:

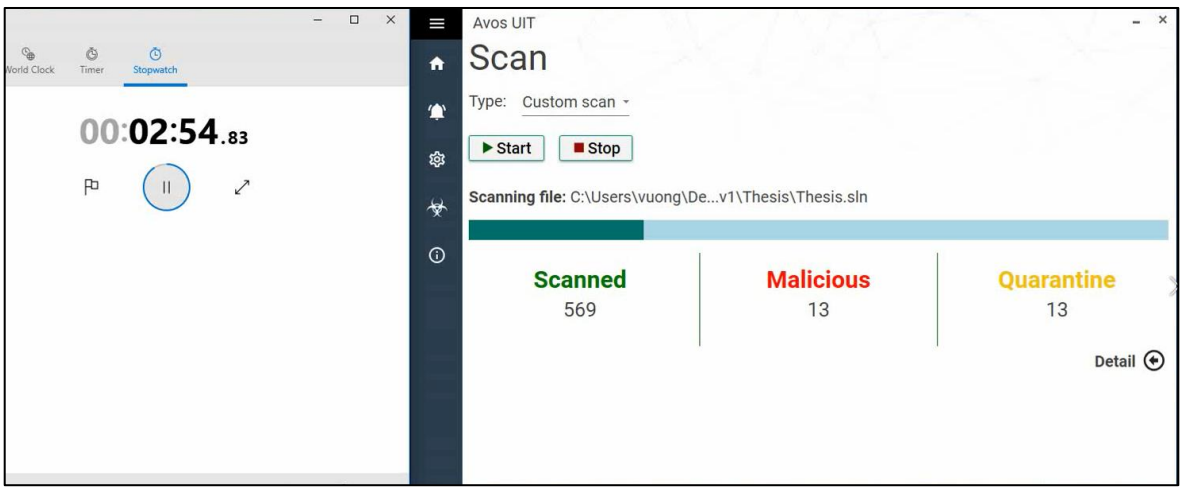



Hình 4.20: Minh chứng hiệu suất chức năng Scan

| **Tên chức năng** | **Tổng số tệp tin đã quét** | **Tổng số tệp tin có hại** | **Tổng số tệp tin vô hại** | **Thời gian hoàn thành** |
|---|---|---|---|---|
| Full scan | 162939 | 7 | 162932 | 9:17:54 (s) |
| Custom scan | 569 | 13 | 556 | 2:54 (s) |
| Usb scan | 604 | 4 | 600 | 1:07 (s) |

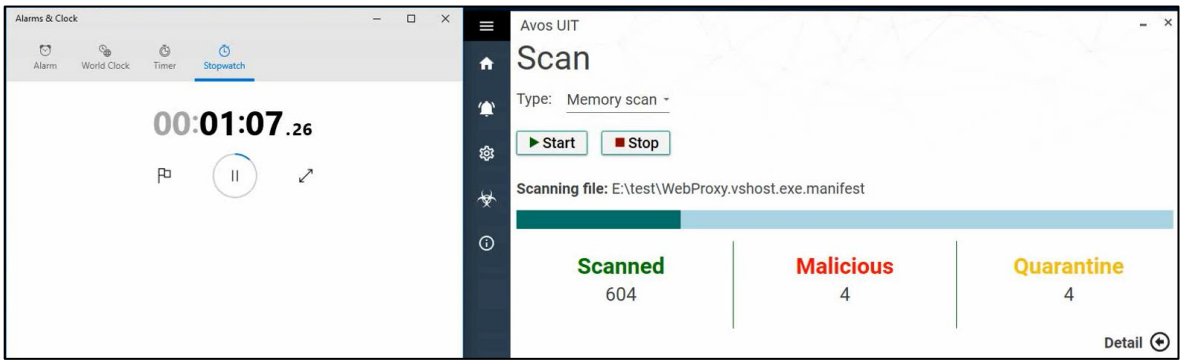
Bảng 4.5: Bảng kết quả scan các tùy chọn trong Scan

Minh chứng:




Hình 4.21: Thời gian của chức năng Custom Scan


Hình 4.22: Thời gian chức năng Full Scan

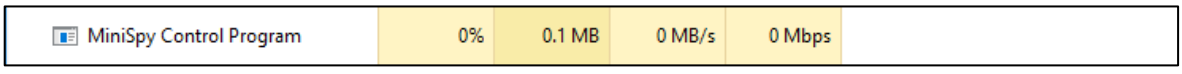



Hình 4.23: Thời gian chức năng USB Scan

### 4.5.3. Đánh giá chức năng Realtime

| | CPU | RAM | DISK |
|---|---|---|---|
| Minispy | 0 % | 0.1 MB | 0 MB/s |

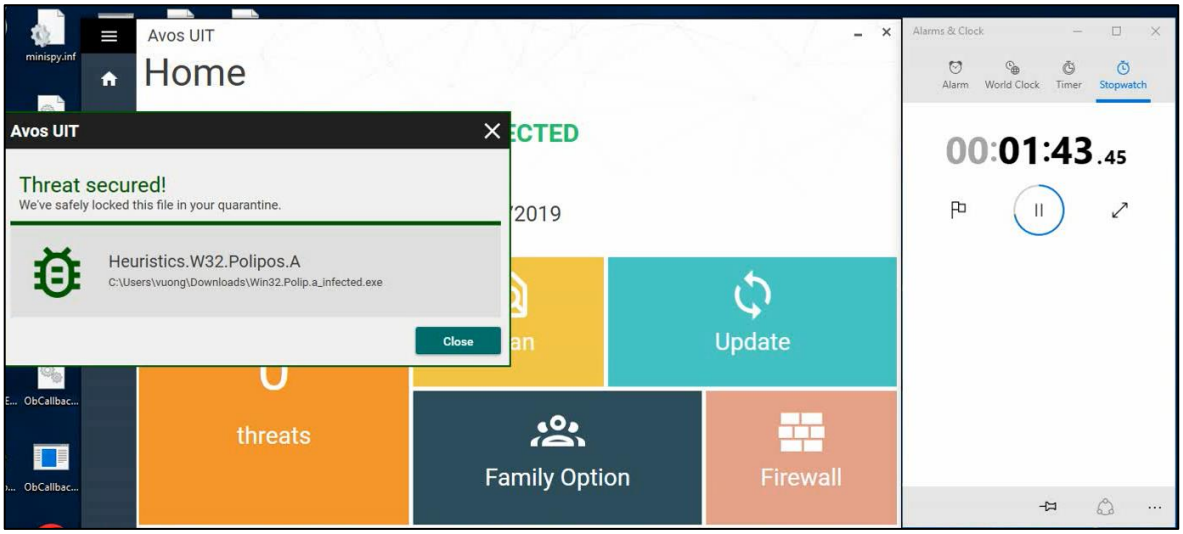
Minh chứng:



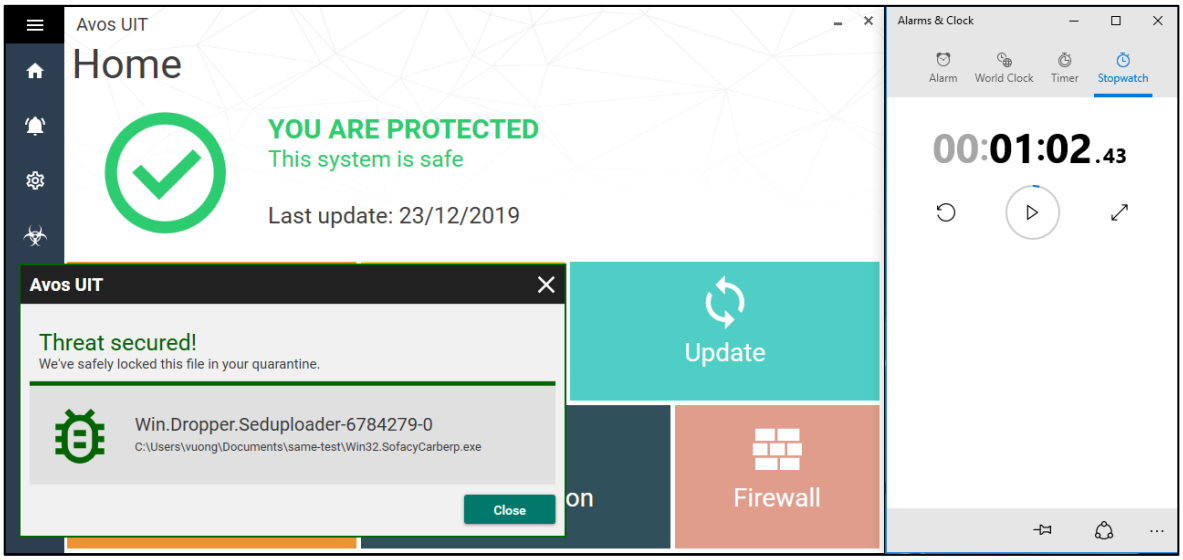
| Tên tệp có hại | Thời gian phát hiện (s) |
|---|---|
| Win32.Polip.a_infected.exe | 1:43 |
| scanslam.exe | 2:11 |
| Win32.DarkTequila.exe | 1:15 |
| Win32.GravityRAT.exe | 1:04 |
| Win32.SofacyCarberp.exe | 1:02 |

Bảng 4.6: Bảng hiệu suất chức năng Realtime

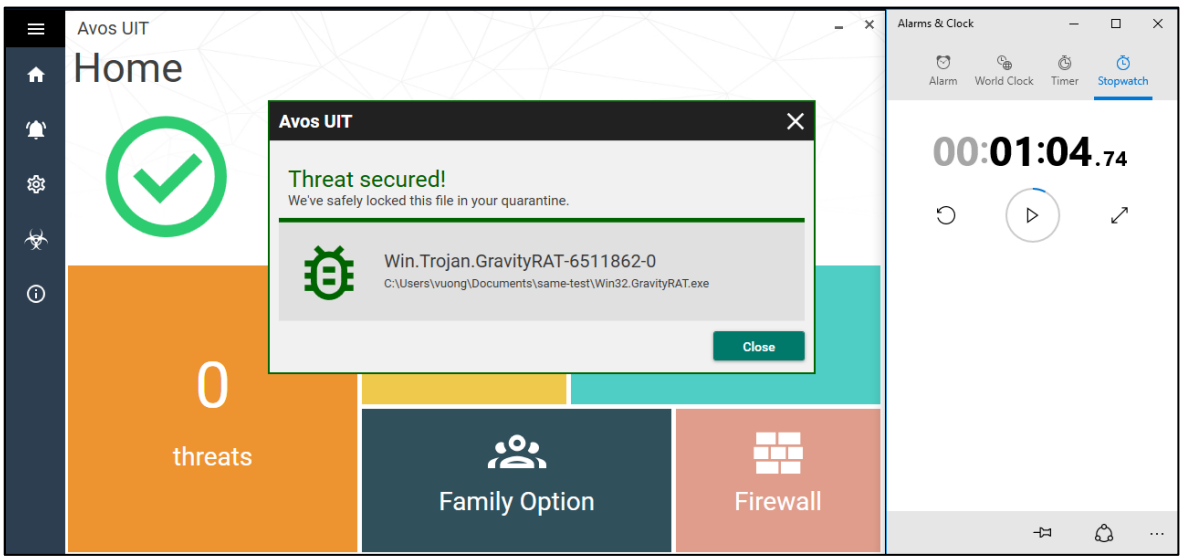
Mình chứng:




Hình 4.24: Thời gian của chức năng Realtime(Polipos.A)




Hình 4.25: Thời gian của chức năng Realtime(Sofacy)



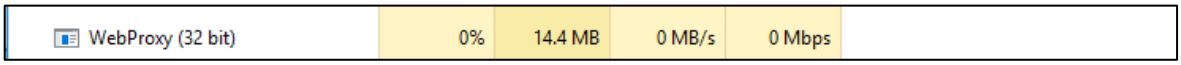

Hình 4.26: Thời gian của chức năng Realtime(GravityRAT)

### 4.5.4. Đánh giá chức năng Family Options

| | CPU | RAM | DISK |
|---|---|---|---|
| WebProxy | 0 % | 14.4 MB | 0 MB/s |

Bảng 4.7: Bảng hiệu suất chức năng Family Options

Minh chứng:




Hình 4.27: Minh chứng chức năng Family Options

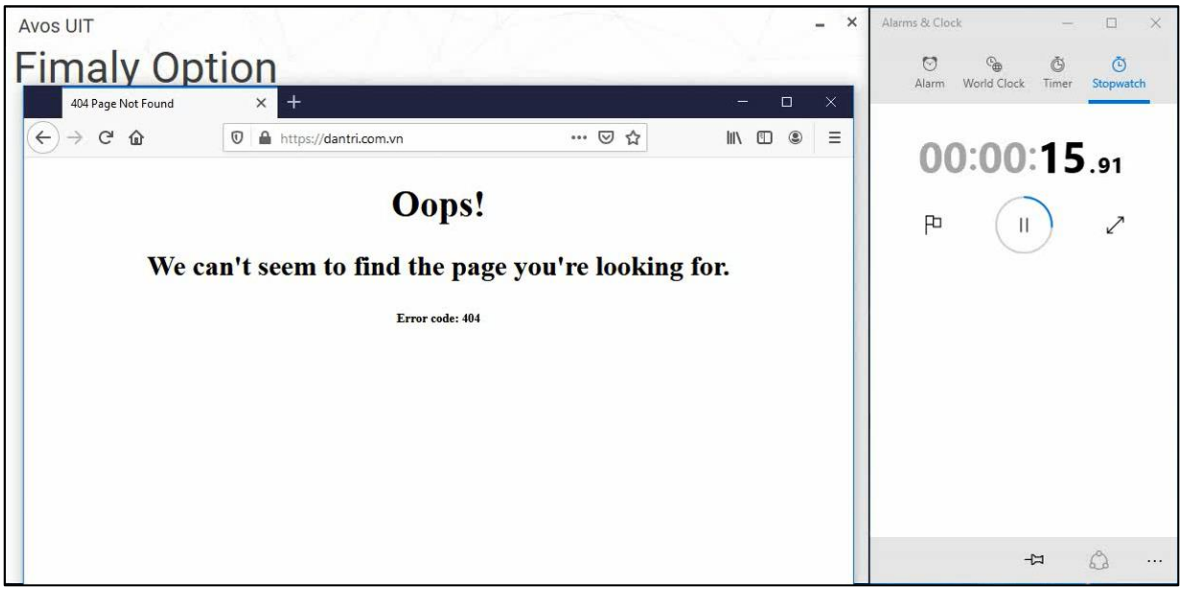
| Tên URL | Kết quả | Thời gian hoàn thành (s) |
|---|---|---|
| dantri.com.vn | Pass | 15 |
| baomoi.com | Pass | 17 |
| baomoi.com | Pass | 13 |
| tintuconline.com.vn | Pass | 14 |
| news.zing.vn | Pass | 12 |
| soha.vn | Pass | 11 |
| danviet.vn | Pass | 14 |

Bảng 4.8: Bảng kết quả thời gian hoàn thành chặn URL

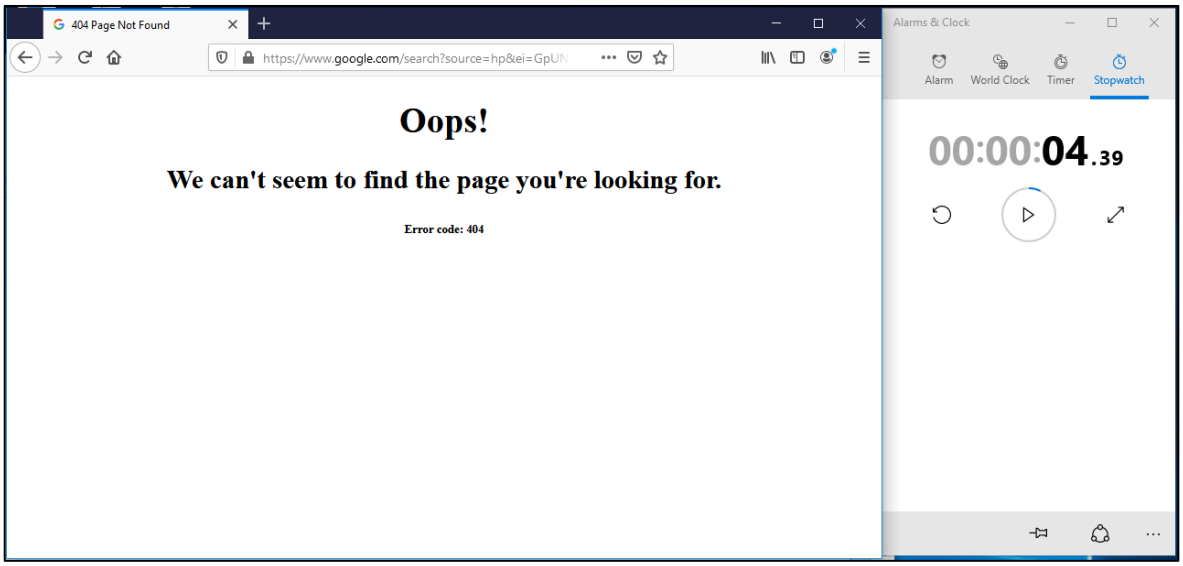
| Tên từ khóa | Kết quả | Thời gian hoàn thành (s) |
|---|---|---|
| Game | Pass | 4 |
| Phim | Pass | 3 |
| Hài | Pass | 4 |
| Bạo lực | Pass | 3 |
| Đồi trụy | Pass | 3 |

Bảng 4.9: Bảng kết quả chặn từ khóa

Minh chứng:




Hình 4.28: Thời gian chặn URL




Hình 4.29: Thời gian chặn từ khóa

# Chương 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 5.1. Kết luận

Trong phiên bản lần này, Avos UIT có khả năng:

- Phát hiện mã độc theo thời gian thực.
- Quét toàn bộ, một thư mục chỉ định và memory.
- Tất cả mã độc khi phát hiện sẽ được di chuyển vào vùng cách ly.
- Có khả năng xuất báo cáo.
- Chặn các nội dung không mong muốn: địa chỉ web, từ khóa lướt web, chương trình thực thi.
- Có thể cập nhật signature.
- Phát hiện USB được cắm vào máy tính.

So sánh với mã nguồn mở an tivirus khác, Avos UIT có giao diện tương tác dễ sử dụng, sử dụng MiniFilter driver một kĩ thuật được hỗ trợ bởi Windows để giám sát, theo dõi các tiến trình trong hệ thống, USB được cắm vào máy, chặn nội dung không mong muốn: địa chỉ web, từ khóa lướt web, chương trình thực thi.

## 5.2. Hướng phát triển

Tuy nhiên, Avos UIT vẫn còn tồn đọng một số hạn chế sau:

- Hiệu năng và tốc độ nhận diện vẫn còn thấp
- Trải nghiệm truy cập web khi bật tính năng chặn nội dung không mong muốn vẫn còn chậm.

Trong tương lai, nhóm sẽ cải thiện hiệu năng cũng như tốc độ quét nhanh hơn, thêm một số chức năng mới hơn, chiếm ít tái nguyên ít nhất có thể. Tích hợp tính năng sandbox. Triển khai toàn bộ tính năng bằng C++ kể cả GUI. Chạy trên được nhiều nền tảng của Windows.

# Chương 6. TÀI LIỆU THAM KHẢO

[1] Central University of Rajasthan,"How Anti-virus Software Works??". Availabel:
https://www.researchgate.net/publication/308800880_How_Anti
rus_Software_Works , 2016

[2] Istanbul Technical University,"Ransomware, Detection and Prevention Techniques, Cyber Security, Malware Analysis". Availabel:
https://www.researchgate.net/publication/326191046_Ransomware_Detection_and_Prevention_Techniques_Cyber_Security_Malware_Analysis, 2017.

[3] The Hong Kong Polytechnic University,"Design and implementation of a malware detectionsystem based on network behavior".Availabel: https://www.researchgate.net/publication/264802094_Design_and_implementation_of_a_malware_detection_system_based_on_network_behavior, 2015.

[4] Dennis Turpitlka, "How to Write Windows Drivers" [Online]. Available : https://www.electronicdesign.com/windows/how-write-windows-drivers Accessed , 2016.

[5] Tigzy,"Making an antivirus engine: the guidelines" [Online]. Available: https://www.adlice.com/making-an-antivirus-engine-the- guidelines/ Accessed : Adlice software blog 2013.

[6] Royal Hollyoway University of London,"Understanding behavioural detection of antivirus" - [Online] Available:

https://pdfs.semanticscholar.org/515b/f8dd74147dc4e46a68e8e098dbc1171a9747.pdf Accessed: Technical report RHHUL-ISG-2016-10 5 April 2016.

[7] Future Technology Device International Ltd,"How to detect the connection and removal of usb devices on a system" [Online] Available:

https://www.ftdichip.com/Support/Documents/AppNotes/AN_152_Detecting_USB_%20Device_Insertion_and_Removal.pdf , 20-09-2010.

[8] “Microsoft “PsSetCreateProcessNotifyRoutineEx routine” [Online] Available:

https://docs.microsoft.com/en-us/windows-hardware/drivers/ddi/content/ntddk/nf-ntddk-pssetcreateprocessnotifyroutineex?redirectedfrom=MSDN, Accessed: Jan 2,2017.

[9] Rohitab.com, Get process name from PID in kelnel-mode driver [Online] Available :

http://www.rohitab.com/discuss/topic/40560-get-process-name-form-pid-in-kernel-mode-driver/ , accessed : Dec 10 2013.

[10] “Microsoft “WFP architecture” 2017 . [Online]. Available: https://docs.microsoft.com/en-us/windows/win32/fwp/windows-filtering-platform-architecture-overview?redirectedfrom=MSDN Accessed : Jan. 2, 2017.