

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**



BÁO CÁO ĐỒ ÁN CHUYÊN NGÀNH

Đề tài: Research and deploy NG-Antivirus solution

GVHD: ThS. Nguyễn Duy

Nhóm:

Nguyễn Đại Nghĩa 21521182

Hoàng Gia Bảo 21521848

MỤC LỤC

I.	GIỚI THIỆU CHUNG	2
II.	PHẠM VI THỰC HIỆN	3
III.	KIẾN TRÚC HỆ THỐNG ĐỀ XUẤT	5
IV.	PHƯƠNG PHÁP TRIỂN KHAI	6
V.	CÔNG CỤ SỬ DỤNG	8
VI.	DEMO	11
VII.	KẾT LUẬN	43
VIII.	TÀI LIỆU THAM KHẢO	43

I. GIỚI THIỆU CHUNG

1. Bối cảnh:

Trong thời đại công nghệ số ngày càng phát triển, các mối đe dọa an ninh mạng đang trở nên phức tạp và nguy hiểm hơn bao giờ hết. Các kỹ thuật tấn công hiện đại như tấn công không dùng tệp (fileless attacks), mã độc nhúng trong bộ nhớ, và scripting độc hại thông qua PowerShell đang khiến các phần mềm antivirus (AV) truyền thống dần mất hiệu quả. Những giải pháp truyền thống này chỉ tập trung vào việc phát hiện mã độc dựa trên chữ ký (signature-based detection) hoặc heuristic, và không đủ khả năng chống lại các cuộc tấn công phức tạp, đa giai đoạn.

Trong bối cảnh này, Next-Generation Antivirus (NGAV) ra đời như một bước tiến quan trọng trong lĩnh vực bảo mật điểm cuối (endpoint security). NGAV sử dụng công nghệ trí tuệ nhân tạo (AI), học máy (machine learning), và phân tích hành vi để phát hiện và ngăn chặn các cuộc tấn công tinh vi. Không chỉ dừng lại ở việc phát hiện mã độc, NGAV còn tập trung vào việc giám sát và phân tích các sự kiện liên quan đến tệp, quy trình, ứng dụng và kết nối mạng, từ đó xây dựng một lớp bảo vệ toàn diện và chủ động.

Một xu hướng nổi bật trong lĩnh vực này là sự tích hợp giữa NGAV với các giải pháp Endpoint Detection and Response (EDR), chẳng hạn như Wazuh, nhằm nâng cao khả năng phát hiện và phản ứng với các mối đe dọa trong thời gian thực. Khi kết hợp với các công cụ như ClamAV, một công cụ quét mã độc mã nguồn mở, các tổ chức có thể tận dụng sức mạnh của cả hai hệ thống để xây dựng một giải pháp bảo mật điểm cuối hiệu quả, toàn diện và tiết kiệm chi phí.

Trong đề tài này, chúng em tập trung nghiên cứu và triển khai NGAV như một giải pháp bảo mật tiên tiến, kết hợp với các công cụ mã nguồn mở để tối ưu hóa hiệu quả bảo vệ. Đề tài không chỉ cung cấp cái nhìn tổng quan về NGAV mà còn thử nghiệm khả năng tích hợp giữa NGAV, ClamAV, và Wazuh, nhằm đánh giá tính hiệu quả của giải pháp trong môi trường thực tế.

2. Thách thức:

Việc triển khai một giải pháp **Next-Generation Antivirus (NGAV)** đặt ra nhiều thách thức, bao gồm:

❖ Phát hiện các mối đe dọa không dùng tệp (fileless attacks):

- Tấn công không dùng tệp ngày càng phổ biến, khi mã độc chỉ tồn tại trong bộ nhớ mà không ghi lên ổ đĩa, khiến các hệ thống bảo mật truyền thống khó phát hiện.
- Yêu cầu phân tích hành vi chuyên sâu và giám sát thời gian thực để phát hiện các hoạt động đáng ngờ.

- ❖ **Khả năng nhận diện các mối đe dọa mới (zero-day threats):**
 - Mã độc mới chưa có chữ ký trong cơ sở dữ liệu của các công cụ truyền thống, đòi hỏi hệ thống phải sử dụng mô hình học máy và trí tuệ nhân tạo để dự đoán.
- ❖ **Tích hợp và tương thích công nghệ:**
 - Kết hợp ClamAV, Wazuh, và các công cụ như CAPE Sandbox đòi hỏi phải xây dựng cơ chế giao tiếp và xử lý dữ liệu hiệu quả, đồng thời đảm bảo các thành phần hoạt động trơn tru trên cùng một hệ thống.
- ❖ **Tối ưu hiệu suất và tài nguyên:**
 - Phân tích hành vi, học máy và quét tệp liên tục tiêu tốn tài nguyên hệ thống, dẫn đến khả năng ảnh hưởng đến hiệu năng máy tính trong môi trường thực tế.
- ❖ **Xử lý dữ liệu lớn và phức tạp:**
 - Các mô hình học máy cần lượng dữ liệu lớn để huấn luyện và dự đoán chính xác. Dữ liệu này thường đa dạng, phức tạp và yêu cầu phải được xử lý trước khi sử dụng.
- ❖ **Đảm bảo độ chính xác cao:**
 - Hạn chế tỷ lệ **False Positive** (phát hiện nhầm mối đe dọa không tồn tại) và **False Negative** (bỏ sót mối đe dọa thực sự) là một yêu cầu quan trọng để tránh gây phiền phức cho người dùng hoặc làm suy yếu khả năng bảo vệ của hệ thống.

II. PHẠM VI THỰC HIỆN

- ❖ **Phát hiện Mối đe dọa Chính xác hơn**
 - **Giám sát và Phát hiện Mối đe dọa trong Thời gian thực:** Tích hợp Wazuh để giám sát và theo dõi các hoạt động trên các điểm cuối trong thời gian thực, phát hiện nhanh chóng các hành vi bất thường có thể đe dọa hệ thống.
 - **Phát hiện Mã độc đã Biết và chưa Biết:** Sử dụng ClamAV để kiểm tra các tệp nghi ngờ, nhận diện mã độc đã được xác định qua cơ sở dữ liệu, cũng như các mã độc chưa biết nhờ phương pháp phân tích heuristic và cập nhật cơ sở dữ liệu thường xuyên.
 - **Phát hiện Tấn công không Dùng Tệp (Fileless Attacks):** Tận dụng phân tích hành vi từ Wazuh để phát hiện các quy trình hoặc hành vi bất thường không liên quan đến tệp cụ thể, giúp ngăn chặn các cuộc tấn công ngay cả khi không có tệp độc hại trực tiếp.

❖ Quản lý Bảo mật Thiết bị Ngoại vi

- **Kiểm soát USB với USBGuard:** Triển khai USBGuard để kiểm soát việc kết nối USB, ngừng kết nối thiết bị không xác định và chỉ cho phép các thiết bị đã được xác thực, giảm thiểu nguy cơ lây lan mã độc qua thiết bị ngoại vi.
- **Cô lập và Hạn chế Rủi ro từ USB:** Cấu hình hệ thống để tự động quét các thiết bị USB khi kết nối và cô lập các thiết bị USB nghi ngờ ngay lập tức, ngăn chặn sự lây lan mã độc từ điểm kết nối.

❖ Cải thiện Quản lý Sự kiện Bảo mật

- **Tổng hợp Dữ liệu Bảo mật:** Tích hợp dữ liệu từ Wazuh và ClamAV, tạo bảng điều khiển tập trung để giám sát và quản lý các sự kiện bảo mật, giúp đội ngũ bảo mật dễ dàng theo dõi, phân tích và phản ứng kịp thời.
- **Phân tích Nguyên nhân Gốc rẽ:** Xây dựng quy trình phân tích nguyên nhân gốc rẽ để xác định nguyên nhân sâu xa của sự cố bảo mật và giải quyết nhanh chóng, ngăn ngừa sự cố tái diễn, bảo đảm sự ổn định hệ thống.

❖ Triển khai Giải pháp Bảo mật Hiệu quả và Tiết kiệm

- **Giải pháp Bảo mật Nguồn Mở:** Sử dụng các công cụ bảo mật mã nguồn mở như Wazuh và ClamAV để giảm chi phí bản quyền mà vẫn duy trì chất lượng bảo vệ hiệu quả.
- **Khả năng Tùy chỉnh Cao:** Xây dựng một giải pháp bảo mật có thể tùy chỉnh, phù hợp với nhu cầu của doanh nghiệp vừa và nhỏ hoặc các môi trường đặc thù, giúp mở rộng và tối ưu hóa bảo mật theo yêu cầu.

❖ Nâng cao Khả năng Ngăn chặn Mối đe dọa Mới Nổi

- **Dự đoán và Ngăn chặn Mối đe dọa Mới:** Tích hợp dữ liệu từ Wazuh và ClamAV để phát hiện các mẫu hành vi mới, giúp hệ thống có khả năng dự đoán và ngăn chặn các cuộc tấn công chưa từng được ghi nhận.
- **Chiến lược Bảo mật Chủ động:** Phát triển chiến lược bảo mật chủ động sử dụng phân tích dự đoán và thông tin tình báo mối đe dọa, giúp phát hiện và ngăn chặn mối đe dọa tiềm ẩn trước khi chúng có thể gây hại.

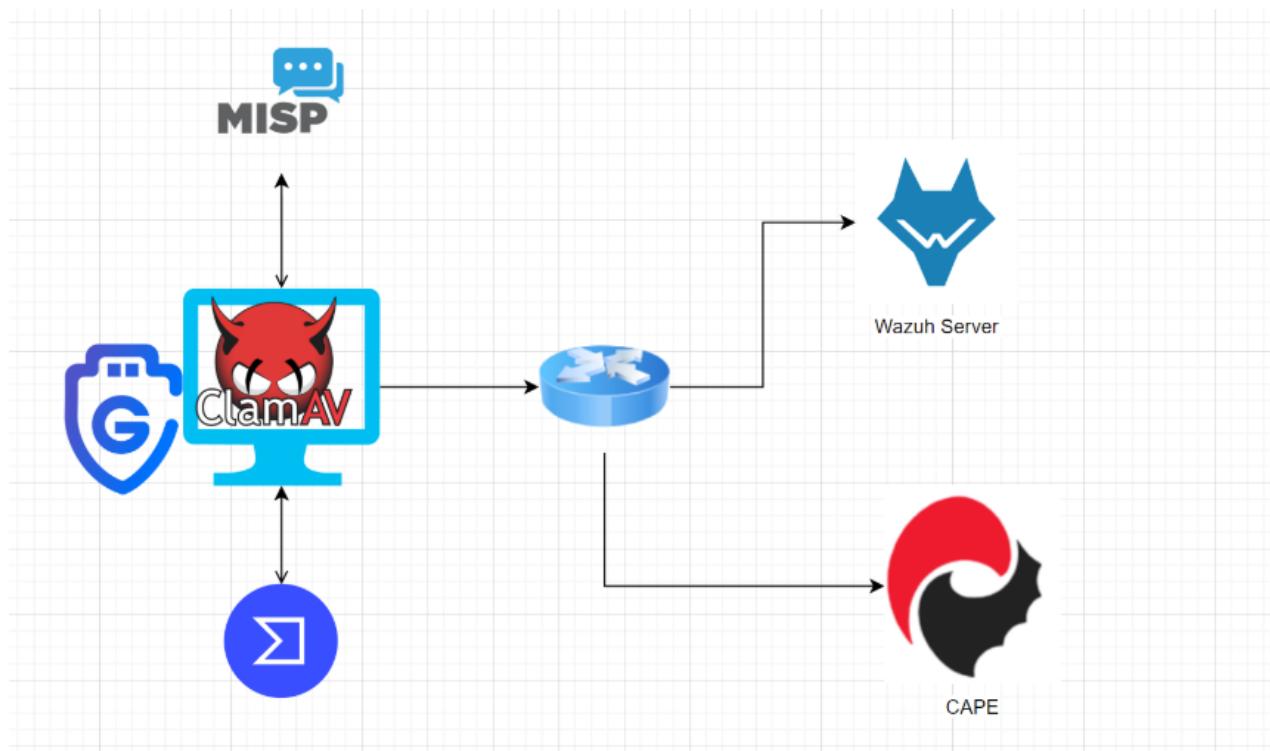
❖ Tăng cường Phân tích và Cô lập Mã độc qua Sandbox (CapeV2)

- **Phân tích Mã độc trong Sandbox:** Triển khai CapeV2 sandbox để phân tích hành vi của mã độc trong môi trường an toàn. Các tệp nghi ngờ sẽ được đưa vào sandbox để kiểm tra chi tiết và phát hiện các hoạt động độc hại.

- **Xây dựng Cơ sở Dữ liệu Mẫu Mã độc:** Mã độc phát hiện và phân tích trong sandbox sẽ giúp làm giàu cơ sở dữ liệu của ClamAV, cải thiện khả năng nhận diện các mối đe dọa mới và chưa được biết đến

III. KIẾN TRÚC HỆ THỐNG ĐỀ XUẤT

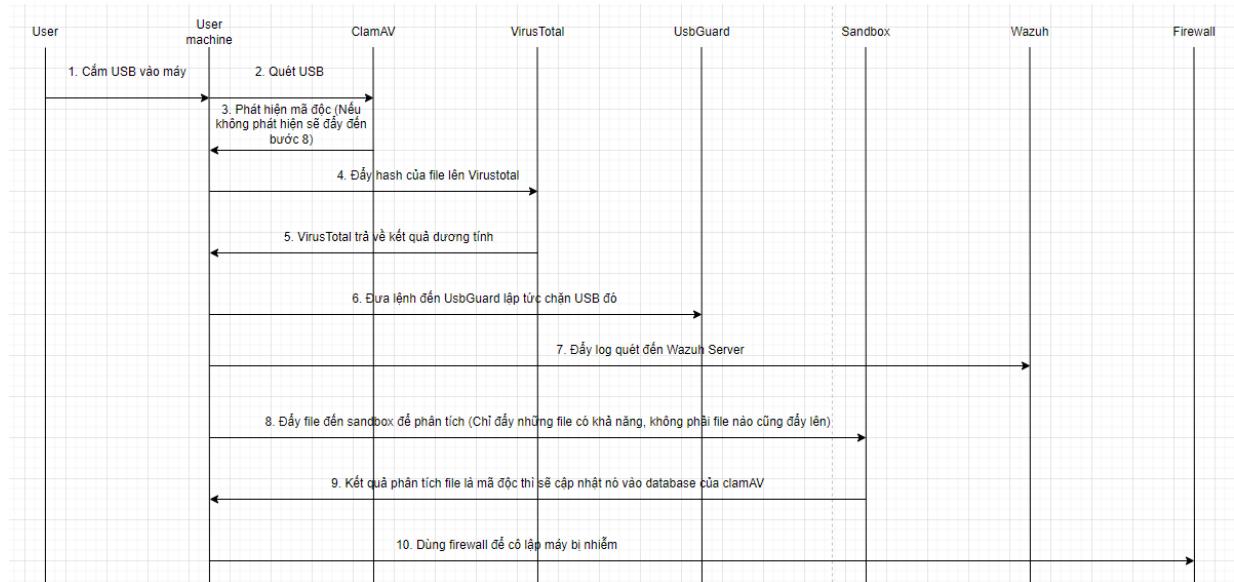
Mô hình triển khai:



Bao gồm một máy client ubuntu server đã cài đặt ClamAV, USBGuard, MISP, Wazuh agent.

Một máy Wazuh server và một máy cài Sandbox Cape.

Sequence flow:



IV. PHƯƠNG PHÁP TRIỂN KHAI

1. Chuẩn bị môi trường triển khai

a. Cài đặt các thành phần cơ bản:

- o Thiết lập môi trường hệ điều hành.
- o Cài đặt các công cụ cần thiết: ClamAV, UsbGuard, Wazuh, CAPE Sandbox.

b. Cấu hình ClamAV:

- o Cài đặt ClamAV từ kho phần mềm.
- o Cập nhật cơ sở dữ liệu mã độc bằng lệnh freshclam.
 - o Thiết lập ClamAV để tự động quét USB khi có thiết bị được kết nối qua udev rules.

c. Cài đặt và cấu hình UsbGuard:

- o Cài đặt UsbGuard bằng lệnh package manager.
- o Tạo danh sách các thiết bị USB được phép hoạt động.
 - o Tích hợp chính sách với ClamAV để chặn thiết bị USB chứa mã độc.

d. Cài đặt và cấu hình Wazuh:

- o Triển khai máy chủ Wazuh để thu thập log từ các thành phần khác.

- Cài đặt các agent trên máy khách để giám sát hoạt động USB, mạng, và file hệ thống.
- e. Triển khai CAPE Sandbox:
- Cài đặt và cấu hình CAPE Sandbox trên máy chủ riêng biệt.
 - Tích hợp với hệ thống chính để gửi tệp không xác định từ ClamAV để phân tích sâu.
- f. Tích hợp VirusTotal và URLhaus:
- Thiết lập kết nối API với VirusTotal để xác minh kết quả quét từ ClamAV.
 - Sử dụng URLhaus để phát hiện các hoạt động tải xuống bất thường.

2. Quy trình triển khai và hoạt động

a. Quy trình xử lý thiết bị USB:

- Khi USB được kết nối, ClamAV tự động quét toàn bộ nội dung trên USB. Kết quả quét được gửi tới VirusTotal qua API.
- Nếu phát hiện mã độc, UsbGuard chặn USB và ghi log sự kiện.
- Nếu mã độc là worm hoặc trojan, iptables được kích hoạt để chặn kết nối mạng của hệ thống.
- Nếu không phát hiện mã độc, file nghi vấn được chuyển tới CAPE Sandbox để phân tích sâu.
- Nếu sandbox phát hiện mã độc, hash của mã độc được thêm vào cơ sở dữ liệu ClamAV.

b. Quy trình giám sát hoạt động tải xuống:

- Khi người dùng tải tệp, URL của tệp được kiểm tra trên cơ sở dữ liệu URLhaus.
- Nếu URL nằm trong danh sách đen, Wazuh tạo cảnh báo và ghi nhận sự kiện.

c. Quản lý log và cảnh báo:

- Tất cả log từ ClamAV, UsbGuard, CAPE Sandbox, và URLhaus được gửi tới Wazuh.

- Wazuh phân tích log để tạo các cảnh báo theo thời gian thực.

d. Bảo trì và nâng cấp hệ thống:

- Định kỳ cập nhật cơ sở dữ liệu ClamAV để đảm bảo khả năng phát hiện mã độc mới nhất.
- Kiểm tra và tối ưu cấu hình của UsbGuard, iptables, và Wazuh.
- Theo dõi hiệu suất của CAPE Sandbox và cập nhật phần mềm để cải thiện khả năng phân tích.
- Ngoài ra còn áp dụng thêm machine learning để gia tăng khả năng phát hiện ra malware lừa

V. CÔNG CỤ SỬ DỤNG

Wazuh

Wazuh là một nền tảng mã nguồn mở dành cho giám sát an ninh, quản lý tuân thủ và phát hiện mối đe dọa. Nó kết hợp các khả năng của một hệ thống phát hiện xâm nhập (IDS) và quản lý thông tin và sự kiện an ninh (SIEM).

Tính năng chính:

- Thu thập, phân tích và lưu trữ log từ nhiều nguồn khác nhau.
- Giám sát tính toàn vẹn của tệp (FIM).
- Phát hiện mối đe dọa theo thời gian thực dựa trên các quy tắc được định nghĩa trước.
- Quản lý tuân thủ các tiêu chuẩn như GDPR, PCI DSS, ISO 27001.
- Tích hợp với Elastic Stack (Elasticsearch, Logstash, Kibana) để phân tích và hiển thị dữ liệu.

Ứng dụng thực tế: Giám sát hệ thống, phát hiện các cuộc tấn công mạng, và quản lý bảo mật trong các môi trường doanh nghiệp.

ClamAV

ClamAV (Clam AntiVirus) là một công cụ diệt virus mã nguồn mở, thường được sử dụng để quét và phát hiện mã độc trong hệ thống.

Tính năng chính:

- Phát hiện và loại bỏ virus, trojan, worm, và phần mềm độc hại khác.
- Cung cấp cơ sở dữ liệu mã độc được cập nhật thường xuyên.

- Hỗ trợ quét các file nén (ZIP, RAR) và các định dạng email phổ biến.
- Tích hợp với hệ thống để quét theo yêu cầu hoặc theo lịch trình.

Ứng dụng thực tế: Triển khai trên máy chủ để quét tệp tải xuống, email hoặc kiểm tra nội dung USB.

CapeSandbox

CAPE Sandbox (Capture Analysis Platform for Execution) là một công cụ phân tích mã độc mã nguồn mở. Nó cho phép phân tích các mẫu phần mềm độc hại trong môi trường cách ly an toàn (sandbox).

Tính năng chính:

- Phân tích hành vi của tệp đáng ngờ (bao gồm thực thi mã độc).
- Cung cấp thông tin chi tiết về quá trình, mạng, và các hành động hệ thống của tệp.
- Hỗ trợ phân tích các định dạng tệp như PE, Office, PDF, và tệp nén.
- Giao diện API để tích hợp với các hệ thống khác.

Ứng dụng thực tế: Phân tích phần mềm độc hại chưa được phát hiện bởi các công cụ truyền thống và cung cấp thông tin chi tiết để nâng cấp cơ sở dữ liệu mã độc.

USBGuard

USBGuard là một công cụ mã nguồn mở giúp kiểm soát quyền truy cập của các thiết bị USB vào hệ thống.

Tính năng chính:

- Áp dụng chính sách kiểm soát thiết bị USB (cho phép, từ chối, hoặc hạn chế).
- Giám sát và ghi lại hoạt động của thiết bị USB.
- Tăng cường bảo mật bằng cách ngăn chặn các cuộc tấn công qua USB như BadUSB.
- Dễ dàng tích hợp với các hệ thống quản lý bảo mật khác.

Ứng dụng thực tế: Bảo vệ hệ thống khỏi các thiết bị USB không đáng tin cậy hoặc chứa mã độc.

MISP

MISP (Malware Information Sharing Platform) là một nền tảng mã nguồn mở dành cho chia sẻ thông tin về mối đe dọa (Threat Intelligence Sharing).

Tính năng chính:

- Thu thập và chia sẻ dữ liệu về các chỉ số thỏa hiệp (IoC) và mối đe dọa.
- Cung cấp giao diện để phân tích và xử lý dữ liệu tình báo.
- Hỗ trợ các tiêu chuẩn như STIX và TAXII để tích hợp với các nền tảng khác.
- Hợp tác trong cộng đồng để chia sẻ thông tin về các cuộc tấn công và mã độc.

Ứng dụng thực tế: Nâng cao khả năng phát hiện và phản ứng với mối đe dọa bằng cách chia sẻ thông tin với các tổ chức khác.

VirusTotal

VirusTotal là một dịch vụ trực tuyến giúp phân tích tệp và URL đáng ngờ bằng cách sử dụng nhiều công cụ diệt virus và phân tích mã độc.

Tính năng chính:

- Phân tích tệp và URL bằng hơn 70 công cụ diệt virus và cơ sở dữ liệu mã độc.
- Hỗ trợ API để tích hợp với các hệ thống bảo mật khác.
- Cung cấp báo cáo chi tiết về các mối đe dọa tiềm ẩn.
- Lưu trữ thông tin về các mẫu mã độc để sử dụng trong tương lai.

Ứng dụng thực tế: Kiểm tra và xác minh kết quả quét từ các công cụ diệt virus khác.

URLhaus

URLhaus là một dự án cộng đồng nhằm phát hiện và ngăn chặn các URL độc hại, chủ yếu tập trung vào việc ngăn chặn các tệp thực thi độc hại.

Tính năng chính:

- Cung cấp danh sách URL độc hại được cập nhật liên tục.
- Tích hợp với các công cụ bảo mật để chặn các URL nguy hiểm.
- Báo cáo và theo dõi các nguồn lây nhiễm mã độc.
- Tương thích với các hệ thống SIEM và firewall.

Ứng dụng thực tế: Bảo vệ hệ thống khỏi các cuộc tấn công qua URL và ngăn chặn tải xuống các tệp độc hại từ các nguồn không đáng tin cậy.

VI. DEMO

Dưới đây là code em viết để chạy ClamAV với VirusTotal:

```
#!/bin/bash

# API key của VirusTotal
API_KEY="96f66d2a834b377352b64fbb882c545c704b6ec19487a17f7129994b6d7cb07c"

# Thư mục cách ly file
QUARANTINE_DIR="/var/quarantine"
mkdir -p $QUARANTINE_DIR

# Tạo danh sách các file đã xử lý
PROCESSED_FILES="/tmp/processed_files.txt"
> $PROCESSED_FILES

# Tạo file log mới với timestamp
LOG_FILE="/var/log/clamav/scan_$(date +'%Y%m%d_%H%M%S').log"

# Chạy ClamAV để quét hệ thống và ghi vào file log mới
echo "Starting ClamAV scan..."
sudo /usr/bin/clamdscan --fdpass /home --infected --log=$LOG_FILE
# /usr/bin/clamdscan -r /home --infected --log=$LOG_FILE

# Đặt quyền sở hữu file log cho người dùng và nhóm clamav
sudo chown clamav:clamav $LOG_FILE

# Đọc log ClamAV mới và xử lý file bị nhiễm
cat $LOG_FILE | grep "FOUND" | while read LINE; do
    # Lấy đường dẫn file bị nhiễm từ log và loại bỏ dấu :
    FILE=$(echo "$LINE" | awk '{print $1}' | sed 's/://g')

    # Kiểm tra xem file đã được xử lý chưa
    if grep -q "$FILE" "$PROCESSED_FILES"; then
        echo "File $FILE has already been processed. Skipping."
        continue
    fi

    # Kiểm tra đường dẫn file
    if [[ -z "$FILE" ]]; then
        echo "Error: Could not extract file path from log. Log entry: $LINE"
        continue
    fi

    # Kiểm tra file có tồn tại không
    if [ ! -f "$FILE" ]; then
        echo "Sending $FILE to VirusTotal for analysis"
        FILE_HASH=$(sha256sum "$FILE" | awk '{print $1}')

        # Gửi hash đến VirusTotal
        RESPONSE=$(curl -s --request GET --url "https://www.virustotal.com/vtapi/v2/file/report?apikey=$API_KEY&resource=$FILE_HASH")

        # In phản hồi từ VirusTotal để kiểm tra
        echo "VirusTotal response: $RESPONSE"

        POSITIVES=$(echo $RESPONSE | jq '.positives')
        TOTAL=$(echo $RESPONSE | jq '.total')

        # Nếu tỉ lệ phát hiện trên VirusTotal > 10 engines thì xóa file
        if [[ $POSITIVES -gt 10 ]]; then
            # Nếu tỉ lệ phát hiện trên VirusTotal > 10 engines thì xóa file
            if [[ $POSITIVES -gt 10 ]]; then
                echo "Highly suspicious file: $FILE (Detected by $POSITIVES/$TOTAL engines). Deleting file."
                rm -f "$FILE" && echo "$FILE deleted successfully."
            # Nếu tỉ lệ phát hiện nhỏ hơn hoặc bằng 10 thì cách ly file
            else
                echo "Quarantining moderately suspicious file: $FILE"
                mv "$FILE" "$QUARANTINE_DIR" && echo "$FILE moved to quarantine."
            fi
        # Dánh dấu file đã được xử lý
        echo "$FILE" >> $PROCESSED_FILES
    else
        echo "File $FILE does not exist or cannot be found."
    fi
done
```

```
# Nếu tỉ lệ phát hiện trên VirusTotal > 10 engines thì xóa file
if [[ $POSITIVES -gt 10 ]]; then
    echo "Highly suspicious file: $FILE (Detected by $POSITIVES/$TOTAL engines). Deleting file."
    rm -f "$FILE" && echo "$FILE deleted successfully."
# Nếu tỉ lệ phát hiện nhỏ hơn hoặc bằng 10 thì cách ly file
else
    echo "Quarantining moderately suspicious file: $FILE"
    mv "$FILE" "$QUARANTINE_DIR" && echo "$FILE moved to quarantine."
fi

# Dánh dấu file đã được xử lý
echo "$FILE" >> $PROCESSED_FILES
else
    echo "File $FILE does not exist or cannot be found."
fi
done
```

Đoạn code trên sẽ hoạt động như sau:

1. Quét tệp bằng ClamAV và ghi log.
 2. Đọc log để tìm các tệp bị nhiễm.
 3. Kiểm tra xem tệp đã được xử lý chưa.
 4. Gửi hash của tệp tới VirusTotal để xác minh.
 5. Nếu tệp bị nghi là mã độc:
 6. Xóa tệp hoặc cách ly dựa trên mức độ nghi ngờ.
 7. Ghi lại tên tệp đã xử lý vào danh sách để tránh xử lý lại.

Còn đây là đoạn config để đẩy log về Wazuh.

```
<!-- Configure Wazuh agent to collect and forward the ClamAV logs to the Wazuh server for analysis -->
<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/clamav/clamav.log</location>
</localfile>
```

Sau khi em chạy test thử code trên với signature Eicar, thì đây là response của nó:

Có thể thấy được là kết quả có đến 60/69 engines detect được nên là file đã bị xóa đi.

Đồng thời check bên Wazuh thì em cũng đã nhận được log:

Sep 11, 2024 @ 20:31:05.946	ClamAV: Virus detected	8	52502
Table JSON Rule			
@timestamp	2024-09-11T13:31:05.946Z		
_id	4B1H4ZEBjuKuOsBD3DtY		
agent.id	002		
agent.ip	192.168.118.100		
agent.name	snort		
data.extra_data	Eicar-Signature		
data.id	69630e4574ec6798239b091cda43dca0		
data.url	/home/nghianguyen/test/eicar_test_file.txt		
decoder.name	clamd		
decoder.parent	clamd		
full_log	2024-09-11T13:31:04.800510+00:00 snort clamd[4723]: /home/nghianguyen/test/eicar_test_file.txt: Eicar-Signature(69630e4574ec6798239b091cda43dca0:69) FOUND		
id	1726061465.497069		
input.type	log		

input.type	log
location	/var/log/syslog
manager.name	nghianguyen-virtual-machine
predecoder.program_name	clamd
predecoder.timestamp	2024-09-11T13:31:04.800510+00:00
rule.description	ClamAV: Virus detected
rule.firedtimes	2
rule.gdpr	IV_35.7.d
rule.gpg13	4.2
rule.groups	clamd, freshclam, virus
rule.id	52502
rule.level	8
rule.mail	false
rule.nist_800_53	SI.3, SI.4
rule pci_dss	5.1, 5.2, 11.4
rule.tsc	A1.2, CC6.1, CC6.8, CC7.2, CC7.3
timestamp	2024-09-11T20:31:05.946+0700

Tiếp theo là em sẽ tích hợp với lại USBGuard vào cho việc tự động scan và chặn usb.

Config của em bên USBGuard:

```

#           device
#
#PresentControllerPolicy=allow

#
# Inserted device policy.
#
# How to treat USB devices that are already connected
# *after* the daemon starts. One of:
#
# * block      - deauthorize every present device
# * reject     - remove every present device
# * apply-policy - evaluate the ruleset for every present
#                  device
#
#InsertedDevicePolicy=apply-policy

#
# Control which devices are authorized by default.
#
# The USBDaemon daemon modifies some the default authorization state attributes
# of controller devices. This setting, enables you to define what value the
# default authorization is set to.
#
# * keep       - do not change the authorization state
# * none       - every new device starts out deauthorized
# * all        - every new device starts out authorized
# * internal   - internal devices start out authorized, external devices start
#                 out deauthorized (this requires the ACPI tables to properly
#                 label internal devices, and kernel support)
#
#AuthorizedDefault=all

#
# Restore controller device state.
#
# The USBDaemon daemon modifies some attributes of controller

```

Em cũng tạo một cái udev để tự động chạy khi có usb kết nối vào:

```

GNU nano 7.2
/etc/udev/rules.d/91-usbguard-scan.rules
ACTION=="add", SUBSYSTEM=="block", KERNEL=="sd[a-z][0-9]", RUN+="/usr/bin/logger 'USB device added and rule triggered'"
ACTION=="add", SUBSYSTEM=="block", KERNEL=="sd[a-z][0-9]", RUN+="/bin/systemctl start usb_scan.service"

```

```

GNU nano 7.2
[Unit]
Description=USB Scan Service
After=usb_modeswitch@%i.service

[Service]
Type=oneshot
ExecStart=/usr/local/bin/usb_scan_and_allow.sh

```

Còn đây là đoạn code tích hợp thêm USBGuard vào:

```
#!/bin/bash

# API key của VirusTotal
API_KEY="96f66d2a834b377352b64fbb882c545c704b6ec19487a17f7129994b6d7cb07c"

# Tạo file log mới với timestamp
LOG_FILE="/var/log/clamav/usb_scan_$(date +'%Y%m%d_%H%M%S').log"

# Thư mục lưu trữ tạm thời các USB đã được quét trong phiên làm việc hiện tại
SCANNED_USB_IDS="/tmp/scanned_usb_ids.txt"

# Kiểm tra và tạo tệp nếu chưa tồn tại
if [ ! -f "$SCANNED_USB_IDS" ]; then
    /usr/bin/touch $SCANNED_USB_IDS
fi

# Ghi vào log bắt đầu thực hiện script
/usr/bin/logger "Starting usb_scan_and_allow.sh script"

# Liệt kê tất cả thiết bị USB đang kết nối
for DEVICE_ID in $(/usr/bin/usbguard list-devices | /usr/bin/grep -E "allow|block" | /usr/bin/awk '{print $1}' | /usr/bin/tr -d ':'); do
    /usr/bin/logger "Processing Device ID: $DEVICE_ID"

    # Kiểm tra nếu thiết bị đã được quét chưa
    if /usr/bin/grep -q "$DEVICE_ID" "$SCANNED_USB_IDS"; then
        /usr/bin/logger "Device ID $DEVICE_ID already scanned. Skipping."
        continue
    fi

    # Tìm đường dẫn thiết bị có hệ thống tập tin
    DEVICE_PATH=$(lsblk -lnp -o NAME,TRAN | /usr/bin/grep 'usb' | /usr/bin/awk '{print $1}')
    /usr/bin/logger "Device Path: $DEVICE_PATH"

    # Kiểm tra nếu tìm thấy đường dẫn thiết bị
    if [ -z "$DEVICE_PATH" ]; then
        /usr/bin/logger "Failed to find device path for DEVICE_ID $DEVICE_ID. Skipping scan."
        continue
    fi

    # Tìm phân vùng chứa hệ thống tập tin cần cần thiết
    PARTITION_PATH=$(lsblk -lnp -o NAME,FSTYPE | /usr/bin/grep "$DEVICE_PATH" | /usr/bin/grep -E 'vfat|ntfs|ext4|exfat|fat32' | /usr/bin/awk '{print $1}')
    /usr/bin/logger "Partition Path: $PARTITION_PATH"

    # Kiểm tra nếu tìm thấy phân vùng chứa hệ thống tập tin
    if [ -z "$PARTITION_PATH" ]; then
        /usr/bin/logger "Failed to find a valid partition on device $DEVICE_PATH. Skipping scan."
        continue
    fi

    # Tạo điểm gắn kết tạm thời
    MOUNT_PATH="/media/usb_$DEVICE_ID"
    /usr/bin/mkdir -p $MOUNT_PATH
    /usr/bin/logger "Creating mount point at $MOUNT_PATH"

    # Thêm khoảng chờ trước khi mount
    /usr/bin/sleep 2

    # Gắn kết thiết bị vào điểm gắn kết ở chế độ chỉ đọc
    /usr/bin/logger "Attempting to mount $PARTITION_PATH to $MOUNT_PATH"
    /usr/bin/mount -o ro $PARTITION_PATH $MOUNT_PATH

    # Kiểm tra xem thiết bị có gắn kết thành công không
    if /bin/mount | /usr/bin/grep -q "$MOUNT_PATH"; then
        /usr/bin/logger "Scanning USB device with ID $DEVICE_ID at $MOUNT_PATH"

        # Chạy ClamAV để quét USB
        /usr/bin/clamscan --fdpass --infected --log=$LOG_FILE $MOUNT_PATH

        # Đọc log để kiểm tra xem có file nào bị nhiễm không
        INFECTED_FILES=$(grep "FOUND" $LOG_FILE | /usr/bin/awk '{print $1}' | /usr/bin/sed 's/://g')

        if [ ! -z "$INFECTED_FILES" ]; then
            /usr/bin/logger "Infected files found on device $DEVICE_ID!"
            for FILE in $INFECTED_FILES; do
                /usr/bin/logger "Sending $FILE to VirusTotal for analysis..."
                FILE_HASH=$(shasum "$FILE" | /usr/bin/awk '{print $1}')

                # Gửi hash đến VirusTotal
                RESPONSE=$(curl -s -X POST "https://www.virustotal.com/vtapi/v2/file/report?apikey=$API_KEY&resource=$FILE_HASH")
                # In phản hồi từ VirusTotal để kiểm tra
                /usr/bin/logger "VirusTotal response: $RESPONSE"

                POSITIVES=$(echo $RESPONSE | jq '.positives')
                TOTAL=$(echo $RESPONSE | jq '.total')

                # Nếu tỉ lệ phát hiện trên VirusTotal > 10 engines thì chặn thiết bị
                if [[ $POSITIVES -gt 10 ]]; then
                    /usr/bin/logger "Highly suspicious file: $FILE (Detected by $POSITIVES/$TOTAL engines). Blocking device $DEVICE_ID."
                    /usr/bin/usbguard block-device $DEVICE_ID
                    break
                fi
            done
        else
            /usr/bin/logger "No dangerous files found on device $DEVICE_ID."
        fi

        # Thêm thiết bị vào danh sách đã quét trong phiên này
        echo "$DEVICE_ID" >> "$SCANNED_USB_IDS"

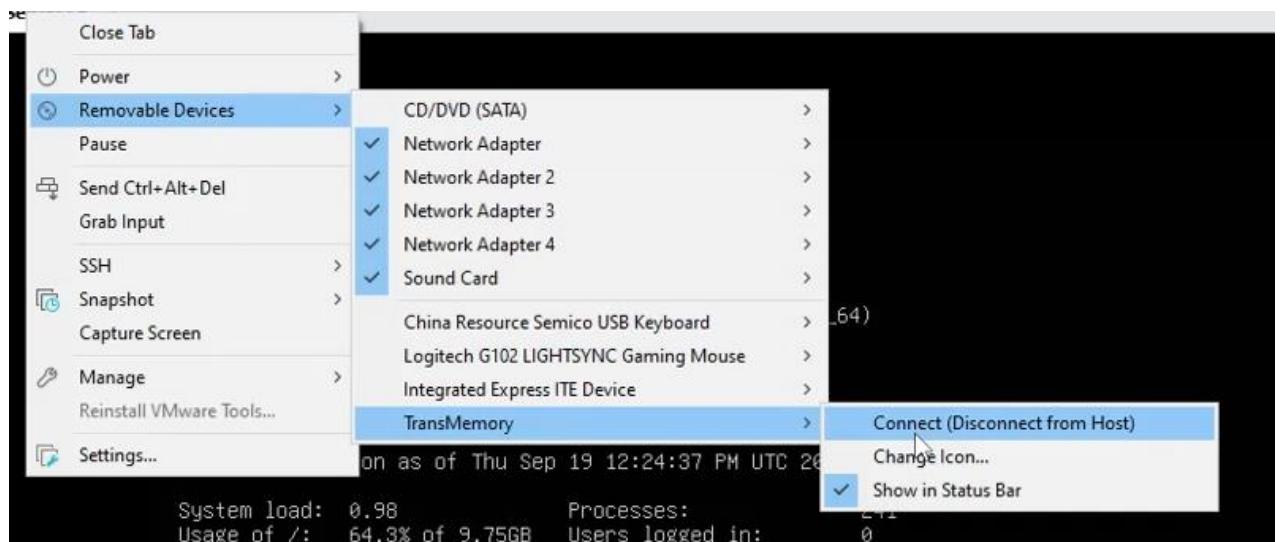
        # Gỡ bỏ gắn kết thiết bị sau khi quét xong
        /usr/bin/umount $MOUNT_PATH
        /usr/bin/rmdir $MOUNT_PATH
    else
        /usr/bin/logger "Mount failed for $PARTITION_PATH on $MOUNT_PATH"
        /usr/bin/logger "Failed to mount device with ID $DEVICE_ID. Skipping scan."
    fi
done
```

Cơ bản thì nhiệm vụ của đoạn code này cũng giống với code nãy, nhưng em có thêm là mount usb vào rồi quét, sẽ đưa usb vào cho USBGuard chặn nếu phát hiện mã độc có trong usb.

Ban đầu em sẽ đưa usb sạch vào để test:

Kết quả là nó vẫn được allow, trong hệ thống ổ đĩa nó cũng xuất hiện luôn.

Sau đây thì em tiến hành đưa usb có chứa signature của Eicar vào để test:



Khi connect vào thì script sẽ tự động scan usb:

Thì lúc này usb đã bị chẩn, đồng thời trong danh sách ổ đĩa cũng không thấy.

Cứ mỗi lần cắm vào là script sẽ scan, mặc dù cho là usb này đã bị chặn nhưng nếu gỡ usb đó ra và xóa virus, rồi cắm lại vào máy thì nó vẫn sẽ được allow bình thường, tránh tình trạng là usb bị block vĩnh viễn.

Ngoài ra thì đoạn script của em cũng có thêm firewall trong trường hợp detect được virus là worm hay trojan, nó sẽ chặn hết các cổng mạng, traffic không ra hay vào được, nhằm cô lập máy tính:

```
#!/bin/bash

TROJAN_KEYWORDS=("Trojan" "Worm")
```

```
# Kiểm tra loại mã độc để áp dụng firewall
for MALWARE in "${TROJAN_KEYWORDS[@]}"; do
    if echo "$SCAN_RESULT" | /usr/bin/grep -qi "$MALWARE"; then
        /usr/bin/logger "Detected $MALWARE in $FILE. Isolating the machine using firewall rules."

        # Lấy danh sách tất cả subnet của máy
        SUBNETS=$(ip -o -f inet addr show | awk '/scope global/ {print $4}')
        for SUBNET in $SUBNETS; do
            /usr/bin/logger "Applying firewall rules to isolate machine from subnet $SUBNET."
            # Firewall rules không dùng /usr/bin/
            iptables -I OUTPUT -d $SUBNET -j DROP
            iptables -I INPUT -s $SUBNET -j DROP
        done
        /usr/bin/logger "Machine isolated from all connected subnets due to detection of $MALWARE."
        break
    fi
done
```

Trước tiên thì em test thử ping ra máy khác và ping ra ngoài mạng, thì tất cả đều đã thành công:

```
nghianguyen@snort:~$ ping 10.11.12.10
PING 10.11.12.10 (10.11.12.10) 56(84) bytes of data.
64 bytes from 10.11.12.10: icmp_seq=1 ttl=64 time=0.195 ms
64 bytes from 10.11.12.10: icmp_seq=2 ttl=64 time=0.168 ms
64 bytes from 10.11.12.10: icmp_seq=3 ttl=64 time=0.177 ms
64 bytes from 10.11.12.10: icmp_seq=4 ttl=64 time=0.156 ms
^C
--- 10.11.12.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3051ms
rtt min/avg/max/mdev = 0.156/0.174/0.195/0.014 ms
nghianguyen@snort:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=128 time=32.2 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=128 time=32.3 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=128 time=32.1 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=128 time=33.2 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 32.149/32.466/33.238/0.451 ms
```

Nhưng sau khi áp dụng script trên vào để scan usb có chứa worm:

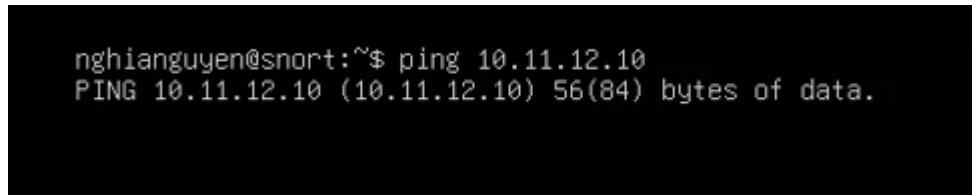
```

nghianguyen@snort:~$ ping 10.11.12.10
PING 10.11.12.10 (10.11.12.10) 56(84) bytes of data.
64 bytes from 10.11.12.10: icmp_seq=1 ttl=64 time=0.195 ms
64 bytes from 10.11.12.10: icmp_seq=2 ttl=64 time=0.168 ms
64 bytes from 10.11.12.10: icmp_seq=3 ttl=64 time=0.177 ms
64 bytes from 10.11.12.10: icmp_seq=4 ttl=64 time=0.156 ms
^C
--- 10.11.12.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3051ms
rtt min/avg/max/mdev = 0.156/0.174/0.195/0.014 ms
nghianguyen@snort:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=128 time=32.2 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=128 time=32.3 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=128 time=32.1 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=128 time=33.2 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 32.149/32.466/33.238/0.451 ms
nghianguyen@snort:~$ sudo nano /usr/local/bin/usb_scan_and_allow.sh
[sudo] password for nghianguyen:
nghianguyen@snort:~$ nghianguyen@snort:~$ client_loop: send disconnect: Connection reset
C:\Users\ngolia>

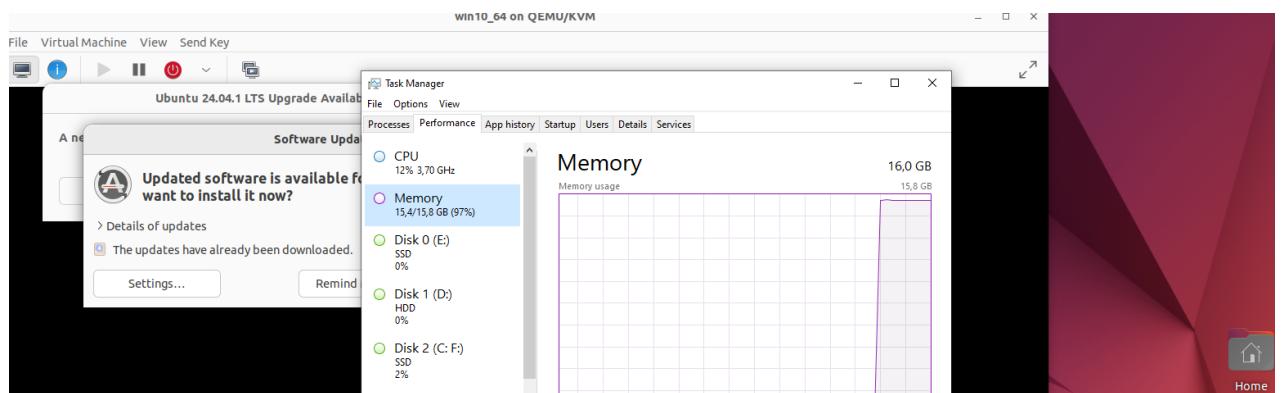
```

Ngay lập tức màn hình cmd máy chính của em đã mất kết nối ssh đến máy ảo của em.

Đồng thời sử dụng máy ảo để ping cũng không được:

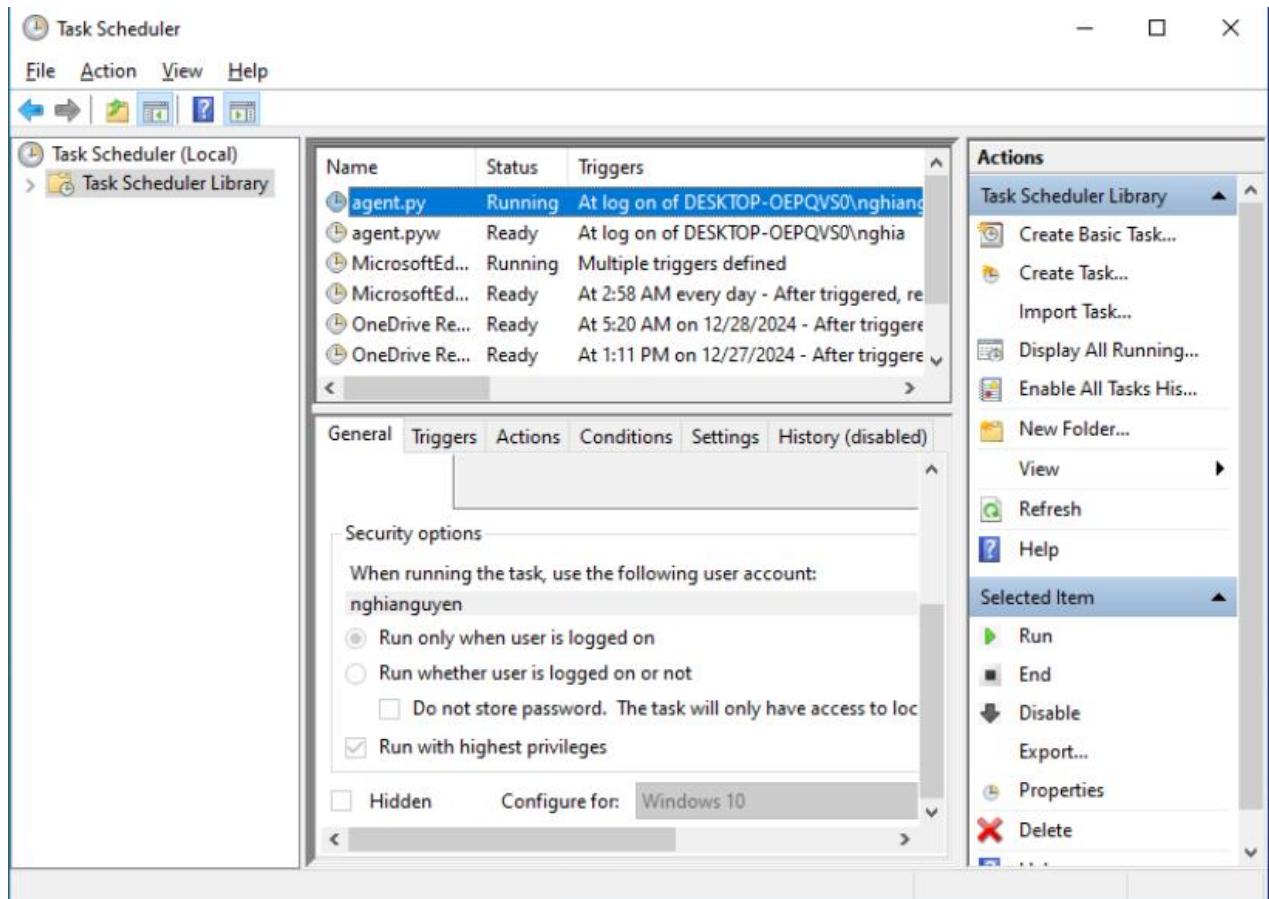


Tiếp theo là đến sandbox Capev2, do là việc Capev2 ngôn ngữ quá nhiều tài nguyên và máy thì hạn chế về ram nên là em chỉ có thể thiết lập rồi chạy sandbox độc lập, chưa thể chạy chung nó với các máy ảo khác để integrate các tính năng lại.



Để thiết lập được sandbox thì trước tiên là em sẽ phải cài kvm và rất nhiều dependencies cho nó trên một máy ảo Ubuntu, sau đó sử dụng đĩa iso Window để cài đặt hệ điều hành cho máy ảo kvm này.

Đồng thời trong máy ảo Window này em sẽ thiết lập một số thứ như là tắt đi tự động update, firewall, window defender, đặc biệt là sẽ cài Cape agent vào và thiết lập cho nó chạy mỗi khi khởi động máy hay đăng nhập vào:



Cài đặt xong máy ảo kvm thì em sẽ snapshot lại rồi tiếp đến sẽ cài đặt Capev2 cùng với dependencies và config cho nó, một số config như sau:

Cuckoo.conf:

```

[resultserver]
# The Result Server is used to receive in real time the beh
# produced by the analyzer.
# Specify the IP address of the host. The analysis machines
# to contact the host through such address, so make sure it
# NOTE: if you set resultserver IP to 0.0.0.0 you have to s
# `resultserver_ip` for all your virtual machines in machin
ip = 192.168.122.1

# Specify a port number to bind the result server on.
port = 2042

# Force the port chosen above, don't try another one (we ca
# port dynamically if we can not bind this one, but that is
# in some setups)
force_port = yes

pool_size = 0

# Should the server write the legacy CSV format?
# (if you have any custom processing on those, switch this
store_csvs = off

# Maximum size of uploaded files from VM (screenshots, drop
# The value is expressed in bytes, by default 100MB.
upload_max_size = 1000000000

# To enable trimming of huge binaries go to -> web.conf ->
# Prevent upload of files that passes upload_max_size?
do_upload_max_size = no

```

```

[database]
# Specify the database connection string.
# Examples, see documentation for more:
# sqlite:///foo.db
# postgresql://foo:bar@localhost:5432/mydatabase
# mysql://foo:bar@localhost/mydatabase
# If empty, default is a SQLite in db/cuckoo.db.
# SQLite doesn't support database upgrades!
# For production we strongly suggest go with PostgreSQL
connection = postgresql://cape:SuperPuperSecret@localhost:5432/cape
# If you use PostgreSQL: SSL mode
# https://www.postgresql.org/docs/current/libpq-ssl.html#LIBPQ-SSL-SSLMODE-STATEMENTS
pgsql_ssl_mode = disable

# Database connection timeout in seconds.
# If empty, default is set to 60 seconds.
timeout =

# Log all SQL statements issued to the database.
log_statements = off

```

```

# Specify the name of the machinery module to use, this module will
# define the interaction between Cuckoo and your virtualization software
# of choice.
machinery = kvm

# Enable screenshots of analysis machines while running.
machinery_screenshots = on

```

Kvm.conf:

```

[kvm]
# Specify a comma-separated list of available machines to be used. For each
# specified ID you have to define a dedicated section containing the details
# on the respective machine. (E.g. cuckoo1,cuckoo2,cuckoo3)
machines = win10_64

interface = virbr0
# To connect to local or remote host
dsn = qemu:///system

# To allow copy & paste. For details see example below
#[win10_64]
#label = win10_64
#platform = windows
#ip = 192.168.122.100
#arch = x86
#tags = win10
#snapshot = Snapshot1
#resultserver_ip = 192.168.122.101
#reserved = no

[win10_64]
# Specify the label name of the current machine as specified in your
# libvirt configuration.
label = win10_64

# Specify the operating system platform used by current machine
# [windows/darwin/linux].
platform = windows

# Specify the IP address of the current virtual machine. Make sure that the
# IP address is valid and that the host machine is able to reach it. If not,
# the analysis will fail. You may want to configure your network settings in
# /etc/libvirt/<hypervisor>/networks/
ip = 192.168.122.100

```

```

# (Optional) Specify the snapshot
# name, the KVM MachineManager
# Example (Snapshot1 is the source
snapshot = snapshot2

```

```
# Set the machine architecture
# Required to auto select proper machine
# x64 or x86
arch = x64
```

Processing.conf:

```
[detections]
enabled = yes
# Signatures
behavior = yes
yara = yes
suricata = yes
virustotal = yes
clamav = no

[virustotal]
enabled = yes
on_demand = no
timeout = 60
# remove empty detections
remove_empty = yes
# Add your VirusTotal API key here. The default API key, kindly provided
# by the VirusTotal team, should enable you with a sufficient throughput
# and while being shared with all our users, it shouldn't affect your use.
key = 96f66d2a834b377352b64fbb882c545c704b6ec19487a17f7129994b6d7cb07c
do_file_lookup = yes
do_url_lookup = yes
urlscrub = (^http://serw.clicksor.com/redir.php?url=|&InjectedParam=.+$)
```

```

[suricata]
# Notes on getting this to work check install_suricata function:
# https://github.com/kevoreilly/CAPEv2/blob/master/installer/cape2.sh

enabled = yes
#Runmode "cli" or "socket"
runmode = socket
#Outputfiles
# if evelog is specified, it will be used instead of the per-protocol log files
evelog = eve.json

# per-protocol log files
#
#alertlog = alert.json
#httplog = http.json
#tlslog = tls.json
#sshlog = ssh.json
#dnslog = dns.json

fileslog = files-json.log
filesdir = files
# Amount of text to carve from plaintext files (bytes)
buffer = 8192
#Used for creating an archive of extracted files
7zbin = /usr/bin/7z
zippass = infected
##Runmode "cli" options
bin = /usr/bin/suricata
conf = /etc/suricata/suricata.yaml
##Runmode "socket" Options
socket_file = /tmp/suricata-command.socket

```

Reporting.conf:

```

[mongodb]
enabled = yes
host = 127.0.0.1
port = 27017
db = cuckoo
# Set those values if you are using mongodb authentication
# username =
# password =
# authsource = cuckoo

# Set this value if you are using mongodb with TLS enabled
# tlscacert =

# Automatically delete large dict values that exceed mongos 16MB limitation
# Note: This only deletes dict keys from data stored in MongoDB. You would
# still get the full dataset if you parsed the results dict in another
# reporting module or from the jsondump module.
fix_large_docs = yes

```

```

# Community
[misp]
enabled = yes
apikey =
url =
#Make event published after creation?
published = no
# minimal malscore, by default all
min_malscore = 0
# by default 5 threads
threads =
# this will retrieve information for iocs
# and activate misp report download from webgui
extend_context = no
# upload iocs from cuckoo to MISP
upload_iocs = no
distribution = 0
threat_level_id = 2
analysis = 2
# Sections to report
# Analysis ID will be appended, change
title = Iocs from cuckoo analysis:
network = no
ids_files = no
dropped = no
registry = no
mutexes = no

```

Routing.conf:

```

route = internet

# Network interface that allows a VM to connect to the entire internet, the
# "dirty line" so to say. Note that, just like with the VPNs, this will allow
# malicious traffic through your network. So think twice before enabling it.
# (For example, to route all VMs through eth0 by default: "internet = eth0").
internet = virbr0

# When set to no masquerade rule has been not generated.
nat = yes

# When property nat set to yes. That property not used.
# When property nat set to no and no_local_routing to yes, a vrf configuration
# will be generated and local traffic will go through by internet interface (dirty_line).
no_local_routing = yes

# Routing table name/id for "dirty line" interface. If "dirty line" is
# also default gateway in the system you can leave "main" value. Otherwise add
# new routing table by adding "<id> <name>" line to /etc/iproute2/rt_tables
# (e.g., "200 eth0"). ID and name must be unique across the system (refer to
# /etc/iproute2/rt_tables for existing names and IDs).
rt_table = main

```

Sau khi thiết lập xong thì em sẽ khởi động chạy:

```

nghianguyen@nghianguyen-virtual-machine:~/Sandbox/CAPEv2$ sudo -u cape /etc/poetry/bin/poetry run python3 cuckoo.py
[sudo] password for nghianguyen:

          ),-.-, /
Cuckoo Sandbox      <(a`----','
no chance for malwares!   ('-, ._>')
                           ) _>._/_/
                           _/_/
```

Cuckoo Sandbox 2.4-CAPE
www.cuckoosandbox.org
Copyright (c) 2010-2015

CAPE: Config and Payload Extraction
github.com/kevoreilly/CAPEv2

XLMMacroDeobfuscator: pywin32 is not installed (only is required if you want to use MS Excel)
/usr/bin/tcpdump

```

2024-12-30 10:16:58,923 [lib.cuckoo.core.machinery_manager] INFO: Using MachineryManager[kvm] with max_machines_count=10
2024-12-30 10:16:58,924 [lib.cuckoo.core.scheduler] INFO: Creating scheduler with max_analysis_count=unlimited
2024-12-30 10:16:58,946 [lib.cuckoo.core.machinery_manager] INFO: Loaded 1 machine
2024-12-30 10:16:58,975 [lib.cuckoo.core.machinery_manager] INFO: max_vmstartup_count for BoundedSemaphore = 5
2024-12-30 10:16:58,976 [lib.cuckoo.core.scheduler] INFO: Waiting for analysis tasks
```

Màn hình web chính của Cape như sau:

The screenshot shows the Cape web interface dashboard. At the top, there is a navigation bar with links for Dashboard, Recent, Pending, Search, API, Submit, Statistics, Docs, and Changelog. To the right of the navigation bar is a search bar with placeholder text "Search term as regex" and a "Search" button.

Below the navigation bar, there is a message: "Estimating ~0 analysis per hour, 11 per day."

The main area features two large numerical displays: "41" under "Total tasks" and "6" under "Total samples".

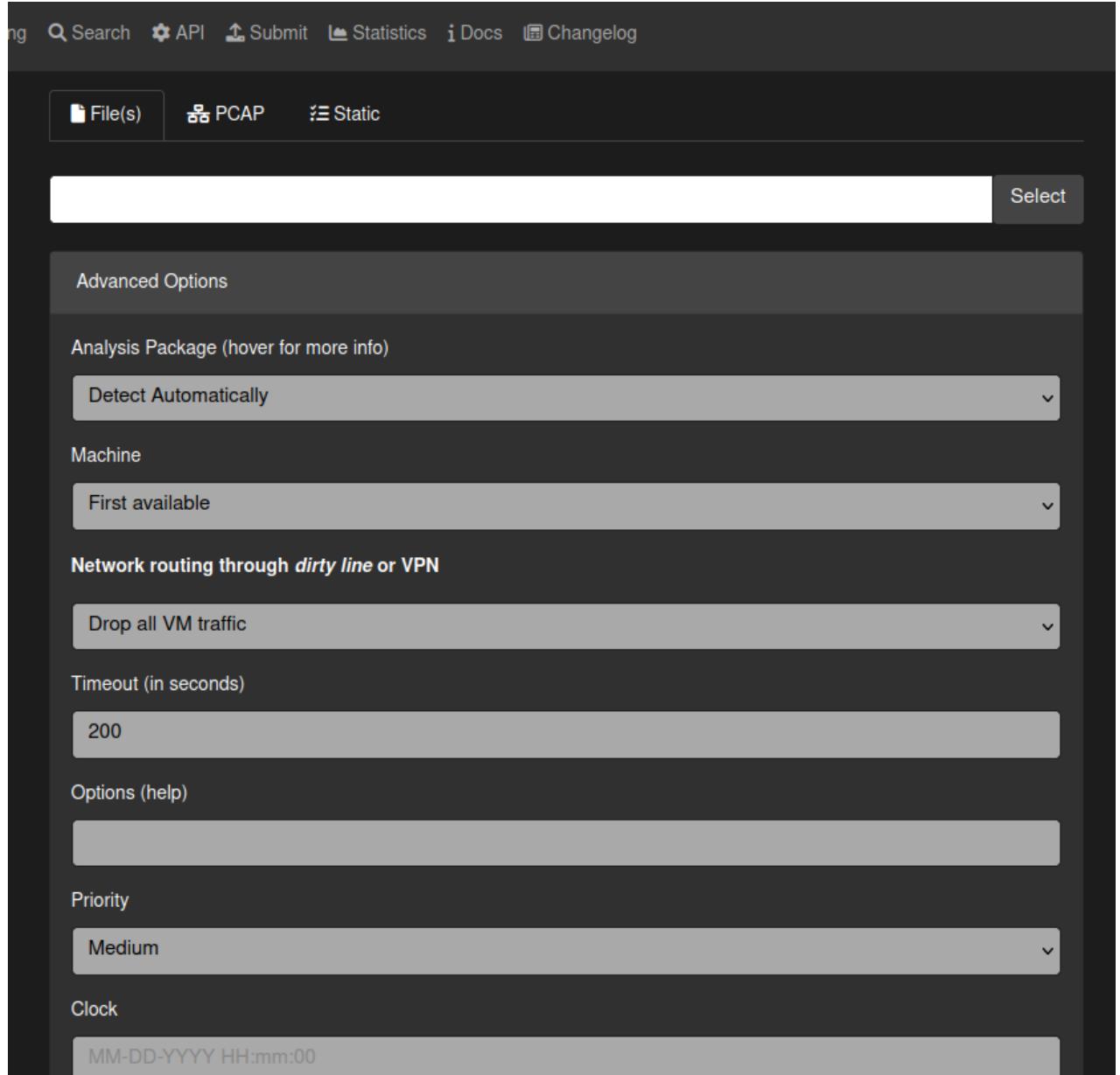
Below these numbers, there are two tables:

- Total tasks:** Shows 3 tasks named "Gozi" with a count of 0.
- Total samples:** Shows 1 sample named "Dynara" with a count of 0.

At the bottom, there is a table showing task states and their counts:

State	Count
pending	0
running	0
distributed	0

Đây là chỗ để submit file cho analyze:



Em tiến hành submit một file mã độc vào để phân tích:

```

2024-12-30 10:17:26.087 [lib.cuckoo.core.machinery_manager] INFO: Task #40: found useable machine win10_64 (arch=x64, platform=windows)
2024-12-30 10:17:26.088 [lib.cuckoo.core.scheduler] INFO: Task #40: Processing task
2024-12-30 10:17:26.158 [lib.cuckoo.core.analysis_manager] INFO: Task #40: File already exists at '/home/nghianguyen/Sandbox/CAPEv2/storage/binaries/a5839e451e46669765657efef3fd5919b730b7f5db1c0ecd932005a3bb9dc988e'
2024-12-30 10:17:26.159 [lib.cuckoo.core.analysis_manager] INFO: Task #40: Starting analysis of FILE '/tmp/cuckoo-tmp/upload_a388q6fy/a5839e451e4666976565.exe'
2024-12-30 10:17:35.083 [lib.cuckoo.core.analysis_manager] INFO: Task #40: Enabled route 'none'.
2024-12-30 10:17:35.084 [modules.auxiliary.QemuScreenshots] INFO: QEMU screenshots module loaded
2024-12-30 10:17:35.085 [lib.cuckoo.core.guest] INFO: Task #40: Starting analysis on guest (lxdwin10_64, ip=192.168.122.100)
2024-12-30 10:17:35.086 [lib.cuckoo.core.guest] INFO: Task #40: Starting analysis on guest (lxdwin10_64, ip=192.168.122.100)
2024-12-30 10:17:37.017 [lib.cuckoo.core.guest] INFO: Task #40: Uploading script files to guest (lxdwin10_64, ip=192.168.122.100)
2024-12-30 10:17:38.139 [lib.cuckoo.core.guest] INFO: Task #40: Started capturing screenshots for win10_64
2024-12-30 10:17:43.437 [lib.cuckoo.core.guest] WARNING: Task #40: Virtual Machine win10_64 /status failed. This can indicate the guest losing network connectivity
2024-12-30 10:17:49.717 [lib.cuckoo.core.guest] WARNING: Task #40: Virtual Machine win10_64 /status failed. This can indicate the guest losing network connectivity
2024-12-30 10:18:14.888 [lib.cuckoo.core.resultsserver] INFO: Task 40: Process 6232 (parent 5432): a5839e451e4666976565.exe, path C:\Users\nghia\AppData\Local\Temp\|a5839e451e4666976565.exe
2024-12-30 10:18:20.332 [lib.cuckoo.core.resultsserver] INFO: Task 40: Process 708 (parent 576): svchost.exe, path C:\Windows\System32\svchost.exe
2024-12-30 10:18:22.410 [lib.cuckoo.core.resultsserver] INFO: Task 40: Process 7140 (parent 576): svchost.exe, path C:\Windows\System32\svchost.exe
2024-12-30 10:18:25.243 [lib.cuckoo.core.resultsserver] INFO: Task 40: Process 2140 (parent 798): WntrPrvSE.exe, path C:\Windows\System32\WntrPrvSE.exe
2024-12-30 10:19:48.607 [lib.cuckoo.core.resultsserver] INFO: Task 40: Process 2988 (parent 798): SecurityHealthHost.exe, path C:\Windows\System32\SecurityHealthHost.exe
2024-12-30 10:19:54.298 [lib.cuckoo.core.resultsserver] INFO: Task 40: Process 6160 (parent 798): RuntimeBroker.exe, path C:\Windows\System32\RuntimeBroker.exe

```

Trên màn hình terminal sẽ thông báo submit thành công, khởi chạy máy ảo để phân tích, đồng thời cũng hiện lên các processes mà mã độc đang đến.

Thông báo sau khi phân tích thành công:

```

2024-12-30 10:21:46,467 [lib.cuckoo.core.guest] INFO: Task #40: Analysis completed successfully (id=win10_64, ip=192.168.122.100)
2024-12-30 10:21:47,261 [lib.cuckoo.core.analysis_manager] INFO: Task #40: Completed analysis successfully.
2024-12-30 10:21:47,264 [lib.cuckoo.core.analysis_manager] INFO: Task #40: analysis procedure completed

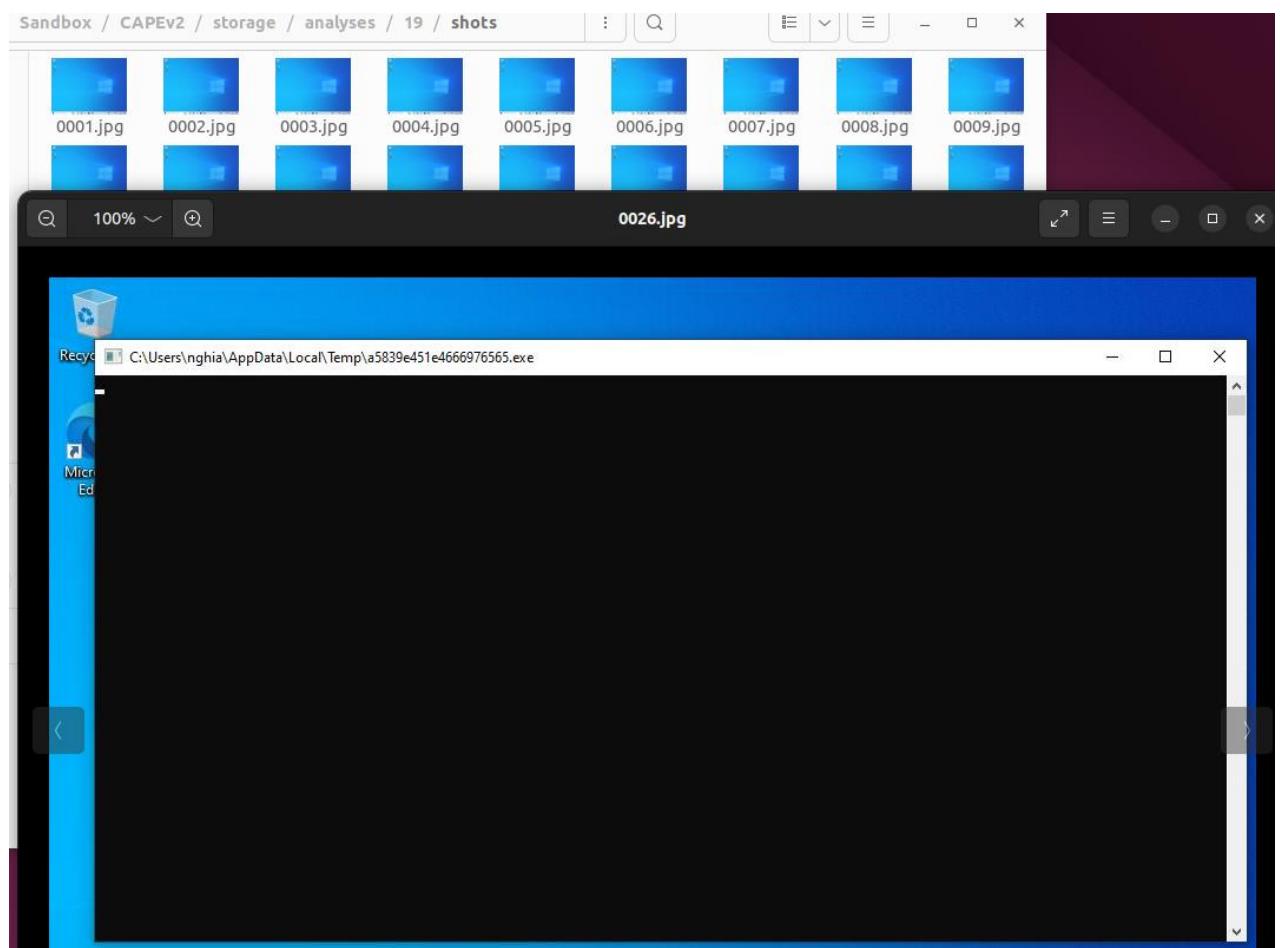
```

Nhưng mà vẫn để nằm ở bộ nhớ ram, nhiều lúc em chạy report thì nó sẽ có thành công hoặc không (bước analyze thì luôn luôn thành công):

Recent Files										
ID	Timestamp	Machine	Package	Filename	MD5	Detections	SuriAlert	VT	Status	
41	2024-12-30 10:25:36	win10_64 (added on)	msi	c5632eaab4462e19c3e87.msi	ec843502815c693bab4554a47a99467	0	-	failed_reporting		
40	2024-12-30 10:21:47	win10_64	exe	a5b939e451e4666976565.exe	8b79949eb899e62cb4f97d296063fb6	0	-	reported		
39	2024-12-30 08:41:39	win10_64 (added on)	msi	c5632eaab4462e19c3e87.msi	ec843502815c693bab4554a47a99467	0	-	failed_reporting		
38	2024-12-30 08:29:41	win10_64 (added on)	exe	packed_discovery.exe	ddb43e70f27b508b5c4da7c50249aacd	0	-	failed_reporting		
37	2024-12-30 08:20:08	win10_64	exe	a5b939e451e4666976565.exe	8b79949eb899e62cb4f97d296063fb6	0	-	reported		
36	2024-12-30 04:51:54	win10_64 (added on)	exe	59fc347dac3dd1c78d62.exe	cc0d0309499150e378a9fed4cd01a0935	Gozi	0	-	failed_reporting	
35	2024-12-30 04:50:40	win10_64	exe	59fc347dac3dd1c78d62.exe	cc0d0309499150e378a9fed4cd01a0935	0	-	reported		
34	2024-12-30 04:49:02	win10_64	exe	59fc347dac3dd1c78d62.exe	cc0d0309499150e378a9fed4cd01a0935	0	-	reported		
33	2024-12-30 04:18:28	win10_64 (added on)	exe	59fc347dac3dd1c78d62.exe	cc0d0309499150e378a9fed4cd01a0935	Gozi	0	-	failed_reporting	
32	2024-12-30 04:09:18	win10_64	exe	59fc347dac3dd1c78d62.exe	cc0d0309499150e378a9fed4cd01a0935	0	-	reported		
31	2024-12-29 09:31:20	win10_64 (added on)	exe	59fc347dac3dd1c78d62.exe	cc0d0309499150e378a9fed4cd01a0935	Gozi	0	-	failed_reporting	

Mặc dù là fail reporting lên web xem trực quan, nhưng em vẫn có thể đọc được file report.json của nó, màn hình screenshot trong lúc chạy mã độc trong thư mục case phân tích của Cape.

Hình chụp khi mã độc được bật:



File report.json:

← → ⌂ file:///home/nghianguyen/Desktop/report.json

JSON Raw Data Headers

Save Copy Collapse All Expand All (slow) Filter JSON

```
statistics:
  processing: []
  signatures: []
  reporting: []
target:
  category: "url"
dropped:
  0:
    name: []
    path: "/home/nghianguyen/Sandbo...819f9ad2a79d1b0b309bc25"
    guest_paths: []
    size: 71757
    crc32: "36866031"
    md5: "e5e3377341056643b0494b6842c0b544"
    sha1: "d53fd8e256ec9d5cef5387872e544a2df9108"
    sha256: "e23040951e464b53b84b11c3...819f9ad2a79d1b0b309bc25"
    sha512: "83f09e48d009a5cf83fa9aa8...d2d6a8085ac2feb1e26c2ef"
    rh_hash: null
    ssdeep: "1536:vAlMWz7vLDtDSVlXXwp...owPFCawP/:vvuWAUxFaoGw/"
    type: "MS Windows icon resource_l, 20x20, 32 bits/pixel"
    yara: []
    cape_yara: []
    clamav: []
    tlsh: "T176638C402A646DD5E824DA...825F8FEBF76C0898C518FC5"
    sha3_384: "228af501320a426ac4f9919a...879ce409417323a7375a154"
    data: null
    die: []
    strings: []
    virustotal:
      cape_type_code: 0
      cape_type: ""
      pid: ""
  1:
  2:
  3:
```

```
    ▼ behavior:
      ▶ processes:          [...]
      anomaly:              []
    ▼ proctree:
      ▼ 0:
        name:                "msiexec.exe"
        pid:                 6316
        parent_id:           1752
        module_path:         "C:\\Windows\\SysWOW64\\msiexec.exe"
        children:             []
        ▶ threads:            [...]
        ▶ environ:            {...}
        ▶ 1:                  {...}
        ▶ 2:                  {...}
        ▶ summary:            {...}
        ▶ enhanced:           [...]
        encryptedbuffers:     []
```

```
    ▼ environ:
        UserName: "nghia"
        ComputerName: "DESKTOP-OEPQVS0"
        WindowsPath: "C:\\Windows"
        TempPath: "C:\\Users\\nghia\\AppData\\Local\\Temp\\"
    ▶ CommandLine: "C:\\Windows\\system32\\D..4BB6-B78D-A8F59079A8D5"
        RegisteredOwner: ""
        RegisteredOrganization: ""
        ProductName: ""
        SystemVolumeSerialNumber: "6edb-959c"
        SystemVolumeGUID: "0fd5b056-0000-0000-0000-300300000000"
        MachineGUID: ""
        MainExeBase: "0x7ff74b850000"
        MainExeSize: "0x00009000"
        Bitness: "64-bit"
    ▶ 2: {...}
    ▶ 3: {...}
    ▶ 4: {...}
    ▼ threads:
        0: "1412"
        1: "876"
        2: "1016"
        3: "1176"
        4: "3684"
        5: "2248"
        6: "780"
    ▶ environ: {...}
    ▼ 2:
        name: "explorer.exe"
        pid: 2196
        parent_id: 4016
```

```
▼ summary:  
  ▼ files:  
    ▶ 0: "C:\\Windows\\WinSxS\\x86_6_none_a863d714867441db"  
    ▶ 1: "C:\\Windows\\WinSxS\\x86_4867441db\\comctl32.dll"  
    2: "C:\\Windows\\WindowsShell.Manifest"  
    3: "C:\\Windows\\Globalization\\Sorting\\sortdefault.nls"  
    4: "C:\\Windows\\SysWOW64\\msi.dll"  
    5: "C:\\Windows\\System32\\msi.dll"  
    6: "\\\\?\\\\SrpDevice"  
    7: "C:\\Windows\\System32\\bcryptPrimitives.dll"  
    8: "C:\\Windows\\System32\\coml2.dll"  
    9: "\\Device\\CNG"  
   ▶ 10: "C:\\Users\\nghia\\AppData\\5632eaa4462e19c3e87.msi"  
   11: "C:\\Windows\\System32\\msctf.dll"  
   12: "C:\\Windows\\System32\\textinputframework.dll"  
   13: "C:\\Windows\\System32\\CoreUIComponents.dll"  
   14: "C:\\Windows\\System32\\CoreMessaging.dll"  
   15: "C:\\Windows\\System32\\ntmarta.dll"  
   16: "C:\\Windows\\System32\\WinTypes.dll"  
   17: "C:\\Windows\\SystemResources\\USER32.dll.mun"  
   18: "C:\\Windows\\System32\\windows.storage.dll"  
   19: "C:\\Windows\\System32\\wlfd.dll"  
   20: "C:\\Windows\\System32\\propsys.dll"  
   --
```

Trên là một số hình ảnh từ file report.json, nó sẽ liệt kê ra các tiến trình chạy, thư mục, file bị ảnh hưởng, cái nào được read hay write từ malware.

Tiếp theo là em cũng có tích hợp thêm vào endpoint của mình URLhaus để check url download về liệu có độc hại hay không.

Đây là file code python để check:

```
#!/var/ossec/framework/python/bin/python3
# Copyright (C) 2015-2022, Wazuh Inc.

import json
import sys
import time
import os
from socket import socket, AF_UNIX, SOCK_DGRAM

try:
    import requests
    from requests.auth import HTTPBasicAuth
except Exception as e:
    print("No module 'requests' found. Install: pip install requests")
    sys.exit(1)

# Global vars

debug_enabled = True
pwd = os.path.dirname(os.path.dirname(os.path.realpath(__file__)))
json_alert = {}
now = time.strftime("%a %b %d %H:%M:%S %Z %Y")

# Set paths
log_file = '{0}/logs/integrations.log'.format(pwd)
socket_addr = '{0}/queue/sockets/queue'.format(pwd)

def main(args):
    debug("# Starting")

    # Read args
    alert_file_location = args[1]

    debug("# File location")
    debug(alert_file_location)

    # Load alert. Parse JSON object.
    with open(alert_file_location) as alert_file:
        json_alert = json.load(alert_file)
    debug("# Processing alert")
    debug(json_alert)

    # Request urlhaus info
```

```

# If positive match, send event to Wazuh Manager
if msg:
    send_event(msg, json_alert["agent"])

def debug(msg):
    if debug_enabled:
        msg = "{0}: {1}\n".format(now, msg)

    print(msg)

    f = open(log_file,"a")
    f.write(msg)
    f.close()

def collect(data):
    urlhaus_reference = data['urlhaus_reference']
    url_status = data['url_status']
    url_date_added = data['date_added']
    url_threat = data['threat']
    url_blacklist_spamhaus = data['blacklists']['spamhaus_dbl']
    url_blacklist_surbl = data['blacklists']['surbl']
    url_tags = data['tags']
    return urlhaus_reference, url_status, url_date_added, url_threat, url_blacklist_spamhaus, url_blacklist_surbl, url_tags

def in_database(data, url):
    result = data['query_status']
    debug(result)
    if result == "ok":
        return True
    return False

def query_api(url):
    params = {'url': url}
    response = requests.post('https://urlhaus-api.abuse.ch/v1/url/', params)
    json_response = response.json()
    if json_response['query_status'] == 'ok':
        data = json_response
        debug(data)
        return data
    else:
        alert_output = {}
        alert_output["urlhaus"] = {}
        alert_output["integration"] = "custom-urlhaus"

```

```

json_response = response.json()
debug("# Error: The URLHAUS Integration encountered an error")
alert_output["urlhaus"]["error"] = response.status_code
alert_output["urlhaus"]["description"] = json_response["errors"][0]["detail"]
send_event(alert_output)
exit(0)

def request_urlhaus_info(alert):
    alert_output = {}
    # If there is no url address present in the alert. Exit.
    if alert["data"]["http"]["redirect"] == None:
        return(0)

    # Request info using urlhaus API
    data = query_api(alert["data"]["http"]["redirect"])

    # Create alert
    alert_output["urlhaus"] = {}
    alert_output["integration"] = "custom-urlhaus"
    alert_output["urlhaus"]["found"] = 0
    alert_output["urlhaus"]["source"] = {}
    alert_output["urlhaus"]["source"]["alert_id"] = alert["id"]
    alert_output["urlhaus"]["source"]["rule"] = alert["rule"]["id"]
    alert_output["urlhaus"]["source"]["description"] = alert["rule"]["description"]
    alert_output["urlhaus"]["source"]["url"] = alert["data"]["http"]["redirect"]
    url = alert["data"]["http"]["redirect"]
    # Check if urlhaus has any info about the url
    if in_database(data, url):
        alert_output["urlhaus"]["found"] = 1

    # Info about the url found in urlhaus
    if alert_output["urlhaus"]["found"] == 1:
        urlhaus_reference, url_status, url_date_added, url_threat, url_blacklist_spamhaus, url_blacklist_surbl, url_tags = collect(data)

        # Populate JSON Output object with urlhaus request
        alert_output["urlhaus"]["urlhaus_reference"] = urlhaus_reference
        alert_output["urlhaus"]["url_status"] = url_status
        alert_output["urlhaus"]["url_date_added"] = url_date_added
        alert_output["urlhaus"]["url_threat"] = url_threat
        alert_output["urlhaus"]["url_blacklist_spamhaus"] = url_blacklist_spamhaus
        alert_output["urlhaus"]["url_blacklist_surbl"] = url_blacklist_surbl
        alert_output["urlhaus"]["url_tags"] = url_tags

```

```

        debug(alert_output)
        return(alert_output)

def send_event(msg, agent = None):
    if not agent or agent["id"] == "000":
        string = '1:urlhaus:{0}'.format(json.dumps(msg))
    else:
        string = '1:[{0}] ({1}) {2}->urlhaus:{3}'.format(agent["id"], agent["name"], agent["ip"] if "ip" in agent else "any", json.dumps(msg))

    debug(string)
    sock = socket(AF_UNIX, SOCK_DGRAM)
    sock.connect(socket_addr)
    sock.send(string.encode())
    sock.close()

if __name__ == "__main__":
    try:
        # Read arguments
        bad_arguments = False
        if len(sys.argv) >= 4:
            msg = '{0} {1} {2} {3} {4}'.format(now, sys.argv[1], sys.argv[2], sys.argv[3], sys.argv[4] if len(sys.argv) > 4 else '')
            debug_enabled = (len(sys.argv) > 4 and sys.argv[4] == 'debug')
        else:
            msg = '{0} Wrong arguments'.format(now)
            bad_arguments = True

        # Logging the call
        f = open(log_file, 'a')
        f.write(msg + '\n')
        f.close()

        if bad_arguments:
            debug("# Exiting: Bad arguments.")
            sys.exit(1)

        # Main function
        main(sys.argv)

    except Exception as e:
        debug(str(e))
        raise

```

Sơ qua thì đoạn code trên sẽ thực hiện nhận đường dẫn file cảnh báo từ Wazuh. Rồi đọc file JSON chứa thông tin cảnh báo.

Lấy URL trong cảnh báo, gửi yêu cầu đến API URLhaus để kiểm tra xem URL có bị đánh dấu là độc hại không.

Nếu có, gửi kết quả (bao gồm thông tin chi tiết) trở lại Wazuh để xử lý.

Config trong file ossec.conf:

```

<integration>
    <name>custom-urlhaus.py</name>
    <hook_url>https://urlhaus-api.abuse.ch/v1/url/</hook_url>
    <rule_id>86601</rule_id>
    <alert_format>json</alert_format>
</integration>

```

Để test thử em đã down thử về một url chứa malware:

```

[!] Command Prompt - ssh nghianguyen@10.11.12.136
nghianguyen@snort:/tmp$ curl -A "Mozi" http://dev.inolab.org/Receipt5142.html --output malware
% Total    % Received % Xferd  Average Speed   Time      Time     Current
          Dload  Upload Total   Spent    Left Speed
100  323k  100  323k    0      0  52263      0  0:00:06  0:00:06  --:--:-- 57823

```

Sau khi down xong lập tức ở Dashboard Wazuh xuất hiện thông báo sau:

Security Alerts					
Time ↓	Technique(s)	Tactic(s)	Description	Level	Rule ID
Jan 15, 2025 @ 18:19:17.127			Suricata: Alert - ET MALWARE PeakLight/Emmenhtal Loader Payload Delivery WebPage Observed	3	86601
Table JSON Rule					
@timestamp	2025-01-15T11:19:17.127Z				
_id	X2Wwa2Q8q826B19uiPqG				
agent.id	002				
agent.ip	192.168.118.100				
agent.name	snort				
data.alert.action	allowed				

data.alert.category	A Network Trojan was detected
data.alert.gid	1
data.alert.metadata.affected_product	Windows_11, Windows_XP_Vista_7_8_10_Server_32_64_Bit
data.alert.metadata.attack_target	Client_Endpoint
data.alert.metadata.confidence	High
data.alert.metadata.created_at	2024_12_11
data.alert.metadata.deployment	SSLDecrypt, Perimeter
data.alert.metadata.malware_family	Emmenhtal, PeakLight
data.alert.metadata.signature_severity	Major
data.alert.metadata.tls_state	TLSDecrypt
data.alert.metadata.updated_at	2024_12_11
data.alert.rev	1

data.alert.signature_id	2058179
data.app_proto	http
data.dest_ip	10.11.12.136
data.dest_port	53602
data.direction	to_client
data.event_type	alert
] data.files >	
{	
"filename": "/Receipt5142.html",	
"size": 40467,	
"stored": false,	
→	
data.flow.bytes_toclient	43794
data.flow.bytes_toserver	1565
data.flow.dest_ip	144.126.138.238
data.flow.dest_port	80
data.flow.pkts_toclient	31
data.flow.pkts_toserver	27
data.flow.src_ip	10.11.12.136
data.flow.src_port	53602
data.flow.start	2025-01-15T11:19:14.684207+0000

data.tx_id	0
decoder.name	json
id	1736939957.178299
input.type	log
location	/var/log/suricata/eve.json
manager.name	nghianguyen-virtual-machine
rule.description	Suricata: Alert - ET MALWARE PeakLight/Emmenhtal Loader Payload Delivery WebPage Observed
rule.firetimes	4
rule.groups	ids, suricata
rule.id	86601
rule.level	3
rule.mail	false
timestamp	2025-01-15T18:19:17.127+0700

**Cập nhật dữ liệu cho ClamAV từ IoC của ThreatIntelligence

Truy cập vào trang chủ của MISP, vào feeds và enable IoC muốn kéo về

nật | <https://192.168.0.130/feeds/index/scope:enabled>

ID	Enabled	Caching	Name	Format	Provider	Org	Source	URL	Headers	Target	Publish	Delta	Override
1	✓	✗	CIRCL OSINT Feed	misp	CIRCL		network	https://www.circl.lu/doc/misp/feed-osint			✗	✗	✗
2	✓	✗	The Botvrij.eu Data	misp	Botvrij.eu		network	https://www.botvrij.eu/data/feed-osint			✗	✗	✗
62	✓	✗	Malware Bazaar	csv	abuse.ch		network	https://bazaar.abuse.ch/export/btx/md5/recent/		Fixed event 1775	✗	✗	✗

Vào feed của Bazaar sau đó tải IoC của các malware về

previous 1 2 3 4 5 6 7 8 9 10 11 12 next > view all

Scope toggle		Deleted	Decay score	Context	Related Tags	Filtering tool	Expand all Objects	Collapse all Attributes	Feed hits	IDS	Distribution	Sightings	Activity	Actions
Date	Type	Value		Tags	Galaxies	Comment	Correlate	Related Events			Enter value to search			
2024-12-30*	84d...3cc	Payload delivery	md5	c8cf0dafd7126eef780f9b233a0ebff	+ + + + +		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Inherit				
2024-12-30*	171...642	Payload delivery	md5	90901b09be6e12a3b4fb8f585774dafa	+ + + + +		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Inherit				
2024-12-30*	59a...787	Payload delivery	md5	aae9a1686546ea72e1ea896ea35370e7	+ + + + +		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Inherit				
2024-12-30*	288...f8e	Payload delivery	md5	e1fbfe1054d3fa3e6d193c60670427b8	+ + + + +		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Inherit				
2024-12-30*	a73...90d	Payload delivery	md5	cb59afea5cdff52b4470b06ac9d94828	+ + + + +		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Inherit				
2024-12-30*	e17...e81	Payload delivery	md5	a3f985a018e2ddfc97ce78fce072bce	+ + + + +		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Inherit				
2024-12-30*	56c...6d0	Payload delivery	md5	9c12a43d9e24058568alc7662714600	+ + + + +		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Inherit				
2024-12-30*	3cd...849	Payload delivery	md5	31ebbedcca9ad62779aad18e6dbdf34bb	+ + + + +		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Inherit				
2024-12-30*	a56...2a7	Payload delivery	md5	dee322b227856477b6b03dd436779bc8	+ + + + +		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Inherit				
2024-12-30*	a7d...266	Payload delivery	md5	b5aeba1a09f5198a71db7337f1b601b6	+ + + + +		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Inherit				
2024-12-30*	dc1...46c	Payload delivery	md5	675f03db23d403573a3a6f708a0e4369	+ + + + +		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Inherit				
2024-12-30*	8fb...a0d	Payload delivery	md5	b0482aa69ec46ca3c3f8cb010ed2d206	+ + + + +		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Inherit				
2024-12-30*	cbf...992	Payload delivery	md5	109277846a6ca7a5e92ec0n0ea83ff62e	+ + + + +		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Inherit				
2024-12-30	023...878	Payload delivery	md5	faa63670c8d713596ac087b84c7f7b0	+ + + + +		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Inherit				
2024-12-30	9a7...aa7	Payload delivery	md5	19177392973853bb37d4f159e8540b70b	+ + + + +		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Inherit				

Hình thông tin các IoC của malware

<https://192.168.0.130/events/view/1775>

The screenshot shows the MISP event details page for event ID 1775. It includes fields like Author, Author org, Author user, Selected Event, Event Level, Analysis, Distribution, Published, Attributes, Recorded change, Last change, and Notifications. A prominent modal dialog is open, titled "Choose the format that you wish to download the event in". The dialog lists several options: MISP JSON, MISP XML, OpenIOC, CSV, STIX 1 XML, STIX 1 JSON, STIX 2, RPZ Zone file, Suricata rules, Snort rules, Bro rules, and Export all attribute values as a text file. Each option has an "Encode Attachments" checkbox. The "Export all attribute values as a text file" option is currently selected. In the top right corner of the browser window, there are two download links: "misp.event.1775 (1).csv" and "misp.event.1775.csv".

Hình chọn cấu trúc tải

Dữ liệu được tải về dưới dạng .csv

Hình nội dung của file .csv được tải về

Sau đó ta sẽ viết 1 chương trình để ghi các dữ liệu này vào file .mdb để bổ sung dữ liệu cho ClamAV

```
# getpy > ...
1 import csv
2
3 # Đường dẫn tệp CSV và tệp .mdb
4 csv_file = 'misp.event.1775.csv' # Thay bằng đường dẫn tệp CSV của bạn
5 mdb_file = 'output.hdb' # Tên tệp .mdb xuất ra
6
7 # Tên malware mặc định
8 malware_name = 'Custom.Malware.Name'
9
10 # Đọc giá trị từ cột "value" và ghi vào tệp .mdb
11 try:
12     with open(csv_file, mode='r', encoding='utf-8') as infile, open(mdb_file, mode='w', encoding='utf-8') as outfile:
13         csv_reader = csv.DictReader(infile)
14         for row in csv_reader:
15             if 'value' in row and row['value']:
16                 hash_value = row['value'].strip() # Lấy hash từ cột "value"
17                 outfile.write(f'{hash_value}:#{malware_name}:80\n') # Thay đổi format theo yêu cầu
18     print("Đã xuất thành công tệp .mdb: {mdb_file}")
19 except FileNotFoundError:
20     print(f"tệp {csv_file} không tồn tại. Vui lòng kiểm tra đường dẫn.")
21 except Exception as e:
22     print(f"Có lỗi xảy ra: {e}")
23 |
```

Hình code

Thử dùng ClamAV để phân tích file malware.zip

MD5 : 88ca7b3de2500e882a7d5525ee37baa1

```
C:\Users\BaoBao>certutil -hashfile D:\TanCongmang\malware.zip MD5
MD5 hash of D:\TanCongmang\malware.zip:
88ca7b3de2500e882a7d5525ee37baa1
CertUtil: -hashfile command completed successfully.
```

```
C:\Users\BaoBao>clamscan "D:\TanCongmang\malware.zip"
Loading:   9s, ETA:  0s [=====]      8.70M/8.70M sigs
Compiling: 3s, ETA:  0s [=====]      41/41 tasks
```

D:\TanCongmang\malware.zip: OK

SCAN SUMMARY

Hình phân tích file trước khi thêm database

```

C:\Users\BaoBao>clamscan "D:\TanCongmang\malware.zip"
Loading: 10s, ETA: 0s [=====>] 8.70M/8.70M sigs
Compiling: 3s, ETA: 0s [=====>] 41/41 tasks

D:\TanCongmang\malware.zip: Custom.Malware.Name.UNOFFICIAL FOUND

----- SCAN SUMMARY -----
Known viruses: 8704121
Engine version: 1.4.1
Scanned directories: 0
Scanned files: 1
Infected files: 1
Data scanned: 1.05 MB
Data read: 0.35 MB (ratio 3.03:1)
Time: 13.582 sec (0 m 13 s)
Start Date: 2024:12:30 18:54:21
End Date: 2024:12:30 18:54:35

```

Hình quét file sau khi thêm database

**Train model để phân biệt malware

Dataset: [MABEL](#)

Model : RandomForest

Kết quả sau khi chạy model để train

Báo cáo phân loại:		precision	recall	f1-score	support
	7ev3n	1.00	1.00	1.00	266
	9002Rat	0.75	0.66	0.70	32
	AESRTRansomware	1.00	1.00	1.00	1
	AXLocker	1.00	0.50	0.67	2
	AceDeceiver	0.00	0.00	0.00	1
	Adhubllka	0.75	1.00	0.86	6
	AdvisorBot	1.00	1.00	1.00	1
	AgendaRansomware	1.00	1.00	1.00	1
	AgentTesla	0.67	0.60	0.63	10
	Amadey	0.50	0.67	0.57	3
	Amavaldo	0.00	0.00	0.00	2
	Andromeda	0.00	0.00	0.00	3
	Arechclient2	0.00	0.00	0.00	1
	AresLoader	0.00	0.00	0.00	1
	Aria-Body	1.00	0.80	0.89	5
	ArkeiStealer	1.00	1.00	1.00	1
	Asbit	0.00	0.00	0.00	2
	AsyncRAT	1.00	0.50	0.67	8
	Aveo	0.00	0.00	0.00	0
	Azorult	0.00	0.00	0.00	3
	Babuk	0.00	0.00	0.00	1
...					
	accuracy			0.95	14134
	macro avg	0.66	0.63	0.63	14134
	weighted avg	0.95	0.95	0.95	14134

Dữ liệu test model : (dữ liệu test này sẽ không có trong bộ dữ liệu khi train model)

File malware down từ bazaar, sau đó viết code python để trích xuất các đặc tính để phân tích tĩnh

Code trích xuất các feature để làm dữ liệu đầu vào cho model

```
train.ipynb ● vc.py analyze.py x 003a903chab3e91ef22602624645342e81b30ec6263f

analyze.py ...
5  def analyze_pe_file(file_path):
6      size_of_optional_header = None
7      if hasattr(pe.OPTIONAL_HEADER, 'SizeOfOptionalHeader'):
8          size_of_optional_header = pe.OPTIONAL_HEADER.SizeOfOptionalHeader
9
10
11      # 12. Size of Headers
12      size_of_headers = pe.OPTIONAL_HEADER.SizeOfHeaders
13
14
15      # Trả về các đặc trưng
16      return {
17          "SHA256 Hash": sha256_hash,
18          "Binary File Size": binary_file_size,
19          "Number of Sections": number_of_sections,
20          "Number of Sections (Decimal)": number_of_sections_decimal,
21          "Pointer to Symbol Table (Decimal)": pointer_to_symbol_table,
22          "Address of Entry Point (Decimal)": address_of_entry_point,
23          "Size of Image (Decimal)": size_of_image,
24          "Magic Number": magic_number
25      }
26
27
PROBLEMS DEBUG CONSOLE TERMINAL PORTS JUPYTER OUTPUT
```

Size of Optional Header: None
Size of Headers: 1024
PS D:\DACHN\ & d:\Python312-mmh\python.exe d:/DACH/analyze.py
SHA256 Hash: 003a903chab3e91ef22602624645342e81b30ec6263fle62165abd479c48942e
Binary File Size: 401408
Number of Sections: 6
Number of Sections (Decimal): 6
Pointer to Symbol Table (Decimal): 0
Address of Entry Point (Decimal): 158849
Size of Image (Decimal): 401408
Magic Number: 332
Characteristics: 258
Size of Code: 279568
Size of Initialized Data: 102912
Size of Uninitialized Data: 0
Size of Optional Header: None
Size of Headers: 1024

Hình 1 số thông tin liên quan tới kích thước file

```
train.ipynb ● vc.py analyze.py test.exe analyze1.py > ...

analyze1.py > ...
1 import pefile
2 import hashlib
3 import math
4 import os
5 from capstone import *
6 from capstone import *
7 from capstone.x86 import X86_REG_EBP # Import định nghĩa X86_REG_EBP
8
9 # Hàm tính entropy cho một đoạn dữ liệu
10 def calculate_entropy(data):
11     if len(data) == 0:
12         return 0
13     byte_count = [0] * 256
14     for byte in data:
15         byte_count[byte] += 1
16     entropy = 0.0
17     for count in byte_count:
18         if count > 0:
19             p = float(count) / len(data)
20             entropy -= p * math.log2(p)

PROBLEMS 3 DEBUG CONSOLE TERMINAL PORTS JUPYTER OUTPUT

● PS D:\DACN> & d:/Python312-mmh/python.exe d:/DACN/analyze1.py
sha256_hash: 003a903cbab3e91ef22602624645342e81b30ee6263f1e62165abd479c48942e
entropy: 6.093243225069396
execution_section_entropy: 6.437423313355524
binary_file_size: 491408
unique_mnemonics: 192
function_calls_prologue: 887
function_calls_immediate_address: 261
```

Hình 1 số feature có liên quan tới func, bytecode

Kiểm tra với malware test :

```
> ^      ytest_pred = rf_model.predict (Xtest)
      # Tính độ chính xác
      accuracy = accuracy_score(ytest, ytest_pred)
      print(ytest_pred)
      print(f"Độ chính xác của mô hình Random Forest: {accuracy * 100:.2f}%")
[99] ✓ 0.0s
...
['7ev3n' '7ev3n']
Độ chính xác của mô hình Random Forest: 100.00%
+ Code + Markdown
```

Link Drive chứa demo:

<https://drive.google.com/drive/folders/1F8rA03rZNAHoCQsrOqtKQqhpdw4pOkDP?usp=sharing>

VII. KẾT LUẬN

Thông qua đồ án này thì nhóm đã triển khai thêm một vài giải pháp khác nhau nhằm giúp nâng cấp các antivirus truyền thống chỉ sử dụng signature cho việc phát hiện malware. Ngoài ra còn có một số tính năng giúp bảo vệ cho endpoint, đây sẽ là bước đệm để nhóm em phát triển đồ án này theo hướng EDR hay XDR. Nhìn chung thì nhóm em vẫn còn một số tính năng vẫn chưa integrate được với mô hình và một vài thứ vẫn chưa được hoàn hảo lắm, điều này sẽ được nhóm khắc phục, phát triển trong tương lai.

VIII. TÀI LIỆU THAM KHẢO

<https://docs.clamav.net/>

<https://documentation.wazuh.com/current/index.html>

<https://usbguard.github.io/>

<https://capev2.readthedocs.io/en/latest/installation/index.html>