-

Bài thực hành: Tìm hiểu về công cụ Volatility

**TÌM HIỂU BÀI THỰC HÀNH**

**PTIT - Volatility**

**Giảng viên:** Đỗ Xuân Chợ

**Họ và tên sinh viên:** Nguyễn Quốc Vượng

**Mã sinh viên:** B21DCAT227

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA AN TOÀN THÔNG TIN**

* + 1. Nội dung và hướng dẫn bài thực hành
  1. Mục đích

Volatility là một khung phần mềm mã nguồn mở dành cho phân tích bộ nhớ và phản ứng sự cố phần mềm độc hại. Nó được viết bằng Python và hỗ trợ các hệ điều hành như Microsoft Windows, MacOS và Linux.

Sinh viên có thể sử dụng Volatility để phân tích và trích xuất thông tin quan trọng từ một bản sao bộ nhớ. Điều này cho phép người phản hồi sự cố xác định khi một cuộc tấn công xảy ra, từ đâu nó bắt nguồn, và nhiều hơn nữa.

Volatility là một công cụ dựa trên Python cho phép trích xuất thông tin từ bộ nhớ đệm, như các quá trình, kết nối mạng, khóa registry, mật khẩu, khóa mã hóa, và nhiều hơn nữa. Volatility cũng có thể thực hiện các nhiệm vụ nâng cao, như phát hiện phần mềm độc hại, phân tích dòng thời gian, khắc phục lỗi bộ nhớ, và tiêm mã. Volatility hoạt động bằng cách sử dụng các plugin cụ thể cho mỗi hệ điều hành và định dạng bộ nhớ. Bạn cũng có thể tạo ra các plugin của riêng mình hoặc sử dụng các plugin do cộng đồng đóng góp.

* 1. Yêu cầu đối với sinh viên

Sinh viên cần có kiến thức cơ bản về hệ điều hành Linux và các khái niệm cơ bản về phân tích malware để hiểu và áp dụng công cụ Volatility. Hiểu biết về các công cụ như VirusTotal có thể hữu ích để bổ sung kiến thức trong quá trình sử dụng Volatility.

* 1. Nội dung thực hành

Volatility có 2 phiên bản, cụ thể là Volatility 2 và Volatility 3. Trong bài lab này, sinh viên sẽ sử dụng Volatility 2 vì nó có một số lượng lớn các plugin hữu ích.

**Khởi động bài lab:**

- Vào terminal, gõ:

*labtainer ptit- volatility*

A computer screen shot of a program

Description automatically generated

*(chú ý: sinh viên sử dụng mã sinh viên của mình để nhập thông tin email người thực hiện bài lab khi có yêu cầu, để sử dụng khi chấm điểm)*

**Các nhiệm vụ:**

- Task 1: Xác định thông tin mẫu.

+ Sau khi khởi động xong 1 terminal ảo sẽ xuất hiện. Ở thư mục hiện tại đã có một mẫu bộ nhớ là cridex.vmem.

+ Sinh viên có thể sử dụng plugin “imageinfo” để có thông tin về mẫu bộ nhớ.

*volatility -f cridex.vmem imageinfo*

*A screenshot of a computer

Description automatically generated*

* Profile của mẫu bộ nhớ này là gì?

Profile của mẫu bộ nhớ này: WinXPSP2X86

+ Thông tin này cho khởi đầu cần thiết để thực hiện lệnh phân tích tiếp theo bằng các plugin khác. Sinh viên cần phải ghi nhớ Suggested Profiles để thay thế vào các lệnh sau này.

*--profile=PROFILE (cần phải thay thế PROFILE)*

- Task 2: Xác định tiến trình độc hại.

+ Có một số plugin trong Volatility cho phép xem các tiến trình đang chạy. Một số trong số này bao gồm: psscan, pstree, pslist, psxview.

+ Sinh viên có thể xem những tiến trình đã được chấm dứt trước đó và những tiến trình nào đã bị rootkit hủy liên kết bằng cách sử dụng “psscan”.

*volatility -f cridex.vmem --profile=PROFILE psscan*

*A screenshot of a computer screen

Description automatically generated*

+ Plugin “pslist” hiển thị địa chỉ bắt đầu, tên tiến trình, PID, PPID, số lượng luồng, số lượng handles, và ngày/giờ.

*volatility -f cridex.vmem --profile=PROFILE pslist*

*A screenshot of a computer

Description automatically generated*

+ Ngoài ra, sinh viên cũng có thể sử dụng plugin “pstree” như một phương thức thay thế cho pslist nhưng với cấu trúc cây để thể hiện mối quan hệ giữa các tiến trình:

*volatility -f cridex.vmem --profile=PROFILE pstree*

*A screenshot of a computer

Description automatically generated*

* Trong danh sách này có 2 tiến trình lạ, sinh viên hãy cho biết 2 tiến trình đó là gì?
* PID của chúng là gì? Tiến trình nào là tiến trình cha và tiến trình nào là tiến trình con?

+ Trước khi tiếp tục, hãy chạy plugin “psxview” để tham chiếu chéo các tiến trình với nhiều danh sách khác nhau. “psxview” được sử dụng để xem và phân tích các tiến trình ẩn và các kỹ thuật che giấu khác trên hệ thống.

*volatility -f cridex.vmem --profile=PROFILE psxview*

*A screenshot of a computer

Description automatically generated*

* Trong kết quả có phát hiện ra tiến trình ẩn nào không? Tại sao?

+ “cmdline” là plugin được sử dụng để trích xuất và hiển thị các đối số của dòng lệnh được sử dụng khi khởi động các tiến trình trong hệ thống.

*volatility -f cridex.vmem –profile=PROFILE cmdline*

*A screenshot of a computer program

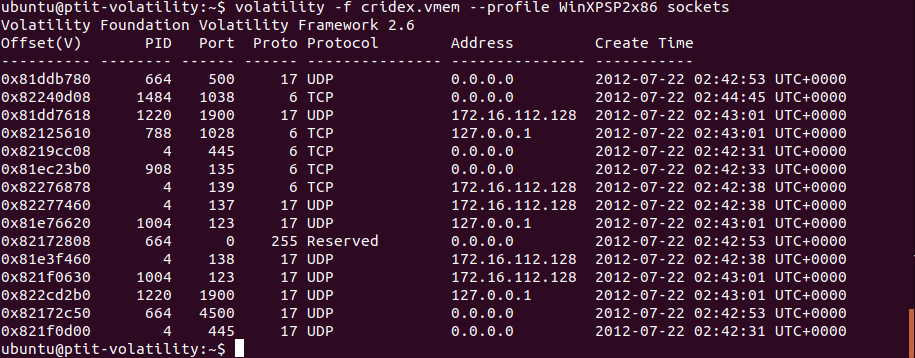
Description automatically generated*

* Sinh viên hãy cho biết đường dẫn của tiến trình độc hại là gì?

- Task 3: Xác định kết nối mạng độc hại.

+ Để xem các kết nối mở trên máy tính. Sinh viên có thể chạy các plugin như “sockets”.

*volatility -f cridex.vmem --profile=PROFILE sockets*

**

+ Ngoài ra, “connscan” là một công cụ quét các kết nối TCP và sockets sẽ phát hiện các socket đang lắng nghe cho bất kỳ giao thức nào (TCP, RAW, UDP, vv).

*volatility -f cridex.vmem --profile=PROFILE connscan*

*A computer screen with numbers and letters

Description automatically generated*

* Có kết nối nào đang được sử dụng bởi PID vừa tìm được ở nhiệm vụ kia không?
* Kết nối đó giao tiếp với địa chỉ IP nào? Qua cổng nào ở địa chỉ local?

- Task 4: Trích xuất tiến trình và bộ nhớ độc hại.

+ Hoạt động bất thường từ hai tiến trình (explorer.exe và reader\_sl.exe) gợi ý rằng có thể có một số mã độc được chèn vào. Để kiểm tra độ chính xác, sinh viên có thể sử dụng plugin “malfind” để xác nhận.

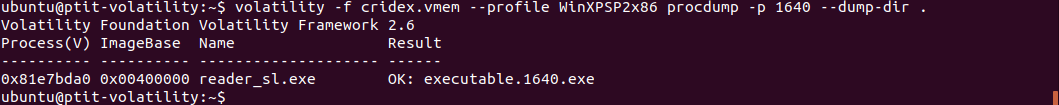
*volatility -f cridex.vmem --profile=PROFILE malfind*

*A computer screen shot of a computer

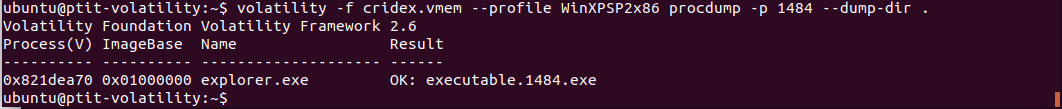
Description automatically generated*

+ Tiếp theo, sinh viên có thể phân tích sâu hơn bằng cách sử dụng các plugin “memdump” và “procdump” để trích xuất tiến trình và bộ nhớ.

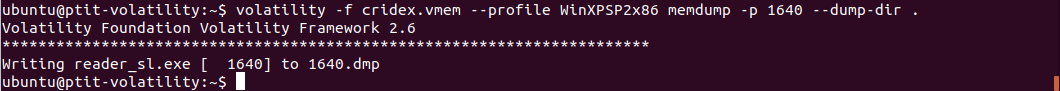
*volatility -f cridex.vmem --profile=PROFILE procdump -p 1640 --dump-dir .*

**

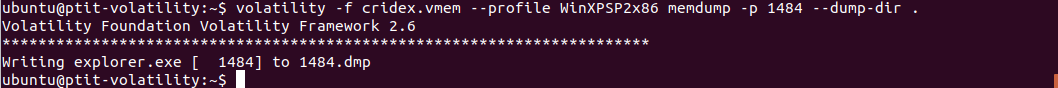
*volatility -f cridex.vmem --profile=PROFILE procdump -p 1484 --dump-dir .*

**

*volatility -f cridex.vmem --profile=PROFILE memdump -p 1640 --dump-dir .*

**

*volatility -f cridex.vmem --profile=PROFILE memdump -p 1484 --dump-dir .*

**

+ Để phân tích các lệnh file **.dmp**, sinh viên có thể sử dụng “strings” và “grep” để tìm mối tương quan với địa chỉ IP 41.168.5.140.

*strings 1640.dmp | grep -Fi "41.168.5.140" -C 5*

*A computer screen with white text

Description automatically generated*

*strings 1484.dmp | grep -Fi "41.168.5.140" -C 5*

*A computer screen shot of a program

Description automatically generated*

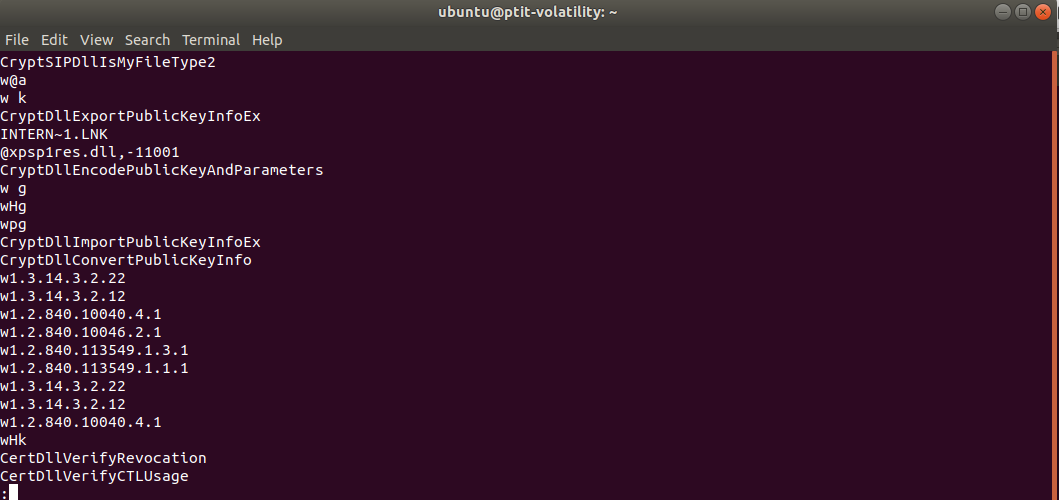
+ Nếu chạy “strings” và “less”, chúng ta có thể tìm thấy một lượng đáng kể các tên miền ngân hàng và tài chính.

*strings 1640.dmp | less*

*A black background with white dots

Description automatically generated with medium confidence*

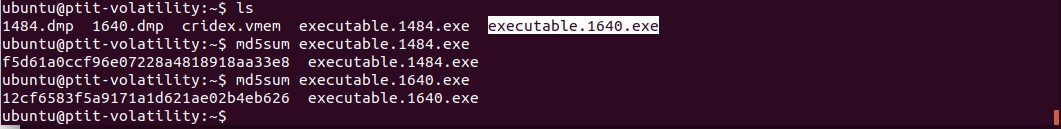
*strings 1484.dmp | less*

**

* Sinh viên hãy cho liệt kê 1 số các tên miền và các địa chỉ IP đáng nghi có trong bộ nhớ.

+ Còn 2 file **.exe** vừa trích xuất được, để phân tích chi tiết hơn thì chúng ta cần làm phân tích tĩnh và động chuyên sâu. Tuy nhiên, sinh viên có thể tạo mã hash và kiểm tra trên các trang công cụ phân tích online như VirusTotal hay Intezer để biết thêm thông tin.

*md5sum <file>*

**

* Sinh viên hãy cho biết 2 chương trình bất thường kia là loại mã độc gì?

*A screenshot of a computer

Description automatically generated*

*A screenshot of a computer

Description automatically generated*

+ Sau đó, sinh viên kết nối đến 1 container khác thông qua ssh bằng lệnh sau:

*ssh ubuntu@192.168.0.20*

*A computer screen shot of a computer program

Description automatically generated*

+ Mật khẩu chính là 1 trong 2 mã hash md5 mà sinh viên vừa tìm được ở bước trên. Tiếp theo, sinh viên hãy in ra màn hình tệp tin filetoview:

*cat filetoview*

*A screenshot of a computer

Description automatically generated*

**Kết thúc bài lab:**

- Trên terminal đầu tiên sử dụng câu lệnh sau để kết thúc bài lab:

*stoplab ptit- volatility*

- Khi bài lab kết thúc, một tệp zip lưu kết quả được tạo và lưu vào một vị trí được hiển thị bên dưới stoplab.

A computer screen shot of a computer code

Description automatically generated

**Khởi động lại bài lab:**

- Trong quá trình làm bài sinh viên cần thực hiện lại bài lab, dùng câu lệnh:

*labtainer -r ptit- volatility*