

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

-----o0o-----



**BÁO CÁO ĐỒ ÁN**

**ĐỀ TÀI: Evaluation of Live Forensic Techniques in Ransomware Attack Mitigation**

**(Đánh giá các kỹ thuật pháp y theo thời gian thực trong việc giảm thiểu tấn công mã độc tống tiền)**

Giảng viên hướng dẫn: Lê Đức Thịnh

Sinh viên thực hiện:

21522067 - Lê Huy Hiệp

21520740 - Lê Trí Dũng

21522756 - Nguyễn Thanh Tuấn

21520690 - Chu Anh Đạt

Lớp: Pháp chứng kỹ thuật số (NT334.P11.ANTT)

**TP. Hồ Chí Minh, ngày 3 tháng 11 năm 2024**

# LỜI CẢM ƠN

Lời đầu tiên, xin chân thành cảm ơn Ban Giám hiệu Trường Đại học Công nghệ thông tin, khoa Mạng máy tính và truyền thông và thầy Lê Đức Thịnh đã tạo nền tảng, điều kiện giúp chúng tôi có hội tiếp cận và nghiên cứu chủ đề này.

Chúng tôi cũng muốn bày tỏ lòng biết ơn sâu sắc đến các tác giả của bài báo khoa học mà chúng tôi đã sử dụng làm cơ sở cho nghiên cứu của mình. Công trình nghiên cứu của quý vị đã cung cấp cho tôi cơ sở lý thuyết và dữ liệu quan trọng để phát triển và thực hiện dự án này.

Trong quá trình nghiên cứu về chủ đề này, có thể do hiểu biết còn nhiều hạn chế nên bài làm khó tránh khỏi những thiếu sót. Chúng tôi rất mong nhận được những lời góp ý chân thành của thầy để bài báo cáo ngày càng hoàn thiện hơn.

Trân trọng.

<b>LỜI CẢM ƠN .....</b>	<b>2</b>
<b>NHẬN XÉT CỦA GIẢNG VIÊN.....</b>	<b>3</b>
<b>1. Mở đầu .....</b>	<b>5</b>
<b>2. Các công việc liên quan .....</b>	<b>5</b>
<b>2.1. Pháp y theo giới gian thực và thu thập bộ nhớ .....</b>	<b>6</b>
<b>2.2. Phân tích mật mã trong Live Forensics.....</b>	<b>6</b>
<b>2.2.1. Các phương pháp kiểm tra bộ nhớ .....</b>	<b>7</b>
<b>2.2.2. Xác định các khóa trong bộ nhớ .....</b>	<b>7</b>
<b>2.2.3. Xác định khóa AES.....</b>	<b>7</b>
<b>3. Thiết kế .....</b>	<b>9</b>
<b>3.1. Thiết kế môi trường thử nghiệm ransomware.....</b>	<b>9</b>
<b>3.2. Thiết kế thí nghiệm.....</b>	<b>9</b>
<b>3.2.1. Phần 1 của thí nghiệm - Xác định khóa trong bộ nhớ .....</b>	<b>9</b>
<b>3.2.2. Phần 2 của thí nghiệm - Tạo dòng thời gian của khóa.....</b>	<b>10</b>
<b>3.2.3. Phần 3 của thí nghiệm - Xác thực các khóa tìm thấy.....</b>	<b>11</b>
<b>3.2.4. Thí nghiệm kết hợp.....</b>	<b>11</b>
<b>4. Thực hiện và kết quả .....</b>	<b>12</b>
<b>4.1. Lựa chọn mẫu ransomware .....</b>	<b>12</b>
<b>4.1.1. Thông tin chi tiết về các mẫu ransomware .....</b>	<b>13</b>
<b>4.1.1.a. NotPetya .....</b>	<b>13</b>
<b>4.1.1.b. Bad Rabbit .....</b>	<b>14</b>
<b>4.1.1.c. Phobos .....</b>	<b>15</b>

4.1.1.d. Annabelle.....	16
4.1.2. Các ransomware khác .....	17
4.2. Cấu hình máy thử nghiệm .....	17
4.3. Các thử nghiệm .....	18
4.3.1. Thử nghiệm phần 1 - Xác định khóa trong bộ nhớ .....	18
4.3.2. Thử nghiệm phần 2 - Tạo biểu đồ thời gian cho khóa .....	19
4.3.3. Thử nghiệm 3 - Xác thực khóa được tìm thấy .....	19
5. Demo và kết quả .....	19
5.1. Not Petya .....	19
I. Xác định khóa trong bộ nhớ.....	19
II. Tạo dòng thời gian của khóa .....	21
III. Xác thực các khóa tìm thấy.....	22
5.2. Bad Rabbit.....	23
5.2.1. Win7 .....	23
I. Xác định khóa trong bộ nhớ.....	23
II. Tạo dòng thời gian của khóa .....	25
III. Xác thực các khóa tìm thấy.....	26
5.2.2. Win10 .....	27
I. Xác định khóa trong bộ nhớ.....	27
II. Tạo dòng thời gian của khóa .....	28
III. Xác thực các khóa tìm thấy.....	29
5.3. Phobos.....	32
III. Xác thực các khóa tìm thấy.....	34
5.4. Annabelle .....	35
I. Xác định khóa trong bộ nhớ.....	35
II. Tạo dòng thời gian của khóa .....	37
III. Xác thực các khóa tìm thấy.....	38
Tham khảo.....	40

## NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....



## 1. Mở đầu

Bài nghiên cứu này nhằm xác định liệu các kỹ thuật hiện được sử dụng trong lĩnh vực pháp y kỹ thuật số có thể được tận dụng để tìm ra các khóa mã hóa do phần mềm độc hại sử dụng hay không, qua đó giảm thiểu tác động của một cuộc tấn công ransomware.

Có một lĩnh vực nghiên cứu riêng biệt tập trung vào việc nghiên cứu và phát triển các kỹ thuật pháp y theo thời gian thực, đặc biệt là phân tích bộ nhớ trực tiếp. Một số nghiên cứu trong lĩnh vực này đã nhằm mục đích phục hồi các mảnh mã hóa, cụ thể là các khóa mã hóa, từ nội dung bộ nhớ của hệ thống nơi các quá trình mã hóa đang hoạt động. Rất nhiều nghiên cứu trong lĩnh vực này đã đạt được thành công lớn, cho phép các nhà nghiên cứu xác định được các khóa mã hóa mà các chương trình mã hóa đang sử dụng và sau đó dùng chúng để giải mã các tập tin. Tuy nhiên, không có nghiên cứu cụ thể nào được tìm thấy trong đó các kỹ thuật này đã được áp dụng cho các máy đang có ransomware hoạt động.

Những đóng góp chính của nghiên cứu này là:

- Đầu tiên, cung cấp xác nhận rằng các kỹ thuật pháp y có thể được sử dụng để điều tra bộ nhớ để biến đổi của các máy bị lây nhiễm ransomware và có thể khai thác các đoạn mã hóa hữu ích để giảm thiểu tác động của cuộc tấn công.
- Thứ hai, tạo dòng thời gian về hành vi của cuộc tấn công ransomware và nêu rõ thời gian khóa mã hóa có sẵn để trích xuất.

## 2. Các công việc liên quan

Ransomware là một trong những mối đe dọa phổ biến và gây thiệt hại nhất mà người dùng internet phải đối mặt hiện nay và thường được phân loại theo loại mã hóa được sử dụng

- Ransomware mã hóa đối xứng (Symmetric Crypto-Ransomware - SCR) sử dụng một khóa cho cả mã hóa và giải mã, cho phép cuộc tấn công hoàn thành trong thời gian ngắn hơn, giảm thiểu khả năng bị phát hiện.
- Ransomware mã hóa bất đối xứng (Asymmetric Crypto-Ransomware - ACR) sử dụng các khóa khác nhau cho mã hóa và giải mã.
- Ransomware mã hóa khóa kết hợp (Hybrid Key Crypto-Ransomware - HCR) trước tiên sử dụng mã hóa đối xứng để mã hóa các tệp của người dùng nhanh nhất có thể. Sau đó, khóa đối xứng này được mã hóa bằng mã hóa bất đối xứng.

## 2.1. Pháp y theo thời gian thực và thu thập bộ nhớ

Các phương pháp phân tích tĩnh tĩnh được sử dụng để phân tích bằng chứng từ một hệ thống máy tính đã bị tắt. Vấn đề với cách tiếp cận này là thông tin quan trọng được lưu trữ trong bộ nhớ dễ bay hơi (ví dụ: RAM) của máy tính sẽ bị mất khi máy bị tắt. Các ví dụ về thông tin có thể có trong bộ nhớ bao gồm các khóa mã hóa, chi tiết kết nối mở, các tiến trình đang chạy, người dùng đang đăng nhập, v.v.

Để giải quyết vấn đề này, một cách tiếp cận pháp y bổ sung được gọi là pháp y theo thời gian thực (live forensics) đã được phát triển, nó chủ yếu nhấn vào dữ liệu dễ bay hơi của máy tính, chỉ có thể thu thập được từ một hệ thống đang chạy. Các kỹ thuật này có thể kết hợp với các kỹ thuật phân tích phần mềm độc hại.

Một khía cạnh quan trọng của pháp y theo thời gian thực là kiểm tra bộ nhớ của hệ thống nơi mã độc đang chạy. Việc kiểm tra này thường được thực hiện ngoại tuyến để nội dung bộ nhớ không bị ảnh hưởng bởi các công cụ kiểm tra hoặc thu thập bộ nhớ. Để thực hiện điều này, bộ nhớ của hệ thống cần được thu thập và lưu lại.

Có ba kỹ thuật chính để thu thập bộ nhớ:

1. **Dựa trên phần mềm (Software-based)**, thường liên quan đến việc thực thi các chương trình trích xuất. Một vấn đề với cách tiếp cận này là việc thực thi phần mềm sẽ ảnh hưởng đến nội dung bộ nhớ của hệ thống bị thu thập.
2. **Dựa trên phần cứng (Hardware-based)**, thường bao gồm việc kết nối các thiết bị như thẻ PCMCIA hoặc USB, và không phải lúc nào cũng thực tế trong các tình huống trực tiếp vì yêu cầu truy cập vật lý vào máy tính [31].
3. **Các kỹ thuật dựa trên công nghệ ảo hóa (Virtualization)**,

Sử dụng phương pháp ảo hóa, một snapshot của bộ nhớ dễ bay hơi của hệ thống được phân tích sẽ được trích xuất bằng các công cụ do phần mềm ảo hóa cung cấp. Snapshot này sau đó được kiểm tra bởi một nhà phân tích sử dụng các công cụ pháp y chuyên dụng. Rõ ràng, để sử dụng phương pháp này, hệ thống phải đang chạy trong môi trường ảo hóa. *Ưu điểm* của cách tiếp cận này là không có dấu vết nào của bất kỳ chương trình trích xuất nào tồn tại trong bộ nhớ được thu thập, và các chương trình độc hại đang chạy không nhận ra rằng chúng đang bị phân tích hoặc rằng việc trích xuất bộ nhớ đã diễn ra.

## 2.2. Phân tích mật mã trong Live Forensics

Một số nghiên cứu trước đây đã được thực hiện về khả năng sử dụng các kỹ thuật pháp y theo thời gian thực để khám phá các khóa mã hóa có thể có trong bộ nhớ của máy tính. Tuy nhiên, không thể tìm thấy tài liệu nào tập trung cụ thể vào ransomware, nhưng các nghiên cứu tương tự về việc xác định khóa trong bộ nhớ dễ bay hơi đã được thực hiện đối với các đường hầm SSH, các ổ đĩa được mã hóa, WinRAR, WinZip và Skype.

Nhiều bài báo nghiên cứu xác nhận rằng để hệ thống có thể mã hóa/giải mã dữ liệu, thuật toán mật mã cần phải có quyền truy cập vào các khóa mã hóa và chúng thường được lưu trữ trong bộ nhớ dễ bay hơi. Balogh [4] cho biết rằng mã hóa trong thời gian thực chỉ được thực hiện trong bộ nhớ, điều này có nghĩa là các khóa mã hóa cũng phải

có mặt ở đó. Trong trường hợp mã hóa đối xứng, điều này có nghĩa là các khóa cần thiết cho việc giải mã cũng có thể được phục hồi từ bộ nhớ. Về quản lý khóa, Maartmann-Moe [26] tuyên bố rằng rõ ràng là các khóa mật mã cần phải có trong bộ nhớ trong quá trình mã hóa khi sử dụng phần cứng máy tính tiêu chuẩn.

Trong các thử nghiệm rộng rãi được tiến hành trên 10 hệ thống mật mã khác nhau, các nhà nghiên cứu [26] luôn có thể truy xuất tất cả các khóa mật mã từ bộ nhớ cho mọi ứng dụng được thử nghiệm, sử dụng công cụ chuyên dụng có tên là **interrogate**. Mặc dù các nhà nghiên cứu này chưa điều tra cụ thể về ransomware, phát hiện của họ cho thấy rằng có thể trích xuất các khóa mật mã của ransomware bằng các kỹ thuật tương tự.

### 2.2.1. Các phương pháp kiểm tra bộ nhớ

Quá trình thông thường để tìm một thứ gì đó là cố gắng xác định một đặc điểm nào đó và sau đó tìm kiếm đặc điểm đó. Một đặc điểm của các khóa mật mã là chúng thường được chọn ngẫu nhiên. Hầu hết mã và dữ liệu không được chọn ngẫu nhiên, và sự khác biệt này là đáng kể. Khi dữ liệu ngẫu nhiên, nó có entropy cao hơn so với dữ liệu có cấu trúc. Điều này có nghĩa là có thể định vị các khóa mật mã giữa các dữ liệu khác bằng cách tìm các đoạn có entropy cao bất thường. Các tác giả phát hiện rằng khối bộ nhớ chứa khóa chính và khóa phụ AES có cấu trúc nhận diện được và có entropy cao.

Trong thực tế, các khóa mã hóa đối xứng chỉ là các chuỗi ngắn dữ liệu trông ngẫu nhiên, thường dài từ 16 đến 32 byte và tồn tại giữa các đoạn dữ liệu có entropy thấp hơn.

### 2.2.2. Xác định các khóa trong bộ nhớ

Để trích xuất các khóa mã hóa từ bộ nhớ, chúng phải được xác định trước. Một số nhà nghiên cứu đã phát hiện rằng các khóa mã hóa trong bộ nhớ có cấu trúc hơn nhiều so với những gì đã tin trước đây, và một số chiến lược để định vị các khóa đã được đề xuất và thảo luận dưới đây.

- Sử dụng phương pháp tìm kiếm entropy cao như đã đề xuất và thử nghiệm bởi Shamir [44]. Vì người ta biết rằng dữ liệu khóa có entropy cao hơn so với dữ liệu không phải là khóa, một cách để định vị khóa là chia dữ liệu thành các phần nhỏ, đo entropy của mỗi phần và hiển thị các vị trí có entropy đặc biệt cao.
- Tìm kiếm các mẫu cụ thể trong bộ nhớ như **key schedule (lịch trình khóa)**, đặc trưng cho một số loại mã hóa. Các mẫu này cũng có thể bao gồm các offsets bộ nhớ đã biết, các chiều dài cụ thể của các vị trí bộ nhớ có **entropy cao** hoặc các mẫu entropy đã biết.

### 2.2.3. Xác định khóa AES

Từ các mẫu ransomware hiện tại, người ta đã xác định rằng phần lớn các ransomware mã hóa hiện đại là dạng hybrid (kết hợp - HCR). Khóa công khai của mã hóa bất đối xứng được cung cấp cùng với ransomware, trong khi khóa bí mật được giữ bởi kẻ tấn công. Vì khóa bí mật của mã hóa bất đối xứng không bao giờ có mặt trên máy tính, bài viết này sẽ tập trung vào việc xác định khóa được sử dụng trong giai đoạn mã hóa đối

xứng của ransomware, mà trong phần lớn các trường hợp là khóa AES (Advanced Encryption Standard).

AES là một thuật toán mã hóa Substitution-Permutation (SP)-network, hoạt động trên các khối 128-bit và có thể sử dụng các khóa dài 128, 192 hoặc 256 bit. Nó được một số nhà nghiên cứu coi là gần như không thể phá vỡ và không thể giải mã nếu không có khóa chính xác.

Các hệ thống mật mã khóa đối xứng hiện đại được xây dựng bằng cách áp dụng lặp đi lặp lại một hàm đơn giản với nhiều vòng lặp. Từ khóa chính, một hàm dẫn xuất sinh ra các khóa phụ khác nhau được sử dụng trong mỗi “vòng lặp”. Đây được gọi là thuật toán key schedule và nhiều nhà nghiên cứu đã tuyên bố rằng thông tin này cần phải có trong bộ nhớ. Việc nắm rõ hệ thống mật mã này có thể được sử dụng để tìm kiếm mẫu khóa trong bộ nhớ. Các tiêu chí tìm kiếm là mối quan hệ toán học giữa khóa chính và các khóa phụ.

Lịch trình khóa thường được tính trước, trong một sự cân bằng giữa bảo mật và hiệu suất, và được lưu trong bộ nhớ trong khi mã hóa/giải mã được thực hiện. Một ví dụ về khóa AES 128-bit trống (toàn bộ là số 0) và lịch trình khóa tương ứng của nó, được trích xuất từ bộ nhớ, được hiển thị trong Hình 1.

00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	Original key
62	63	63	63	62	63	63	63	62	63	63	63	62	63	63	63	Sub Keys
9b	98	98	c9	f9	fb	fb	aa	9b	98	98	c9	f9	fb	fb	aa	
90	97	34	50	69	6c	cf	fa	f2	f4	57	33	0b	0f	ac	99	
ee	06	da	7b	87	6a	15	81	75	9e	42	b2	7e	91	ee	2b	
7f	2e	2b	88	f8	44	3e	09	8d	da	7c	bb	f3	4b	92	90	
ec	61	4b	85	14	25	75	8c	99	ff	09	37	6a	b4	9b	a7	
21	75	17	87	35	50	62	0b	ac	af	6b	3c	c6	1b	f0	9b	
0e	f9	03	33	3b	a9	61	38	97	06	0a	04	51	1d	fa	9f	
b1	d4	d8	e2	8a	7d	b9	da	1d	7b	b3	de	4c	66	49	41	
b4	ef	5b	cb	3e	92	e2	11	23	e9	51	cf	6f	8f	18	8e	

Hình 1: AES Key và Key Schedule

Đáng chú ý là lịch trình khóa cho một khóa AES 128-bit được biểu diễn dưới dạng một mảng phẳng các byte trong bộ nhớ, với 16 byte đầu tiên (hoặc 128 bit) là khóa ban đầu. 160 byte còn lại là các khóa vòng được dẫn xuất từ khóa này. Đối với các khóa AES lớn hơn, lịch trình khóa tương ứng cũng sẽ lớn hơn.

Nhiều công cụ đã được phát triển để sử dụng hiện tượng này nhằm xác định các khóa AES trong bộ nhớ như **AESFinder**, **Volatools**, **interrogate**, và **Findaes**. Tuy nhiên, chưa có nghiên cứu nào cho thấy các công cụ này từng được sử dụng để phân tích ransomware mã hóa cụ thể.



### 3. Thiết kế

#### 3.1. Thiết kế môi trường thử nghiệm ransomware

Để tiến hành các thí nghiệm ransomware một cách hợp lệ và thực tế, môi trường thử nghiệm phải mô phỏng máy tính thật, loại bỏ thông tin nhạy cảm và lý tưởng là tách biệt khỏi internet. Dựa trên nghiên cứu của Bose, phần mềm ảo hóa Oracle VirtualBox được chọn để triển khai môi trường thử nghiệm này.

Môi trường thử nghiệm được thiết kế với ba máy ảo, trong đó hai máy là "máy nạn nhân" (chỉ một máy chạy ransomware tại một thời điểm). Để đảm bảo khả năng quan sát đầy đủ hành vi của ransomware, một máy ảo thứ ba được cấu hình để cung cấp dịch vụ mạng (như DNS, IRC, HTTP) cho máy nạn nhân, mô phỏng máy chủ điều khiển và kiểm soát (C&C).

Các máy này kết nối với nhau qua mạng "host-only" và hoàn toàn cách ly khỏi internet. Máy chủ vật lý là một laptop được ngắt kết nối (air-gapped) để tăng cường mức độ an toàn và cách ly. Cấu hình này giúp ransomware "nhìn nhận" máy nạn nhân như một máy tính thật, khuyến khích nó hoạt động bình thường. Đồng thời, tường lửa và phần mềm chống virus cũng được triển khai để bảo vệ máy chủ vật lý.

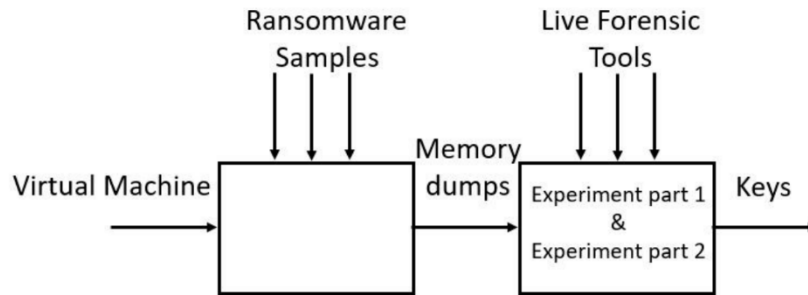
#### 3.2. Thiết kế thí nghiệm

Ở mức trừu tượng nhất, các thí nghiệm được thiết kế có thể được xem như sau. Một mẫu ransomware được thực thi trong môi trường ảo. Trong quá trình thực thi này, các bản sao của bộ nhớ dễ bay hơi của máy được lấy. Các công cụ pháp y sau đó được sử dụng để phân tích các tệp bộ nhớ đã thu thập nhằm cố gắng tìm ra khóa mã hóa. Các khóa tìm thấy sau đó được sử dụng để giải mã các tệp đã bị mã hóa trong quá trình thực thi ransomware để xác nhận rằng khóa mã hóa đối xứng chính xác đã được xác định.

Cuộc điều tra này có thể được chia thành ba thí nghiệm phụ riêng biệt. Kết quả của chúng, khi kết hợp, được sử dụng để xác nhận hoặc bác bỏ giả thuyết rằng các kỹ thuật pháp y theo thời gian thực có thể được sử dụng để giảm thiểu một cuộc tấn công ransomware. Mỗi vòng thí nghiệm bắt đầu với một phiên bản mới của một máy ảo phản ánh một máy trạm thực tế đang được khởi động và một ví dụ về ransomware đang được điều tra được thực thi. Ba thí nghiệm sau đây được thực hiện.

##### 3.2.1. Phần 1 của thí nghiệm - Xác định khóa trong bộ nhớ

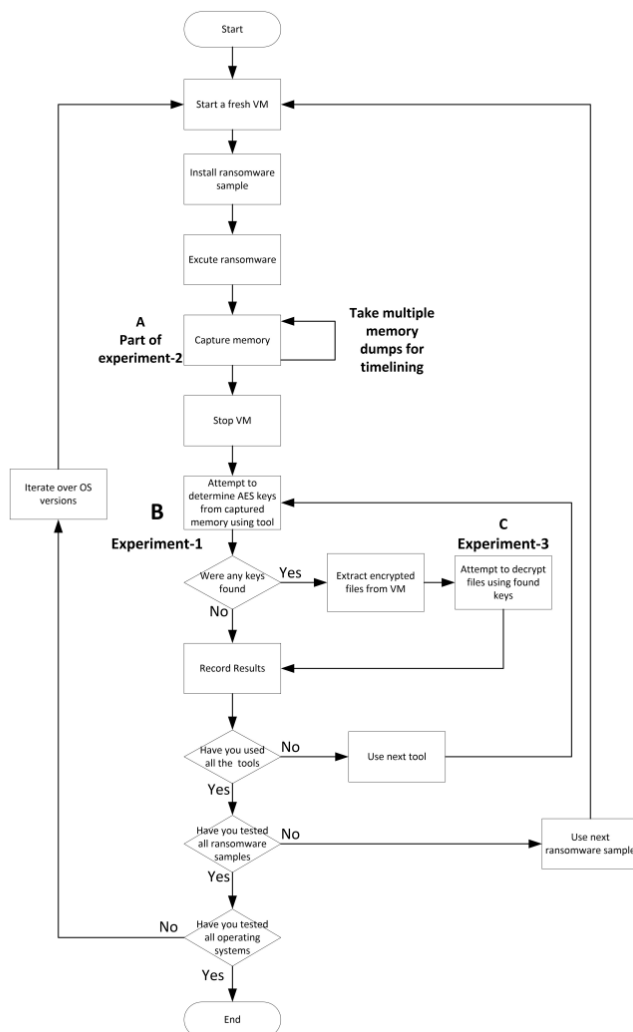
Một bản trích xuất bộ nhớ từ máy nơi ransomware đang được thực thi được thu thập. Trong quá trình lựa chọn các mẫu ransomware cho các thí nghiệm này, đã biết rằng AES đang được sử dụng cho phần mã hóa đối xứng, vì vậy sau khi trích xuất hoàn tất, bộ nhớ được phân tích bằng các công cụ pháp y theo thời gian thực để xác định xem khóa AES có thể được phát hiện hay không. Đặc biệt chú ý đến các khu vực bộ nhớ có entropy cao. Một biểu diễn đồ họa của thí nghiệm này và thí nghiệm tiếp theo được hiển thị trong Hình 2.



Hình 2. Tổng quan về thí nghiệm phần 1 và thí nghiệm phần 2

### 3.2.2. Phần 2 của thí nghiệm - Tạo dòng thời gian của khóa

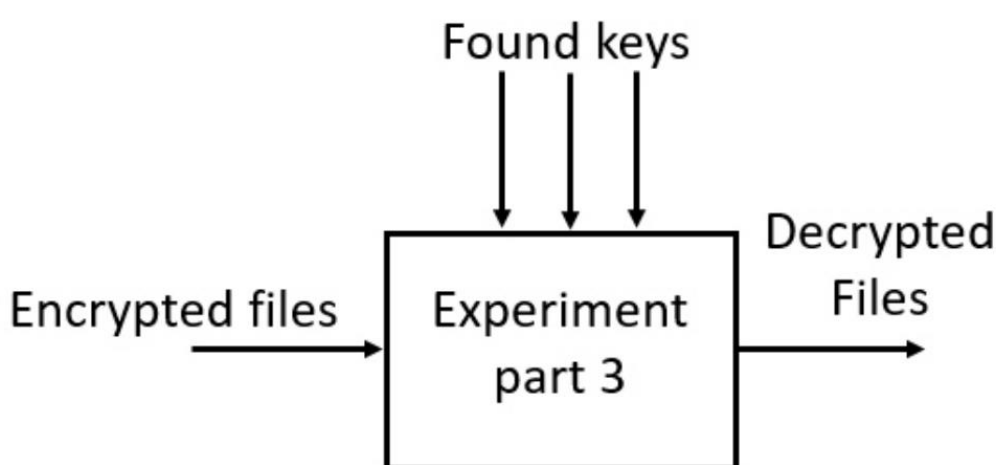
Các bản trích xuất bộ nhớ được thực hiện ở các khoảng thời gian đều đặn trong suốt chu kỳ thực thi hoàn chỉnh của ransomware để xác định tại điểm nào khóa được tải vào bộ nhớ và nó tồn tại trong bao lâu. Điều này hỗ trợ việc thực hiện phần 1 của thí nghiệm bằng cách xác định khi nào thực hiện điểm A trong Hình 4. Kết quả của thí nghiệm này là một dòng thời gian xấp xỉ cho việc thực thi ransomware.



Hình 4. Luồng thực hiện thí nghiệm

### 3.2.3. Phần 3 của thí nghiệm - Xác thực các khóa tìm thấy

Nếu bất kỳ khóa nào được phát hiện trong phần 1 của thí nghiệm, chúng được kiểm tra để xác định xem chúng có thể giải mã bất kỳ tệp kiểm soát nào đã bị ransomware mã hóa hay không, như được chỉ ra bởi các điểm B và C trong Hình 4. Một công cụ được phát triển bởi tác giả sử dụng ngôn ngữ lập trình Python được sử dụng để thực hiện việc giải mã. Một biểu diễn đồ họa của thí nghiệm này được hiển thị trong Hình 3.



Hình 3. Tổng quan về phần 3 của thí nghiệm

### 3.2.4. Thí nghiệm kết hợp

Mặc dù các thí nghiệm được thiết kế trong cuộc điều tra này có nhiều điểm tương đồng với công việc trước đây như sử dụng hệ điều hành Windows trên máy ảo và sử dụng các công cụ tương tự để trích xuất khóa, nhưng các thí nghiệm này khác biệt ở chỗ nhiều công cụ trích xuất khóa trên nhiều hệ điều hành đã được thử nghiệm và các khóa được phát hiện đã được kiểm tra để xác nhận rằng chúng có thể giải mã thành công các tệp kiểm soát. Ngoài ra, nhiều bản trích xuất bộ nhớ được thực hiện trong suốt thí nghiệm để cho phép tạo ra một dòng thời gian cho việc thực thi ransomware. Một biểu diễn đồ họa của thí nghiệm tổng thể được đưa ra trong **Hình 4** và các bước chính được thảo luận dưới đây.

- **Lập lại trên các phiên bản Hệ điều hành:** Khi thảo luận về tính thực tế trong thí nghiệm, Rossow [39] cảnh báo việc thực hiện thí nghiệm chỉ trên một hệ điều hành và sau đó rút ra kết luận chung. Để tránh sự chỉ trích tiềm ẩn này, các thí nghiệm trong nghiên cứu này được tiến hành trên hai phiên bản hệ điều hành Windows được sử dụng phổ biến nhất hiện nay.
- **Khởi động một máy ảo mới:** Kết quả chỉ có thể so sánh nếu mỗi mẫu được thực thi trong một môi trường giống hệt nhau, vì vậy một máy ảo mới được khởi động ở đầu mỗi thí nghiệm.
- **Cài đặt Ransomware:** Mẫu cần thử nghiệm được giải nén từ kho lưu trữ và chuẩn bị để thực thi.
- **Thực thi Ransomware:** Mẫu ransomware được chọn được thực thi từ dòng lệnh.

- **Thu thập Bộ nhớ:** Thời gian để lấy bản sao bộ nhớ hoạt động của máy ảo được xác định bởi kết quả của phần 2 của thí nghiệm. Nếu các khóa có sẵn trong bộ nhớ, thì một bản sao của bộ nhớ máy được lấy, sử dụng các công cụ do phần mềm ảo hóa cung cấp, theo các kỹ thuật tương tự được sử dụng bởi các nhà nghiên cứu khác.
- **Dừng Máy ảo:** Sau khi số lượng cần thiết các bản trích xuất bộ nhớ đã được thực hiện, hoặc nếu ransomware đã hoàn thành, thì máy ảo nạn nhân được dừng lại.
- **Cố gắng xác định các khóa AES từ bộ nhớ đã thu thập:** Phân tích được thực hiện trên các mẫu bộ nhớ đã thu thập với mục tiêu xác định các khóa AES tiềm năng, sử dụng ba công cụ được thảo luận dưới đây:
  1. **findaes** - Một công cụ được phát triển bởi Kornblum dựa trên công trình của Trenholme, cố gắng tìm các khóa sử dụng cấu trúc lịch trình khóa AES.
  2. **interrogate** - Một công cụ được phát triển bởi Maartmann-Moe cũng dựa trên công trình của Trenholme.
  3. **RansomAES** - Một công cụ kết hợp được phát triển bởi tác giả, tích hợp logic từ Volatility Framework cùng với logic từ findaes nhằm cải thiện độ chính xác và hiệu suất của việc phát hiện khóa.
- **Trích xuất các tệp đã mã hóa:** Một tập hợp các tệp kiểm soát thường bị nhắm đến với nội dung đã biết được đặt trên máy nạn nhân trước khi thực thi ransomware. Sau khi ransomware đã thực thi, các tệp này được phân tích để xác định xem chúng đã bị mã hóa hay chưa.
- **Cố gắng giải mã các tệp:** Nếu bất kỳ khóa tiềm năng nào được phát hiện trong giai đoạn phân tích của thí nghiệm, chúng được sử dụng để cố gắng giải mã các tệp kiểm soát đã bị mã hóa được trích xuất từ máy ảo nạn nhân. Vector Khởi tạo AES (IV) cần thiết cho việc giải mã được chứa trong tệp đã mã hóa và được sử dụng kết hợp với các khóa tiềm năng để giải mã tệp.

## 4. Thực hiện và kết quả

### 4.1. Lựa chọn mẫu ransomware

Các cuộc tấn công ransomware gần đây đã được nghiên cứu [11, 18, 23, 34, 51], và dựa trên các phát hiện, quyết định chọn ba mẫu ransomware gần đây nhất để phân tích. Tất cả đều sử dụng AES cho phần mã hóa đối xứng.

**NotPetya:** Cuộc tấn công ransomware này được coi là cuộc tấn công gây thiệt hại lớn nhất từng có.

**Bad Rabbit:** Một phiên bản của họ ransomware NotPetya xuất hiện vào năm 2018.

**Phobos:** Một trong những mẫu ransomware gần đây nhất được tìm thấy, với thông tin chi tiết có sẵn, và cũng là một trong những mẫu phổ biến nhất trong Quý 4 năm 2019.

**Annabelle:** Một ransomware được làm ra để chủ nhân thể hiện kỹ năng của bản thân sử dụng key tĩnh, từng được 1 số nhóm tấn công như nhóm Key Group, vv sử dụng.

Các chi tiết cụ thể về các mẫu ransomware được xác minh bởi VirusTotal ([www.virustotal.com](http://www.virustotal.com)) được mô tả trong Bảng 1. Ba mẫu ransomware được chọn tương

tự nhau ở chỗ chúng đều sử dụng mã hóa đối xứng AES và cũng sử dụng cùng một khóa cho tất cả các tệp tin được mã hóa.

**Table 1. Ransomware Samples**

Name	SHA256 Checksum
NotPetya	027cc450ef5f8c5f653329641ec1fed91f694e0d229928963b30f6b0d7d3a745
Bad Rabbit	630325cac09ac3fab908f903e3b00d0dadd5fdaa0875ed8496fcbb97a558d0da
Phobos	a91491f45b851a07f91ba5a200967921bf796d38677786de51a4a8fe5ddeafd2

Ngoài ra nhóm còn chạy thử thêm 1 ransomware là Annabelle với thông tin chi tiết như sau:

**Annabelle Ransomware Hash**

```
MD5: 0f7a3287-991154b1c728c7c7a0aef72
SHA1: 0260579a73095455fcbaddfa7e98a2bb28f6b
SHA256: 716395ba5c31e7186c46795b19019ac2b4655ad8f1c1c0e12139ba67fa9e1a1
SSDEEP: 993216:18kmoq8r+udX+L3Mg02meh4aaR22Ch+CMk+ly:(Mk+X+1b)g27uR0W9u)
```

#### 4.1.1. Thông tin chi tiết về các mẫu ransomware

##### 4.1.1.a. NotPetya

NotPetya là một loại mã độc tống tiền (ransomware) xuất hiện vào tháng 6 năm 2017. Ban đầu được xem như một biến thể của mã độc Petya, NotPetya nhanh chóng cho thấy mức độ phá hoại rộng lớn với cơ chế lây lan mạnh mẽ hơn. NotPetya không chỉ nhằm mục đích đòi tiền chuộc mà còn gây gián đoạn hoạt động của các tổ chức lớn trên toàn cầu.

Cách hoạt động của NotPetya

Các chiến dịch của NotPetya sử dụng nhiều phương thức phân phối đặc trưng:

- Khai thác lỗ hổng SMBv1 EternalBlue và EternalRomance: NotPetya lây lan qua mạng bằng cách khai thác lỗ hổng SMBv1 EternalBlue và EternalRomance, những lỗ hổng từng được sử dụng trong các cuộc tấn công khác.
- Lạm dụng phần mềm kế toán bị nhiễm mã độc: Mã độc đã được phát tán thông qua bản cập nhật của phần mềm kế toán MeDoc ở Ukraine, cho phép NotPetya tấn công một số lượng lớn các máy tính trong một thời gian ngắn.

Khi được thực thi, NotPetya tiến hành các bước sau:

- Truy xuất quyền admin và lây lan nội bộ: Sử dụng các công cụ như Mimikatz để thu thập thông tin đăng nhập và tự phát tán qua các thiết bị khác trong mạng.
- Ghi đè Master Boot Record (MBR): Sau khi lây nhiễm, NotPetya ghi đè MBR, làm hệ thống không thể khởi động bình thường, đồng thời tạo thông báo đòi tiền chuộc.

- Mã hóa tệp: NotPetya sử dụng mã hóa AES-128 và RSA-2048 để mã hóa các tệp tài liệu trên hệ thống, khiến nạn nhân không thể truy cập dữ liệu.

Dấu hiệu của một cuộc tấn công NotPetya

- Tên miền C2: Sử dụng một tên miền máy chủ điều khiển (C2) để nhận lệnh và gửi thông tin.

- Ghi đè MBR và màn hình đòi tiền chuộc: Hệ thống khởi động với thông báo đòi tiền chuộc, và MBR đã bị chỉnh sửa.

- Phát tán tự động qua SMB: NotPetya tự phát tán qua các hệ thống mạng nội bộ bằng cách khai thác SMBv1 và các thông tin đăng nhập đã thu thập.

Thông báo đòi tiền chuộc của NotPetya tương tự với Petya, tuy nhiên, NotPetya không chứa khóa giải mã, biến nó thành một mã độc phá hoại (wiper) thay vì chỉ là ransomware.

Các quy tắc phát hiện NotPetya

NotPetya có thể được phát hiện bằng cách sử dụng các quy tắc Yara trên các điểm cuối và trong mạng để xác định các hoạt động liên quan.

#### 4.1.1.b. Bad Rabbit

Bad Rabbit là một loại mã độc tống tiền (ransomware) xuất hiện vào năm 2017 và ban đầu được xem như một "hậu duệ" tiềm năng của Petya/NotPetya. Tuy nhiên, dù chia sẻ một số kỹ thuật hoạt động giống với Petya/NotPetya, Bad Rabbit có mã nguồn riêng biệt và rất có khả năng do một nhà phát triển khác tạo ra. Loại mã độc này lấy tên từ thông báo đòi tiền chuộc của nó và trang web .onion trên nền tảng Tor.

#### Cách hoạt động của Bad Rabbit

Các chiến dịch của Bad Rabbit sử dụng những chiến thuật phân phối đặc trưng.

Một trang web bị lây nhiễm sẽ sử dụng kỹ thuật "watering hole" (bẫy cài đặt) để lừa người truy cập tải về bản cập nhật giả mạo của Adobe Flash Player. Trình cài đặt, thường được đặt tên là **install\_flash\_player.exe**, sử dụng chứng chỉ ký giả mạo với tên "Symantec Corporation".

Khi được thực thi, mã độc tải xuống sẽ chạy giai đoạn đầu tiên của Bad Rabbit, bao gồm việc nhập và cài đặt nhiều tệp tin bổ sung, sau đó thực thi phần mã độc chính.

Khi đã được cài đặt, Bad Rabbit thực hiện các bước sau:

1. Tìm kiếm các quy trình đang chạy bằng cách sử dụng hàm băm (hash) của tên quy trình và vô hiệu hóa một danh sách xác định trước các phần mềm bảo mật CNTT.
2. Cài đặt một bản sao của DiskCryptor (một công cụ mã hóa hợp pháp) vào một tệp có tên **cscc.dat** để sau đó mã hóa các tệp của nạn nhân.
3. Nhập phần mã độc chính có tên **infpub.dat**.
4. Tạo một tệp có tên **dispci.exe**, lên lịch một tác vụ để thực thi phần mã độc chính vào lần khởi động tiếp theo, sau đó khởi động lại máy.

Sau khi khởi động lại, mã độc chính sẽ tiến hành mã hóa các tệp.

5. Cố gắng tìm thông tin đăng nhập SMB được lưu trữ trên hệ thống của nạn nhân và tự phát tán sang các máy khác trong mạng bằng cách sử dụng công cụ Mimikatz với cả thông tin đăng nhập bị đánh cắp và các thông tin đăng nhập SMB phổ biến đã được mã hóa sẵn, cũng như khai thác lỗ hổng SMBv1 EternalRomance.
6. Ghi đè lên Master Boot Record (MBR) của hệ thống bị nhiễm và cài đặt bộ nạp khởi động (bootloader) của riêng nó (tương tự mã độc Petya), đồng thời sao chép một nhân độc hại vào cuối ổ cứng của nạn nhân để khởi động.
7. Khởi động lại hệ thống bị nhiễm, làm cho MBR đã chỉnh sửa khởi động nhân Bad Rabbit, hiển thị thông báo đòi tiền chuộc trên màn hình.

Bad Rabbit nhắm vào các tệp có đuôi thông dụng cho tài liệu, tệp nén và tệp sao lưu, máy ảo, đĩa cứng ảo, hình ảnh và các tệp ngôn ngữ kịch bản khác. Bad Rabbit sử dụng mã hóa AES-128 để mã hóa các tệp và sau đó mã hóa khóa AES bằng khóa công khai RSA 2048-bit đã mã hóa sẵn.

### Dấu hiệu của một cuộc tấn công Bad Rabbit

Trong khi hầu hết các loại ransomware thêm một mã định danh riêng biệt vào cuối mỗi tệp đã mã hóa, Bad Rabbit chỉ thêm từ **encrypted** vào các tệp bị mã hóa. Mã độc này sử dụng tên miền **1dnscontrol.com** làm máy chủ điều khiển chính (C2), nhưng tên miền này có thể được dẫn qua các trang web bị nhiễm khác.

Thông báo đòi tiền chuộc của Bad Rabbit trông giống với Petya/NotPetya, nhưng có thêm bộ đếm thời gian nhằm tăng tính cấp bách cho chiến dịch tống tiền.

Bad Rabbit cũng tạo ra các tác vụ theo lịch trình với tên tham chiếu đến *Game of Thrones*, ví dụ như **rhaegal**, **dronos**, **viserion**, và **drogon**, và tạo ra các tệp có tên **infpub.dat**, **dispci.exe**, và **cscd.dat** trong thư mục **C:\Windows\**.

Các quy tắc Yara cũng đã được công khai để nhận diện hoạt động liên quan đến Bad Rabbit trên các điểm cuối và trong mạng.

#### 4.1.1.c. Phobos

Ransomware Phobos xuất hiện vào đầu năm 2019 và đã được ghi nhận là có mối quan hệ mật thiết với một loại ransomware đã biết trước đó là Dharma. Phobos là một trong những ransomware được phát tán thông qua các kết nối Remote Desktop (RDP). Nếu một kẻ tấn công lấy được thông tin đăng nhập của RDP, họ có thể dễ dàng truy cập vào hệ thống từ xa để cài đặt ransomware.

#### Cách hoạt động của phobos:

1. Tiến hành thu thập thông tin xác thực để có thể mở rộng phạm vi hoặc quyền truy cập của mình.
2. Tìm kiếm các tệp để mã hóa và ban đầu sử dụng một khóa AES duy nhất để mã hóa chúng. Tên tệp cũng được thay đổi trong quá trình này.

3. Sau khoảng hai phút sau, một thông báo đòi tiền chuộc được hiển thị, yêu cầu thanh toán để đổi lại việc giải mã.

4. Máy không sau khi bị nhiễm mã độc sẽ không tự động khởi động lại. Nếu người dùng khởi động lại máy, thì cùng một thông báo ransomware sẽ được hiển thị lại sau khi người dùng khởi động.

Dấu hiệu của một cuộc tấn công phobos:

1. Kết nối RDP bị xâm nhập: Phobos thường lây lan qua các kết nối Remote Desktop Protocol (RDP) bị xâm nhập. Nếu máy tính hoặc hệ thống có RDP mở mà không có các biện pháp bảo vệ như mật khẩu mạnh hoặc xác thực hai yếu tố, đây có thể là một dấu hiệu bị tấn công. Kẻ tấn công có thể dùng thông tin đăng nhập RDP bị đánh cắp để truy cập từ xa vào hệ thống.
2. Xuất hiện các tệp có phần mở rộng lạ: Sau khi mã hóa các tệp trên máy, Phobos thường thay đổi tên tệp và thêm phần mở rộng mới vào cuối mỗi tên tệp như .phobos
3. Thông báo đòi tiền chuộc của Phobos như với Petya/NotPetya hay Bad Rabbit.
4. Khác với các ransomware khác phobos sẽ không buộc máy phải tự khởi động lại nhưng nếu người dùng chủ động khởi động lại thì sau khi khởi động sẽ có thông báo mã hóa hiện lên.

#### **4.1.1.d. Annabelle**

Tháng 03/2018, lại có thêm một mã độc mới được phát hiện. Mã độc tống tiền này có tên gọi Annabelle – được phát hiện bởi Bart – một nhà nghiên cứu bảo mật. Trong khi hầu hết mã độc tống tiền được tạo ra để kiếm tiền từ nạn nhân, tác giả của mã độc này lại tạo ra chúng để chứng tỏ kỹ năng của mình.

##### **Cách thức hoạt động:**

Annabelle có khả năng can thiệp vào hệ thống bằng cách sửa đổi các tệp tin và vô hiệu hóa nhiều chương trình phổ biến như Notepad, Notepad++, Internet Explorer, Chrome, và Bcdedit. Sau khi xâm nhập, mã độc cố gắng tự phát tán thông qua một tệp autorun.inf. Tuy nhiên, cách lây lan này không còn hiệu quả trên các phiên bản Windows 10 cập nhật mới nhất, do hệ điều hành không còn hỗ trợ loại tệp này. Dù vậy, người dùng vẫn cần cảnh giác vì điều này không đảm bảo an toàn tuyệt đối cho hệ thống.

Sau khi thực hiện các thao tác trên, Annabelle tiến hành mã hóa dữ liệu trên máy tính bằng một khóa mã hóa cố định và thêm đuôi .ANNABELLE vào cuối tên mỗi tệp tin. Tiếp theo, mã độc tự động khởi động lại hệ thống. Khi người dùng đăng nhập lại, màn hình khóa sẽ hiển thị tên tác giả là iCoreX0812 kèm thông tin liên hệ qua Discord. Để tăng mức độ hiện diện, Annabelle còn thay thế Master Boot Record, khiến thông tin về tác giả luôn xuất hiện trên màn hình mỗi lần máy tính khởi động.

May mắn là ANNABELLE được phát triển để sử dụng khóa mã hóa cứng. Nói cách khác, nó sử dụng cùng một khóa để mã hóa tất cả các tệp của nạn nhân. Do đó, không cần phải trả bất kỳ khoản tiền chuộc nào. Nhà nghiên cứu bảo mật phần mềm độc



hại Michael Gillespie đã phát hành một công cụ giải mã có khả năng khôi phục các tệp được mã hóa bởi ANNABELLE miễn phí.

#### 4.1.2. Các ransomware khác

Các mẫu ransomware sau đây đã được xem xét ban đầu nhưng sau đó bị loại bỏ.

- **WannaCry:** Việc thử nghiệm xác nhận những gì đã được mô tả trong tài liệu rằng chủng loại này sử dụng các khóa AES riêng biệt cho mỗi tệp được mã hóa. Sau khi thực hiện nhiều lần kiểm tra bộ nhớ thu được trong quá trình thực thi phần mềm độc hại này bằng tất cả các công cụ pháp lý trực tiếp, không tìm thấy khóa AES nào có thể khôi phục được.
- **Cerber:** Ransomware này dường như không sử dụng mã hóa AES.
- **Lucky/nmare:** Mẫu ransomware này yêu cầu truy cập internet để có thể tải xuống các mô-đun mã hóa tệp vì chúng không được cung cấp kèm với mẫu ban đầu. Các dịch vụ được cung cấp bởi máy ảo Debian không thể đánh lừa ransomware này để nó thực thi bình thường và việc cho phép truy cập mạng bên ngoài được coi là quá rủi ro.
- **Satan, SamSam và GrandCab:** Không thể kích hoạt các mẫu ransomware này để mã hóa bất kỳ tệp nào hoặc hiển thị thông báo đòi tiền chuộc.

#### 4.2. Cấu hình máy thử nghiệm

Máy tính xách tay được sử dụng để thử nghiệm không có kết nối mạng bên ngoài và như một biện pháp phòng ngừa bổ sung, máy đã được đặt ở chế độ máy bay và thẻ wifi đã bị tắt. Khi thảo luận về các môi trường ảo, máy vật lý này được gọi là máy chủ, vì nó lưu trữ môi trường ảo trong VirtualBox.

Ba máy khách đã được định nghĩa trong môi trường ảo chạy trên máy tính xách tay. Hai máy nạn nhân với các hệ điều hành khác nhau trên mỗi máy, được sử dụng để kiểm tra hành vi của mã ransomware, và một máy dịch vụ mạng được sử dụng để cung cấp bất kỳ dịch vụ mạng nào. Chi tiết về các máy khách ảo được đưa ra trong Bảng 2.

Các máy khách được kết nối với nhau bằng cách sử dụng kỹ thuật kết nối "chỉ có máy chủ" (host-only) được khuyến nghị, tạo ra một mạng LAN ảo riêng biệt cung cấp sự cách ly và chứa đựng các máy khách. Mạng LAN ảo và các máy khách ảo được chạy trong hai cấu hình riêng biệt tùy thuộc vào phiên bản Windows nào đang được thử nghiệm. Các công cụ sau đây đã được sử dụng trong quá trình cấu hình máy Debian: **fakedns.py** – được sử dụng để cung cấp dịch vụ DNS cho mạng, và **INetSim** – Được coi là công cụ miễn phí tốt nhất [45] để cung cấp giả lập dịch vụ mạng tạo ra ảo giác về một mạng giả lập thực tế mà phần mềm độc hại có thể tương tác nếu cần thiết.

**Table 2.** Virtual Hardware Configurations

Machine Name	Operating System	Purpose
Windows 7	Windows 7 Ultimate Build 7600	Ransomware Victim Machine
Windows 10	Windows 10 Pro Build 10586.494	Ransomware Victim Machine
Debian	Debian 5.2.9	Network Services

### 4.3. Các thử nghiệm

#### 4.3.1. Thử nghiệm phần 1 - Xác định khóa trong bộ nhớ

Một máy ảo mới đã được khởi động và một bản sao của bộ nhớ máy đã được lấy trước khi thực thi phần mềm ransomware, do đó, bất kỳ khóa AES nào có trong bộ nhớ của máy trước khi thực thi phần mềm ransomware đều có thể được xác định và loại trừ khỏi kết quả thí nghiệm. Để hỗ trợ khả năng tái tạo thử nghiệm, các lệnh dùng để khởi chạy từng mẫu ransomware được cung cấp bên dưới:

Đối với NotPetya

**c:\windows\system32\rundll32.exe**

**c:\ransomware\netpetya.dll, #1 30**

Đối với BadRabbit

**c:\ransomware\sample.badrabbit1.bin.exe**

Đối với Phobos

**c:\ransomware\1saas.bin.exe**

Đối với Annabelle

**c:\ransomware\Annabelle.exe**

Sau khi chờ đợi một thời gian ngắn, một bản sao bộ nhớ máy của khách đã được lấy. Khoảng thời gian chờ thay đổi từ mười giây đến hai phút tùy thuộc vào chủng ransomware. Thời gian cần thiết để chờ đợi được xác định bằng cách thử và sai. Để chiếm bộ nhớ, lệnh sau được thực thi trên máy chủ:

**VBoxManage.exe debugvm <VBox Machine Name>**

**dumpvmcore --filename <filename>.elf**

Trường hợp sử dụng vmware việc trích xuất bộ nhớ dễ dàng hơn bằng cách:

**Snapshot => <Virtual machine Folder> => <Window Folder>  
=><filename>.vmem**

Sau đó, tệp kết xuất bộ nhớ được phân tích bằng từng công cụ bộ nhớ điều tra trực tiếp đã chọn. Một lần nữa để hỗ trợ khả năng tái tạo, các lệnh được sử dụng được đưa ra dưới đây:

**findaes <filename> .elf (hoặc findaes <filename> .vmem)**

**interrogate -a aes -k 128 <filename>.elf (hoặc interrogate -a aes -k 128 <filename>.vmem)**

**ransomaes -p <ransomeware pid> -t Win7SP0x86 <filename>.elf (hoặc**

**ransomaes -p <ransomeware pid> -t Win7SP0x86 <filename>.vmem)**

Bất kỳ khóa nào được tìm thấy từ việc thực thi các công cụ này đều được ghi lại và sử dụng làm đầu vào cho thử nghiệm 3. Nếu không tìm thấy khóa nào thì thử nghiệm sẽ được kéo dài trong khoảng thời gian một phút và thực hiện các kết xuất bộ nhớ mới. Thử nghiệm kết thúc khi tìm thấy khóa hoặc quá trình thực thi ransomware hoàn tất.

#### **4.3.2. Thử nghiệm phần 2 - Tạo biểu đồ thời gian cho khóa**

Nếu khóa được phát hiện trong thử nghiệm 1 thì thử nghiệm này cũng được thực hiện. Việc kết xuất bộ nhớ được thực hiện thường xuyên trong suốt khung thời gian thực thi của phần mềm chạy. Khoảng thời gian được sử dụng giữa các lần kết xuất bộ nhớ khác nhau tùy thuộc vào mẫu ransomware và được xác định thông qua quá trình thử và sai qua nhiều lần thực thi. Các kết xuất được phân tích bằng cách sử dụng một trong các công cụ đã chọn để xác nhận rằng các khóa của chúng vẫn còn tồn tại. Thời điểm có khóa đã được ghi lại và dùng thời gian cơ bản cho việc thực thi ransomware đã được tạo. Quá trình tạo biểu đồ thời gian chủ yếu là một tác vụ thủ công, kết hợp các kết quả được ghi lại từ các thử nghiệm, quan sát hành vi của hệ thống, các mô tả thu được từ việc xem xét tài liệu và sử dụng mô hình ransomware sáu bước.

#### **4.3.3. Thử nghiệm 3 - Xác thực khóa được tìm thấy**

Nếu các khóa AES đề cử được phát hiện trong thử nghiệm 1 thì chúng sẽ được sử dụng để giải mã các tệp điều khiển. Đây là một phần nhiệm vụ được tự động hóa với một số bước thủ công. Tác giả đã tạo một công cụ decrypt.py để thực hiện giải mã AES cơ bản bằng cách sử dụng đối tượng mật mã 'AES' từ thư viện python 'Crypto.Cipher'.

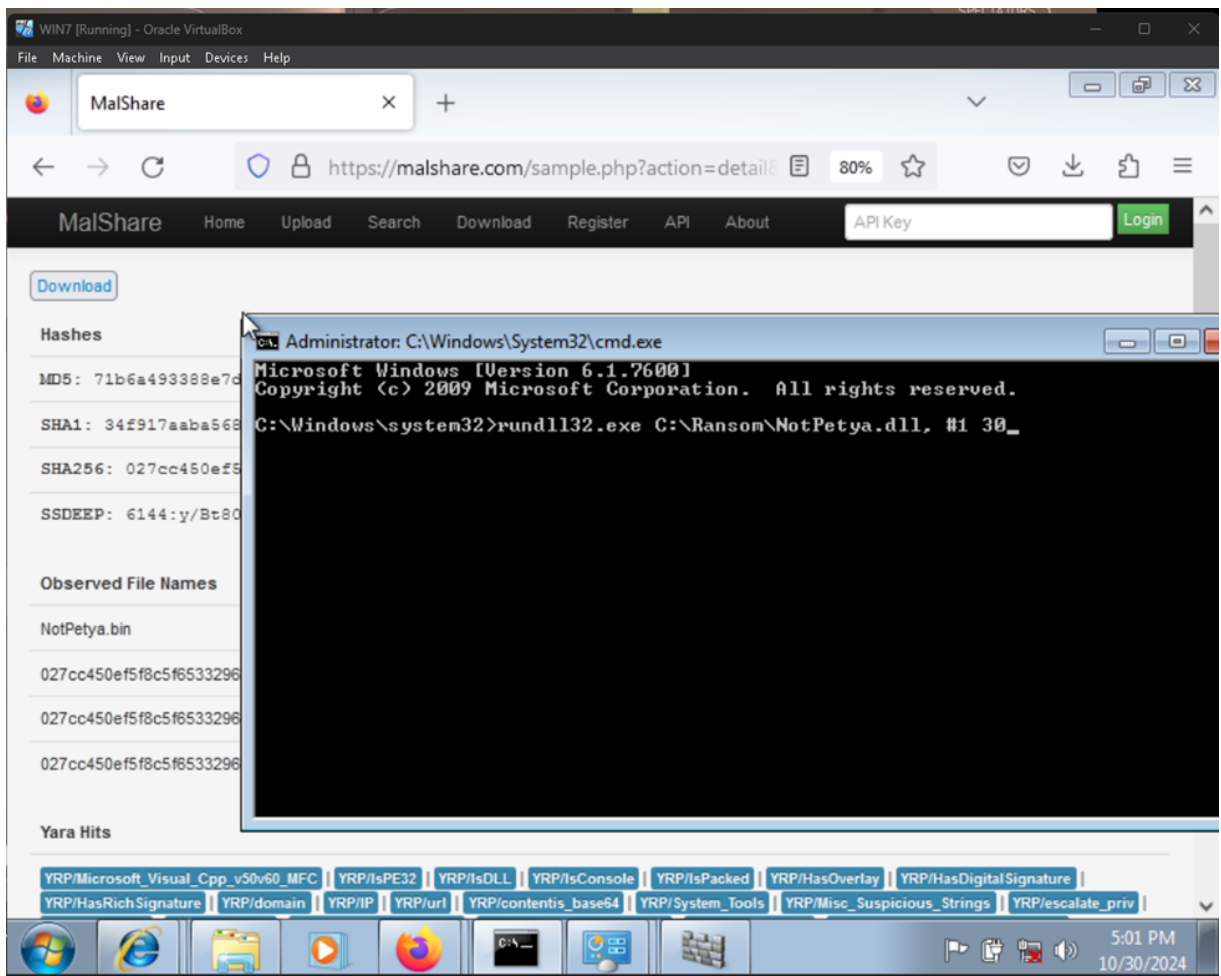
Chương trình có thể xác định IV cần thiết từ tệp được mã hóa được cung cấp và sau đó sử dụng tệp này cùng với các khóa ứng viên được phát hiện để thử và giải mã tệp. Việc chấm dứt nếu tệp được giải mã chính xác vẫn là một tác vụ thủ công. Tệp được giải mã thu được thường yêu cầu sửa đổi bổ sung như thêm tiêu đề hoặc xóa đoạn giới thiệu.

### **5. Demo và kết quả**

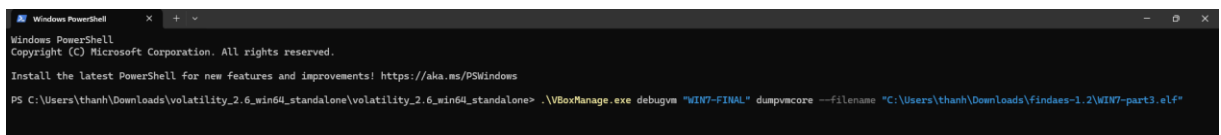
#### **5.1. Not Petya**

##### **I. Xác định khóa trong bộ nhớ**

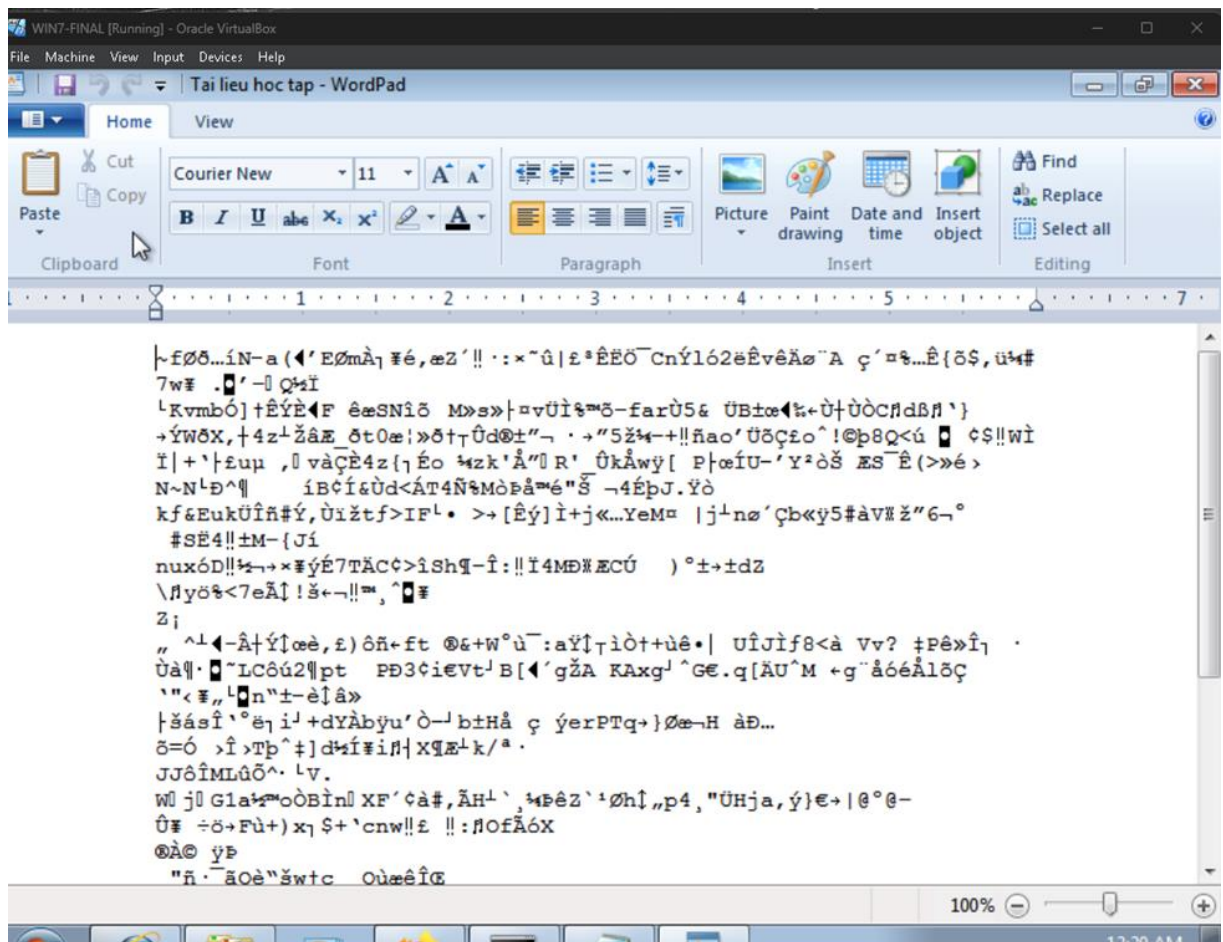
Thực thi virus Not Petya trên Windows7



Sử dụng VboxManage.exe của virtual box để dump file dưới dạng elf:

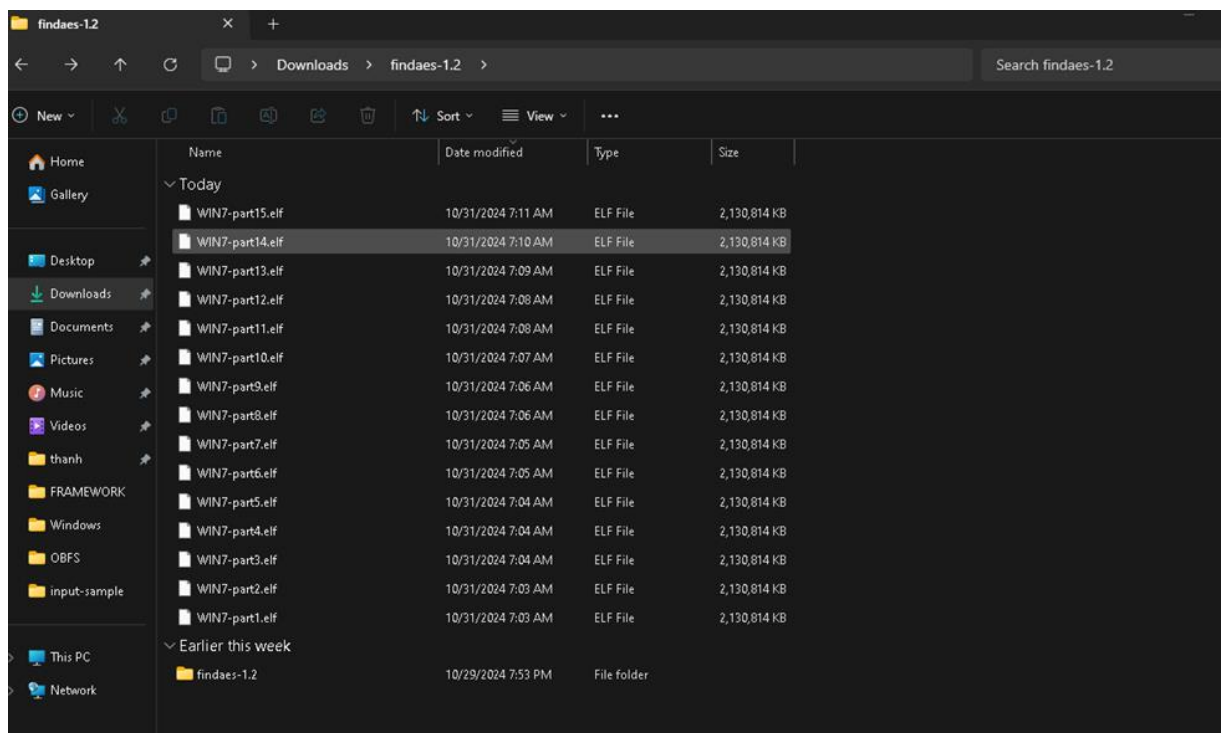


Sau khi chạy Not Petya được 2 phút:

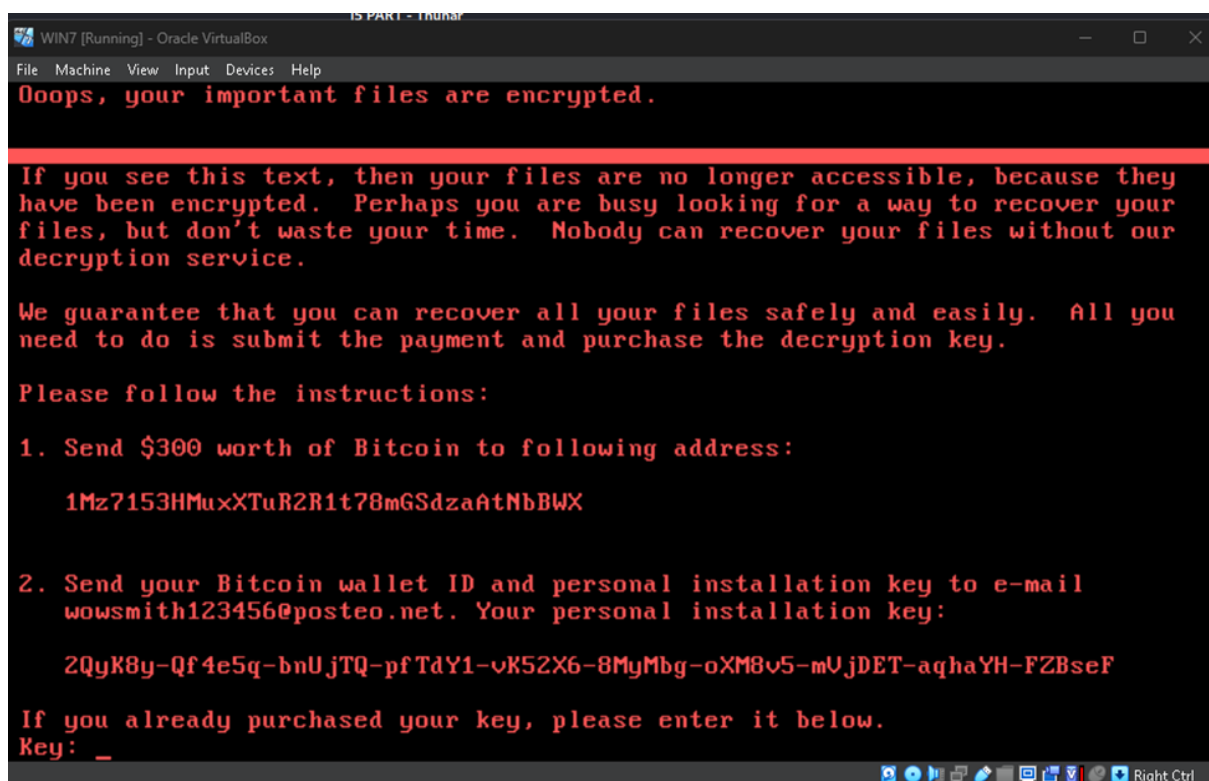


## II. Tạo dòng thời gian của khóa

Ta sẽ dump nhiều lần với mỗi thời điểm khác nhau để đảm bảo rằng key sẽ không bị bỏ sót, trong phần thí nghiệm này sẽ sử dụng tổng cộng lần lượt 15 bản dump để tiến hành tìm kiếm key.



Trong quá trình khởi động lại win, ransom Not Petya sẽ giả checkdisk nhưng nó thực chất là quá trình mã hóa MBR (chương trình khởi động máy tính), sau khi quá trình mã hóa thành công màn hình sẽ hiển thị như sau:



```
IS PART - Truong
WIN7 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Oops, your important files are encrypted.

If you see this text, then your files are no longer accessible, because they
have been encrypted. Perhaps you are busy looking for a way to recover your
files, but don't waste your time. Nobody can recover your files without our
decryption service.

We guarantee that you can recover all your files safely and easily. All you
need to do is submit the payment and purchase the decryption key.

Please follow the instructions:

1. Send $300 worth of Bitcoin to following address:

1Mz7153HMuxXTuR2R1t78mGSdzaAtNbBWx

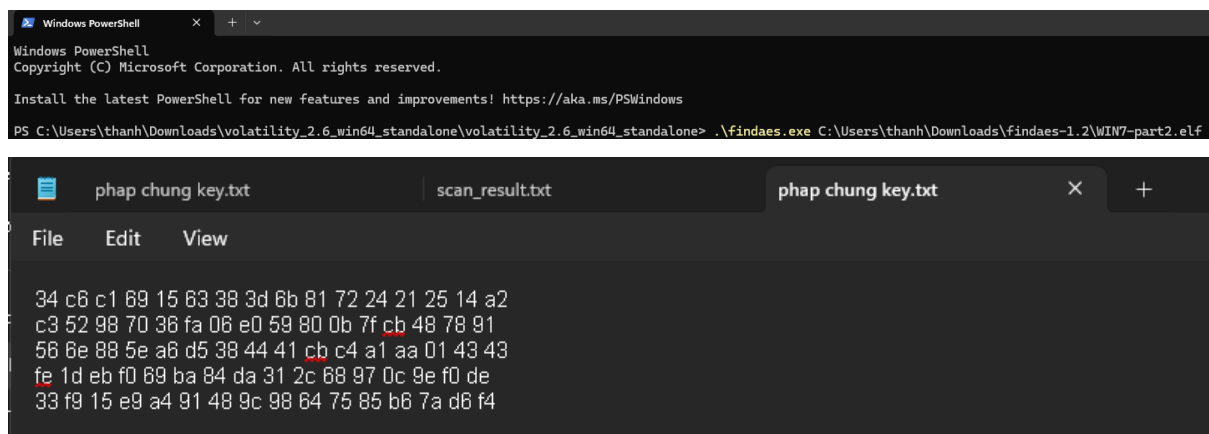
2. Send your Bitcoin wallet ID and personal installation key to e-mail
wowsmith123456@posteo.net. Your personal installation key:

2QyK8y-Qf4e5q-bnUjTQ-pfTdY1-vK52X6-8MyMbg-oXM8v5-mUjDET-aqhaYH-FZBseF

If you already purchased your key, please enter it below.
Key: _
```

### III. Xác thực các khóa tìm thấy

Sử dụng công cụ Findaes để tiến hành tìm các khóa AES có thể tồn tại trong file dump



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\thanh\Downloads\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone> .\findaes.exe C:\Users\thanh\Downloads\findaes-1.2\WIN7-part2.elf

phap chung key.txt scan_result.txt phap chung key.txt
File Edit View

34 c6 c1 69 15 63 38 3d 6b 81 72 24 21 25 14 a2
c3 52 98 70 36 fa 06 e0 59 80 0b 7f cb 48 78 91
56 6e 88 5e a6 d5 38 44 41 cb c4 a1 aa 01 43 43
fe 1d eb f0 69 ba 84 da 31 2c 68 97 0c 9e f0 de
33 f9 15 e9 a4 91 48 9c 98 64 75 85 b6 7a d6 f4
```

Thực hiện filescan tìm file bị mã hóa trước đó:



Tiến hành dump file:

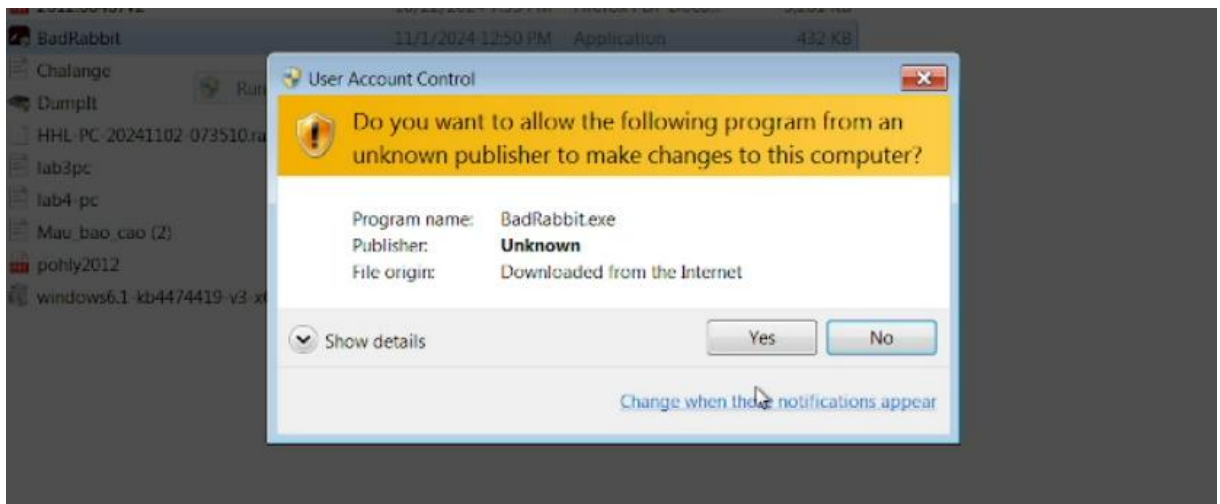
Tiến hành giải mã cho đến khi khôi phục thành công được file.

## 5.2. Bad Rabbit

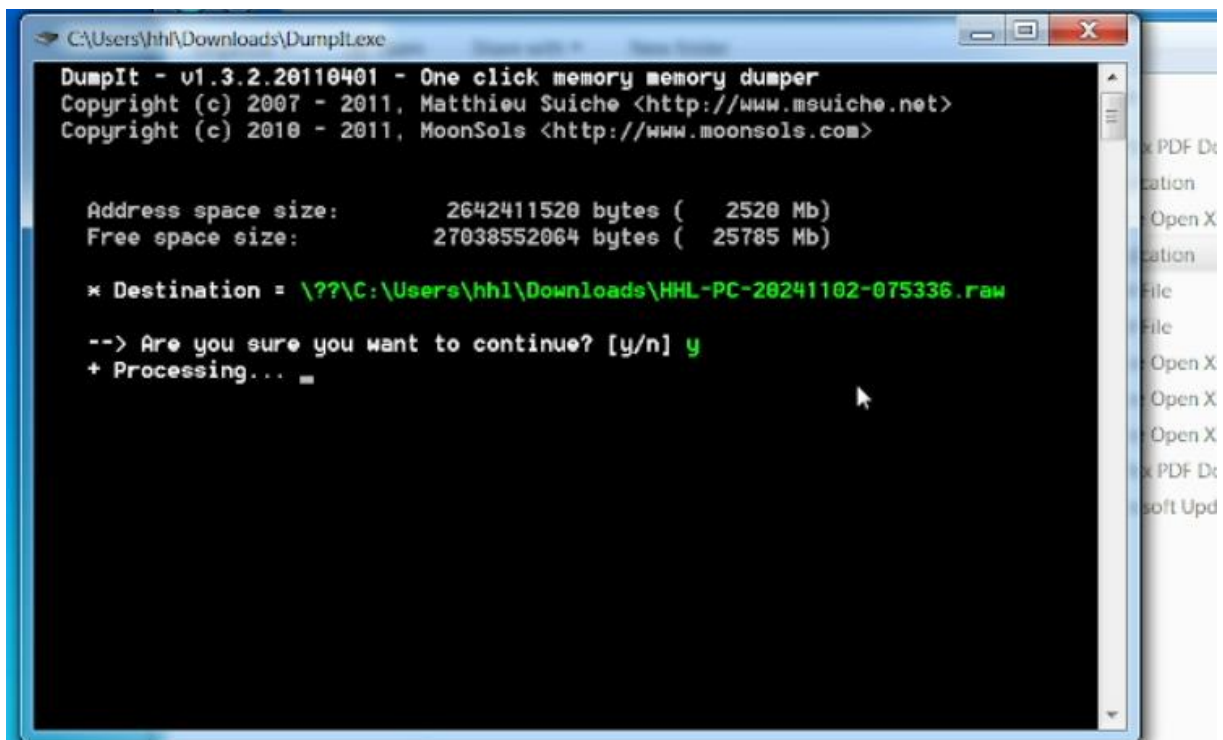
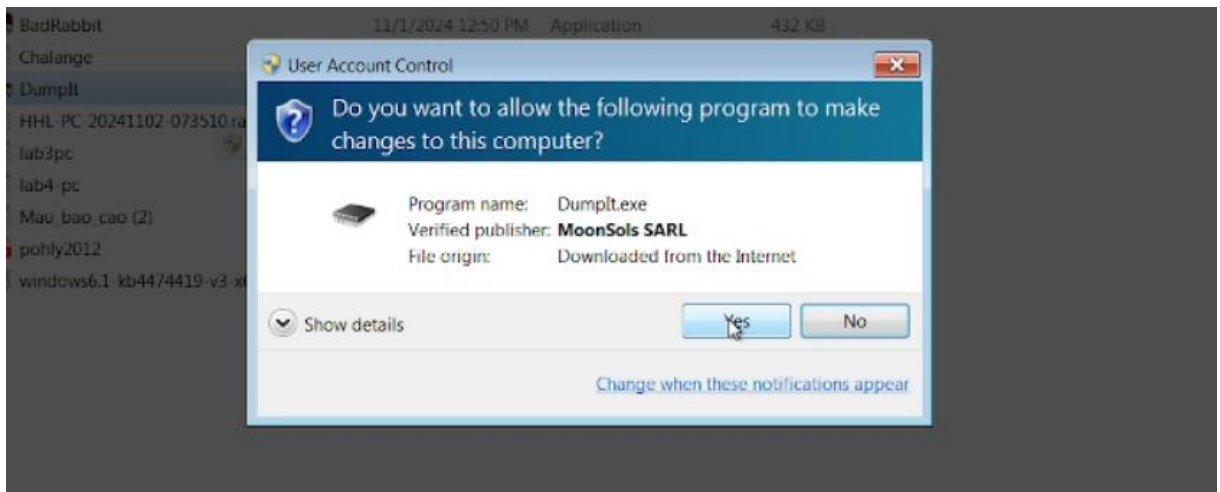
### 5.2.1. Win7

#### I. Xác định khóa trong bộ nhớ

Thực thi virus Bad Rabbit trên Windows7

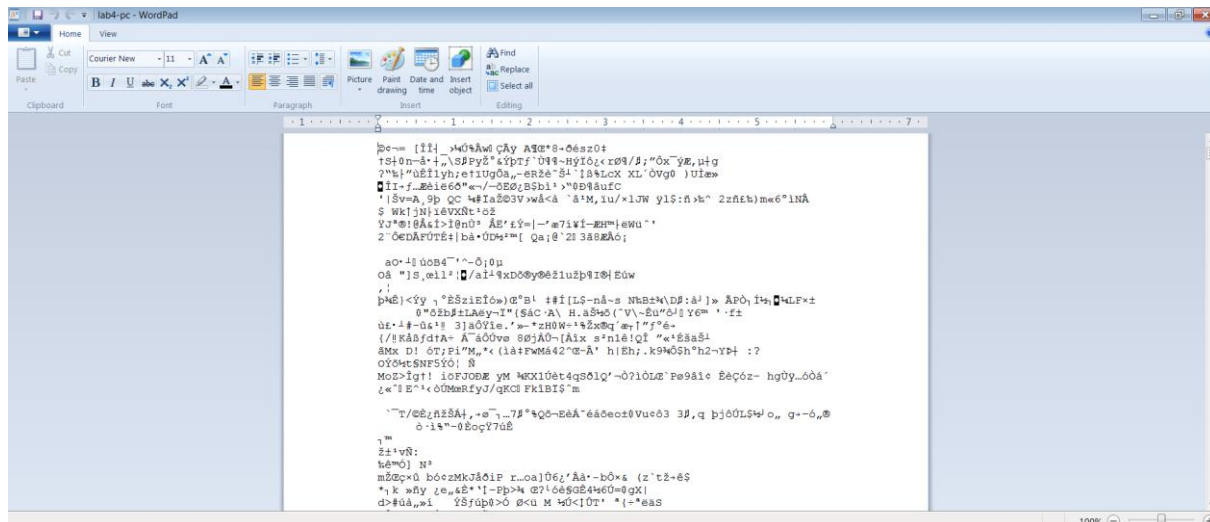


Chạy công cụ dump bộ nhớ





## Kết quả của file bị mã hóa:

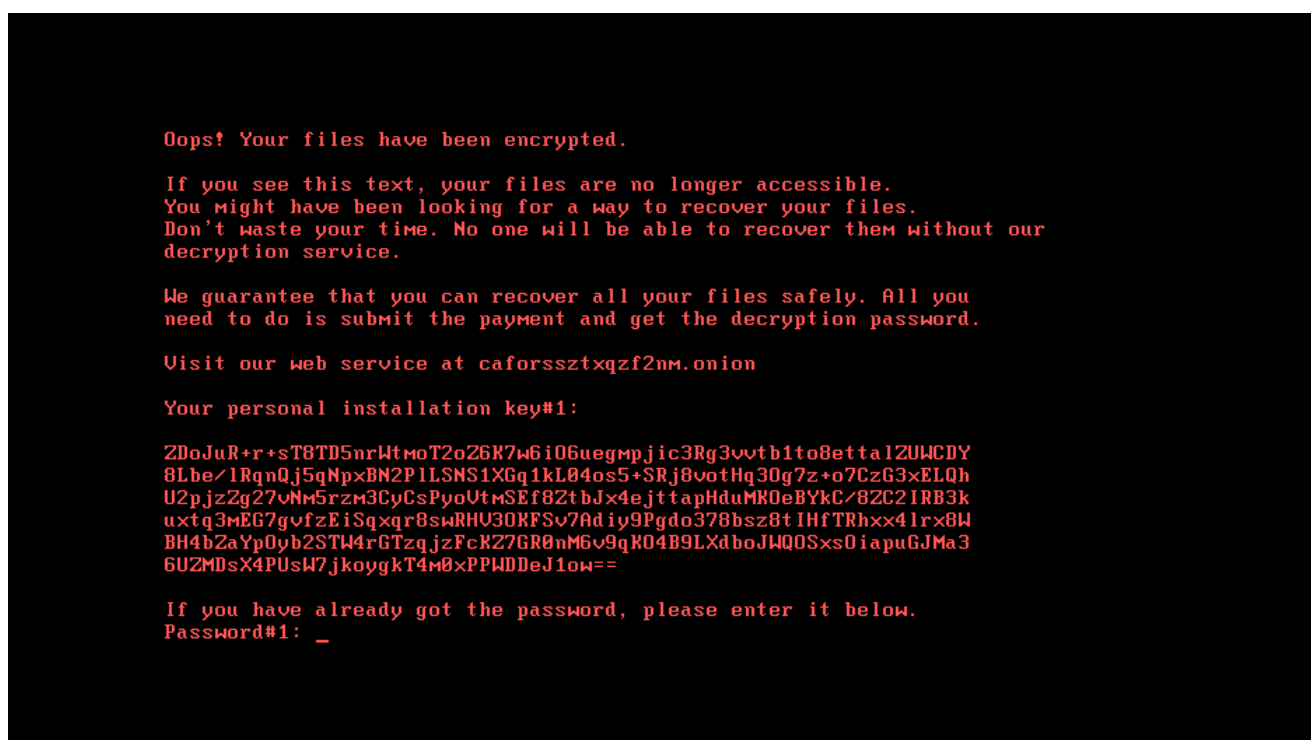


## II. Tạo dòng thời gian của khóa

Ta sẽ dump nhiều lần với mỗi thời điểm khác nhau để đảm bảo rằng key sẽ không bị bỏ sót, trong phần thí nghiệm này sẽ sử dụng tổng cộng lần lượt 5 bản dump để tiến hành tìm kiếm key.

badrabbit1.raw	11/2/2024 12:53 A...	RAW File	2,580,480 ...
badrabbit2.raw	11/2/2024 12:55 A...	RAW File	2,580,480 ...
badrabbit3.raw	11/2/2024 12:57 A...	RAW File	2,580,480 ...
badrabbit4.raw	11/2/2024 1:00 AM	RAW File	2,580,480 ...
badrabbit5.raw	11/2/2024 1:05 AM	RAW File	2,580,480 ...

Trong quá trình này virus Bad Rabbit sẽ giả checkdisk, thực chất quá trình này đang mã hóa MBR (chương trình khởi động máy tính), sau khi quá trình mã hóa thành công màn hình tổng tiền sẽ được hiện ra.



### III. Xác thực các khóa tìm thấy

Sử dụng công cụ Findaes để tiến hành tìm các khóa AES có thể tồn tại trong file dump

```
File Actions Edit View Help
(kali@kali)-[~/Downloads/findaes-1.2]
$ ./findaes /home/kali/Downloads/volatility3/badrabbit1.raw >> key1
(kali@kali)-[~/Downloads/findaes-1.2]
$
(kali@kali)-[~/Downloads/findaes-1.2]
$ ./findaes /home/kali/Downloads/volatility3/badrabbit2.raw >> key2
(kali@kali)-[~/Downloads/findaes-1.2]
$ ./findaes /home/kali/Downloads/volatility3/badrabbit3.raw >> key3
(kali@kali)-[~/Downloads/findaes-1.2]
$ ./findaes /home/kali/Downloads/volatility3/badrabbit4.raw >> key4
(kali@kali)-[~/Downloads/findaes-1.2]
$ ./findaes /home/kali/Downloads/volatility3/badrabbit5.raw >> key5
(kali@kali)-[~/Downloads/findaes-1.2]
$
```

```
1 Searching /home/kali/Downloads/volatility3/badrabbit1.raw
2 Found AES-128 key schedule at offset 0x3703584:
3 53 a6 4a 16 d4 1f 77 33 b2 bf 49 a8 d5 e4 e6 ba
4 Found AES-128 key schedule at offset 0x6acd518:
5 46 63 86 37 8f a1 3f 2f f3 a2 be 74 06 d6 3f ca
6 Found AES-128 key schedule at offset 0x6acd7a0:
7 10 01 2a 74 37 90 88 c2 5f 12 80 3c 89 33 78 e2
8 Found AES-128 key schedule at offset 0x70cc2a8:
9 46 63 86 37 8f a1 3f 2f f3 a2 be 74 06 d6 3f ca
10 Found AES-128 key schedule at offset 0x70cc530:
11 10 01 2a 74 37 90 88 c2 5f 12 80 3c 89 33 78 e2
12 Found AES-256 key schedule at offset 0x10e12210:
13 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
14 Found AES-256 key schedule at offset 0x11111730:
15 f3 e0 ba e6 86 ec d6 8d 4f 8a 0b e1 65 b1 e4 0d df 7d 04 64 58 48 a0 22 73 70 92 74 ec 94 a6 20
16 Found AES-256 key schedule at offset 0x11c15700:
17 f3 e0 ba e6 86 ec d6 8d 4f 8a 0b e1 65 b1 e4 0d df 7d 04 64 58 48 a0 22 73 70 92 74 ec 94 a6 20
18 Found AES-128 key schedule at offset 0x1bb90260:
19 14 4b 18 81 0a 9f b0 4d 90 0c de 81 2b 6b 44 ad
20
```

Tìm thấy các khóa như bên trên, khóa của 5 file dump được tìm thấy là giống nhau.

Sau đó sử dụng Volatoloty3 để scan file và lưu vào máy với plugin findscan:

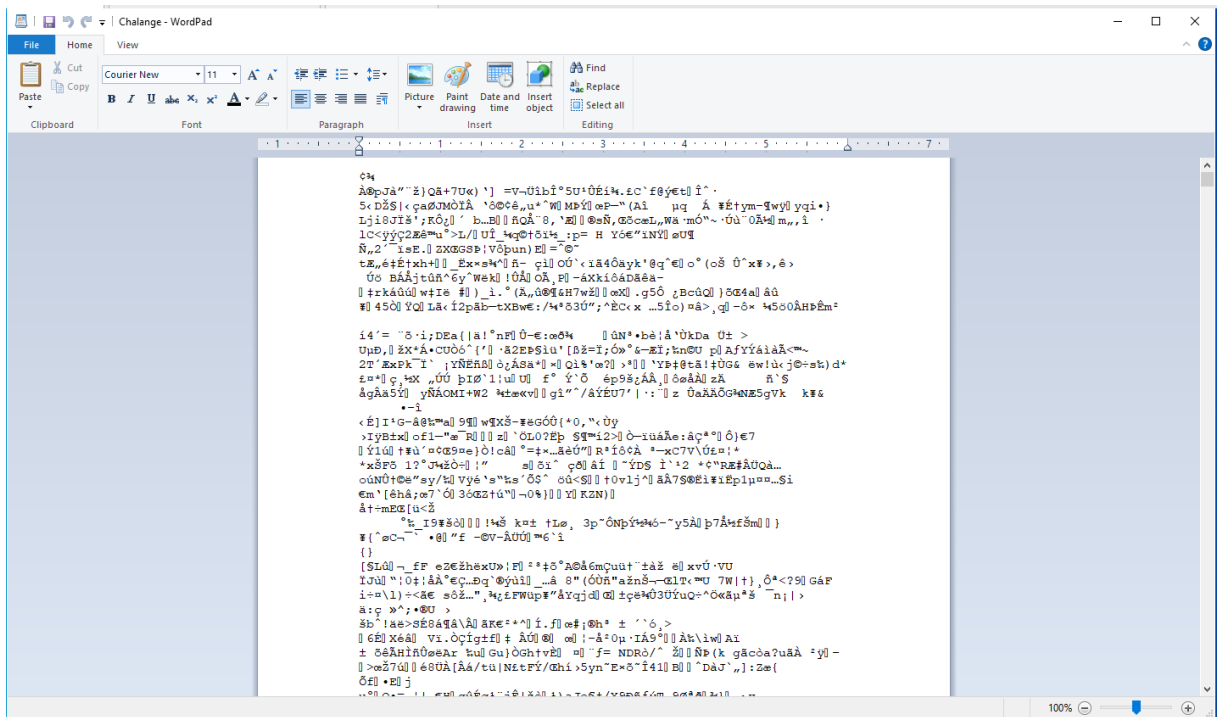
```
Python vol.py -f <tên file> windows.filescan.Filescan >> scan.txt
```

```
(kali@kali)-[~/Downloads/volatility3]
$ python vol.py -f badrabbit5.raw windows.filescan.FileScan >> scan5
Progress: 100.00 PDB scanning finished
```

Tiến hành dump file word đã mã hóa xem có khôi phục bằng key được không

```
2942 0x9aa87d10 \Program Files (x86)\Internet Explorer\iexplore.e
2943 0x9ac08620 \Windows\System32\wpdshext.dll
2944 0x9ac0a200 \Users\hhl\Downloads\Mau_bao_cao (2).docx
2945 0x9ac0a630 \Windows\System32\DriverStore\FileRepository\disp
```





## II. Tạo dòng thời gian của khóa

Ta sẽ dump nhiều lần với mỗi thời điểm khác nhau để đảm bảo rằng key sẽ không bị bỏ sót, trong phần thí nghiệm này sẽ sử dụng tổng cộng lần lượt 5 bản dump để tiến hành tìm kiếm key.

Chalange	11/3/2024 3:30 PM	Office Open XML ...	1,170 KB
badrabbit1	11/3/2024 3:36 PM	RAW File	2,048,000 KB
badrabbit2	11/3/2024 3:36 PM	RAW File	2,048,000 KB
badrabbit3	11/3/2024 3:38 PM	RAW File	2,048,000 KB
badrabbit4	11/3/2024 3:43 PM	RAW File	2,048,000 KB
badrabbit5	11/3/2024 3:45 PM	RAW File	2,048,000 KB

Trong quá trình này virus Bad Rabbit sẽ giả checkdisk, thực chất quá trình này đang mã hóa MBR (chương trình khởi động máy tính), sau khi quá trình mã hóa thành công màn hình tổng tiền sẽ được hiện ra.

Oops! Your files have been encrypted.

If you see this text, your files are no longer accessible.  
You might have been looking for a way to recover your files.  
Don't waste your time. No one will be able to recover them without our  
decryption service.

We guarantee that you can recover all your files safely. All you  
need to do is submit the payment and get the decryption password.

Visit our web service at [caforssztqxzf2nm.onion](http://caforssztqxzf2nm.onion)

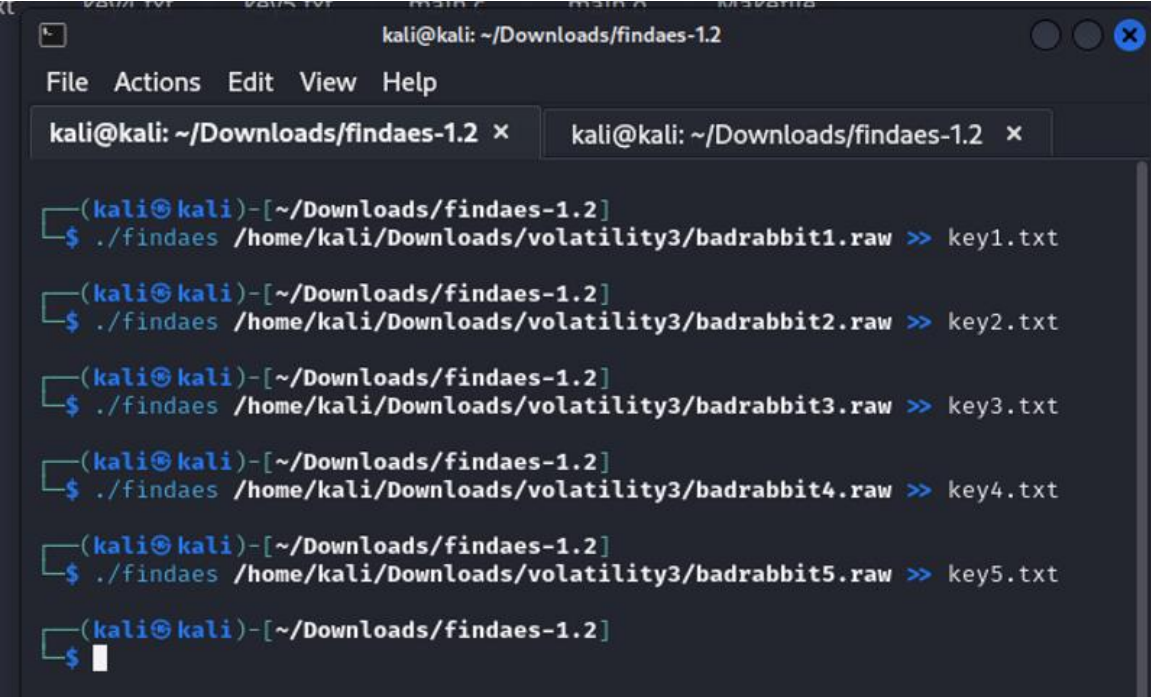
Your personal installation key#1:

```
Z0Kjef04/y9muN/8z0696GbE78uyoSJLB60202tsLcna9lbgAAmWRJN4veYTBXE1
NFxoAgKU8cSB0G05jU2nBanvbEh6spXKJCrv9KxcwhBPcTKcosLAAFuN0dz2CFn
c8LX0ATlu0sQcztkkHg0r/AS8Eaf5ffLS78wgRI3ic8Im8HPRFooh0plnZLEILD
AHmXr6/qrX2T14Av6Bx4F27ma3SdU85rjPxpFYpak9/BhhWqpXRwDo57RWGC21Pf
yEPMy20noS0rfqAkWlRKS9kXmHLe8RvX/Wx4QUKhW86tfF70+bh26JTbrRizPKt
5mqTFB8tAxgrJ/K46Uwppv9NsRYcgDc5Dg==
```

If you have already got the password, please enter it below.  
Password#1: \_

### III. Xác thực các khóa tìm thấy

Sử dụng công cụ Findaes để tiến hành tìm các khóa AES có thể tồn tại trong file dump



```
kali@kali: ~/Downloads/findaes-1.2
File Actions Edit View Help
kali@kali: ~/Downloads/findaes-1.2 x kali@kali: ~/Downloads/findaes-1.2 x

(kali@kali)-[~/Downloads/findaes-1.2]
$ ./findaes /home/kali/Downloads/volatility3/badrabbit1.raw >> key1.txt

(kali@kali)-[~/Downloads/findaes-1.2]
$ ./findaes /home/kali/Downloads/volatility3/badrabbit2.raw >> key2.txt

(kali@kali)-[~/Downloads/findaes-1.2]
$ ./findaes /home/kali/Downloads/volatility3/badrabbit3.raw >> key3.txt

(kali@kali)-[~/Downloads/findaes-1.2]
$ ./findaes /home/kali/Downloads/volatility3/badrabbit4.raw >> key4.txt

(kali@kali)-[~/Downloads/findaes-1.2]
$ ./findaes /home/kali/Downloads/volatility3/badrabbit5.raw >> key5.txt

(kali@kali)-[~/Downloads/findaes-1.2]
$
```



## Hình ảnh của file key1.txt

```
key.txt x key2.txt x key3.txt x
1 Found AES-128 key schedule at offset 0x4389aa0:
2 49 0f 96 f2 67 3d 96 ef 29 bf d9 67 77 20 76 1d
3 Found AES-128 key schedule at offset 0x45f22b0:
4 fd 83 7f 5b 24 c5 e3 47 6d 11 d1 0e b0 62 e4 a0
5 Found AES-128 key schedule at offset 0xc1802f0:
6 f2 64 5b 4b 3b b8 35 de 21 ff b5 b5 23 c9 22 76
7 Found AES-128 key schedule at offset 0x13ddcc40:
8 fa 57 68 2b bd 56 b2 5f af 7f 1d 0b 47 39 c0 52
9 Found AES-256 key schedule at offset 0x13e079b8:
10 17 f3 c2 c8 b9 10 b5 dd 3a b7 49 eb 1a 3b 73 72 a6 53 4d a5 07 31 c7 a0 7d 0f b0 51 1e 20 f2 f3
11 Found AES-256 key schedule at offset 0x13e07c48:
12 66 80 ba 7d 2a 51 12 bf cc 74 4c 06 4b ad 3f 06 af 28 f8 1d 5e f5 c5 ac c4 9e 2d ee 85 e2 2b b3
13 Found AES-256 key schedule at offset 0x1bb9d690:
14 4d 94 6f 22 4c df 3f 73 4d d7 3f 9f 2f 96 51 70 82 d6 11 93 87 dc c3 90 1e f4 b0 d8 72 6c be a5
15 Found AES-256 key schedule at offset 0x1bb9d920:
16 fd 71 0e 6e b0 09 e0 f5 8f 97 8a 07 5e dc 7c 1e ae c4 83 50 d8 b3 77 39 87 f4 42 7f 33 c8 16 8e
17 Found AES-256 key schedule at offset 0x1f5c0ba0:
18 c1 36 b0 9d 14 eb 70 96 44 43 06 a2 ba be 34 bc f4 aa 4c 0a f6 e4 0f 69 e2 82 15 46 2b 94 1e e3
19 Found AES-128 key schedule at offset 0x2bac02d0:
20 c2 3d f8 ab eb 58 87 00 70 cb ba 84 36 00 d5 b4
21 Found AES-128 key schedule at offset 0x2dabc020:
22 f9 af d3 15 a5 57 68 c8 9d 3e 3d 69 a6 69 f7 16
23 Found AES-128 key schedule at offset 0x2dabc7d0:
24 f6 54 1b b2 cb 1a a6 44 fe 61 39 57 6d 0e 86 b9
25 Found AES-256 key schedule at offset 0x317c9ca8:
26 e3 ed 67 7e 7e 2b f4 b8 bb 03 2b b3 e5 6a 5d c5 77 92 ef 96 1f 48 06 92 3c 37 ec 86 93 56 db 2c
27 Found AES-128 key schedule at offset 0x44fd0dc0:
28 83 ab a0 c1 e3 bc d6 ef 12 b0 b1 61 a7 6d f7 f3
29 Found AES-256 key schedule at offset 0x467432f4:
30 30 b5 01 f0 f6 9f c2 b8 3c 09 e1 76 90 90 88 6a bb 55 14 36 fb 15 4a aa 4c fc 66 9f 9e 16 1b 44
31 Found AES-128 key schedule at offset 0x549b13b0:
32 c2 3d f8 ab eb 58 87 00 70 cb ba 84 36 00 d5 b4
33 Found AES-256 key schedule at offset 0x565e3d90:
34 4d 56 40 7e 2d 3b 0a 39 83 6f 89 dd f5 40 ec 48 78 3e 30 ee fa bc 4e fa 34 04 7d 5c 65 86 55 ea
35 Found AES-256 key schedule at offset 0x5d0c0f08:
36 66 08 94 5c a0 ce 0a 28 36 08 f7 0b ea 99 82 95 a0 6e 64 1d c0 a4 f2 8a 05 4a 47 98 02 b2 e7 10
37 Found AES-256 key schedule at offset 0x5f5e3d90:
38 4d 56 40 7e 2d 3b 0a 39 83 6f 89 dd f5 40 ec 48 78 3e 30 ee fa bc 4e fa 34 04 7d 5c 65 86 55 ea
```

Tìm thấy các khóa như bên trên, khóa của 5 file dump được tìm thấy trong trường hợp này là khác nhau.

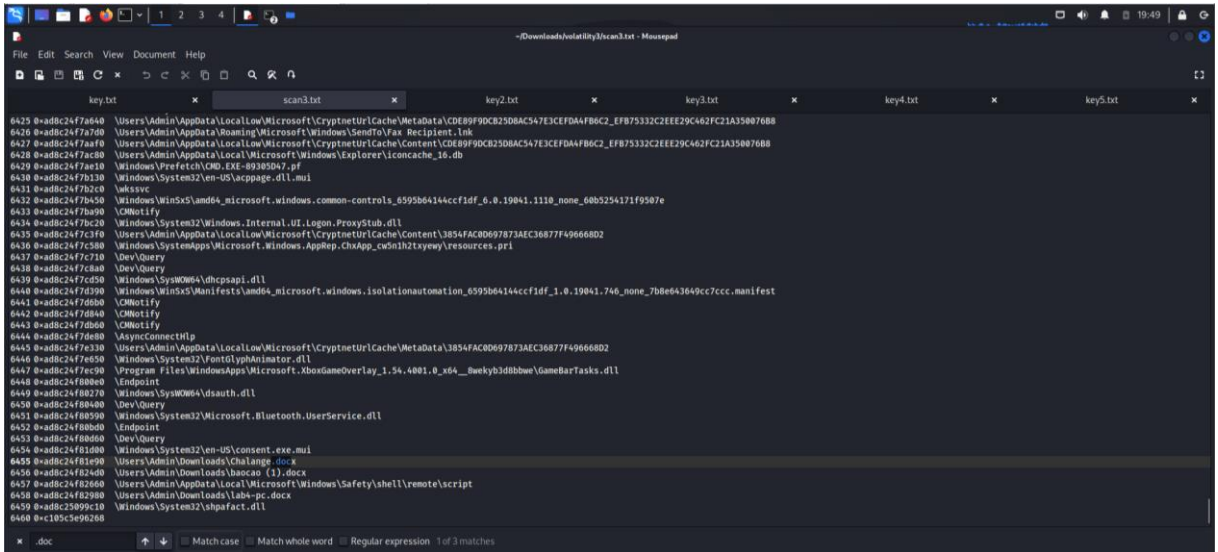
Tiến hành lọc các khóa giống nhau và đưa tất cả khóa vào 1 file

```
~/Downloads/findaes-1.2/key_txt - Mousepad
File Edit Search View Document Help
1 b9 1d ec 2e f0 f5 3a d8 24 11 87 23 84 19 b2 d3
2 66 80 ba 7d 2a 51 12 bf cc 74 4c 06 4b ad 3f 06 af 28 f8 1d 5e f5 c5 ac c4 9e 2d ee 85 e2 2b b3
3 c1 36 b0 9d 14 eb 70 96 44 43 06 a2 ba be 34 bc f4 aa 4c 0a f6 e4 0f 69 e2 82 15 46 2b 94 1e e3
4 30 b5 01 f0 f6 9f c2 b8 3c 09 e1 76 90 90 88 6a bb 55 14 36 fb 15 4a aa 4c fc 66 9f 9e 16 1b 44
5 85 9a de 7c 8d af 55 15 48 96 59 34 c8 63 e6 d0 78 0c 5d c2 32 73 37 10 77 f4 6c 38 1a a7 b0 34
6 c2 3d f8 ab eb 58 87 00 70 cb ba 84 36 00 d5 b4
7 17 f3 c2 c8 b9 10 b5 dd 3a b7 49 eb 1a 3b 73 72 a6 53 4d a5 07 31 c7 a0 7d 0f b0 51 1e 20 f2 f3
8 fd 83 7f 5b 24 c5 e3 47 6d 11 d1 0e b0 62 e4 a0
9 f9 af d3 15 a5 57 68 c8 9d 3e 3d 69 a6 69 f7 16
10 66 08 94 5c a0 ce 0a 28 36 08 f7 0b ea 99 82 95 a0 6e 64 1d c0 a4 f2 8a 05 4a 47 98 02 b2 e7 10
11 4d 56 40 7e 2d 3b 0a 39 83 6f 89 dd f5 40 ec 48 78 3e 30 ee fa bc 4e fa 34 04 7d 5c 65 86 55 ea
12 fd 71 0e 6e b0 09 e0 f5 8f 97 8a 07 5e dc 7c 1e ae c4 83 50 d8 b3 77 39 87 f4 42 7f 33 c8 16 8e
13 6d 36 0f 3b 19 51 6f 97 3f 5d 90 56 5b 71 74 67 8e 49 00 a9 45 70 7b 00 77 96 30 22 85 fb 6d a2
14 e3 ed 67 7e 2b f4 b8 bb 03 2b b3 e5 6a 5d c5 77 92 ef 96 1f 48 06 92 3c 37 ec 86 93 56 db 2c
15 b6 88 b4 85 cd f1 77 b7 e4 02 19 f5 80 56 5a 41
16 f6 54 1b b2 cb 1a a6 44 fe 61 39 57 6d 0e 86 b9
17 4d 94 6f 22 4c df 3f 73 4d d7 3f 9f 2f 96 51 70 82 d6 11 93 87 dc c3 90 1e f4 b0 d8 72 6c be a5
18 f2 64 5b 4b 3b b8 35 de 21 ff b5 b5 23 c9 22 76
19 49 0f 96 f2 67 3d 96 ef 29 bf d9 67 77 20 76 1d
20 83 ab a0 c1 e3 bc d6 ef 12 b0 b1 61 a7 6d f7 f3
21 fa 57 68 2b bd 56 b2 5f af 7f 1d 0b 47 39 c0 52
22
```

Có tổng cộng 21 khóa AES được tìm thấy

Sau đó sử dụng Volatoloty3 để scan file và lưu vào máy với plugin findscan:

```
Python vol.py -f <tên file> windows.filescan.Filescan >> scan.txt
```

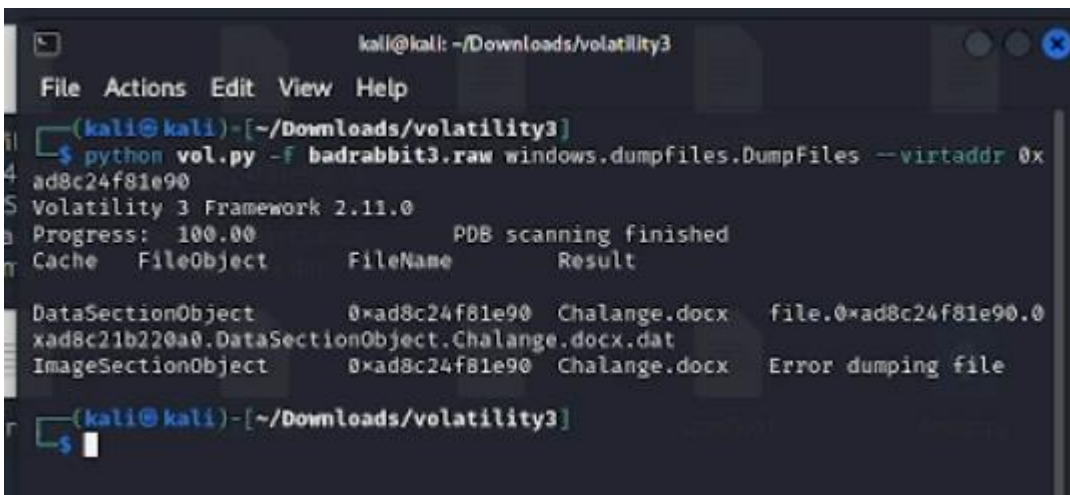


```

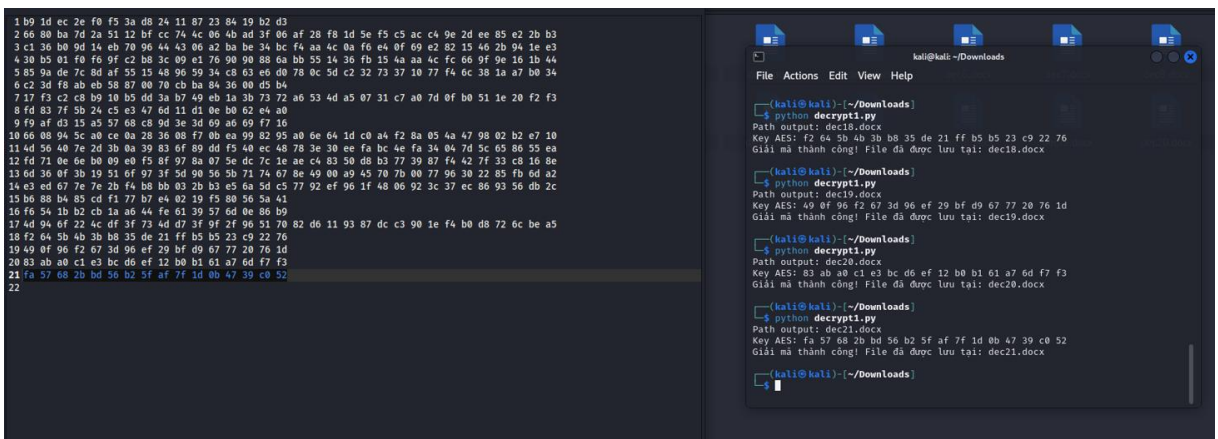
6454 0xad8c24f81d00 \Windows\System32\en-US\consent.exe.mui
6455 0xad8c24f81e90 \Users\Admin\Downloads\Chalange.docx
6456 0xad8c24f824d0 \Users\Admin\Downloads\baocao (1).docx
6457 0xad8c24f82660 \Users\Admin\AppData\Local\Microsoft\Windows\Safet

```

Tiến hành dump file word đã mã hóa xem có khôi phục bằng key được không

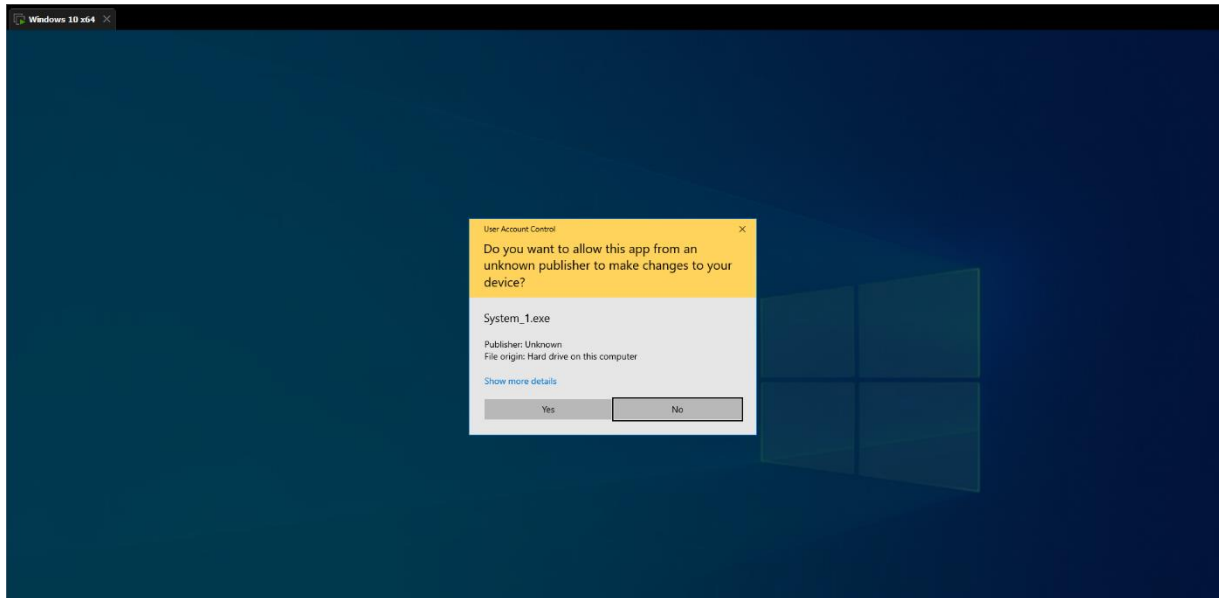


Tiến hành giải mã với từng key đã tìm được cho đến khi tìm được key đúng

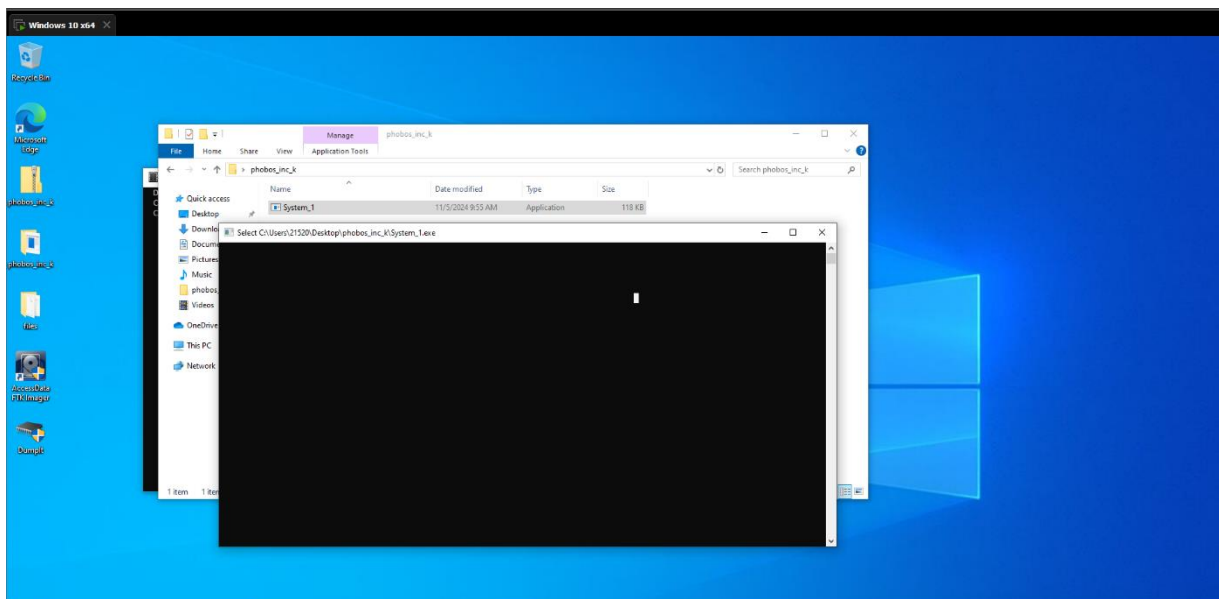


### 5.3. Phobos

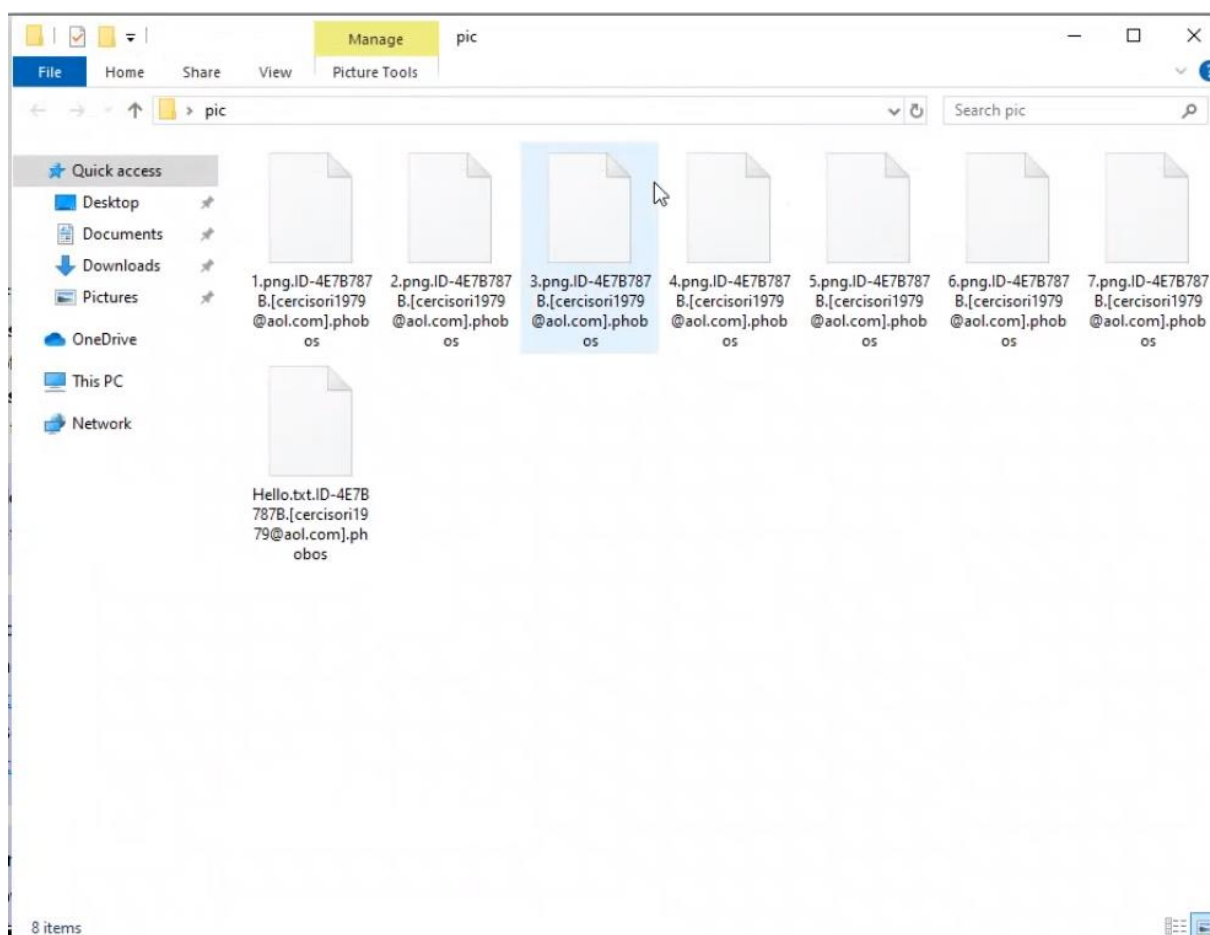
## Thực thi virus trên Windows10



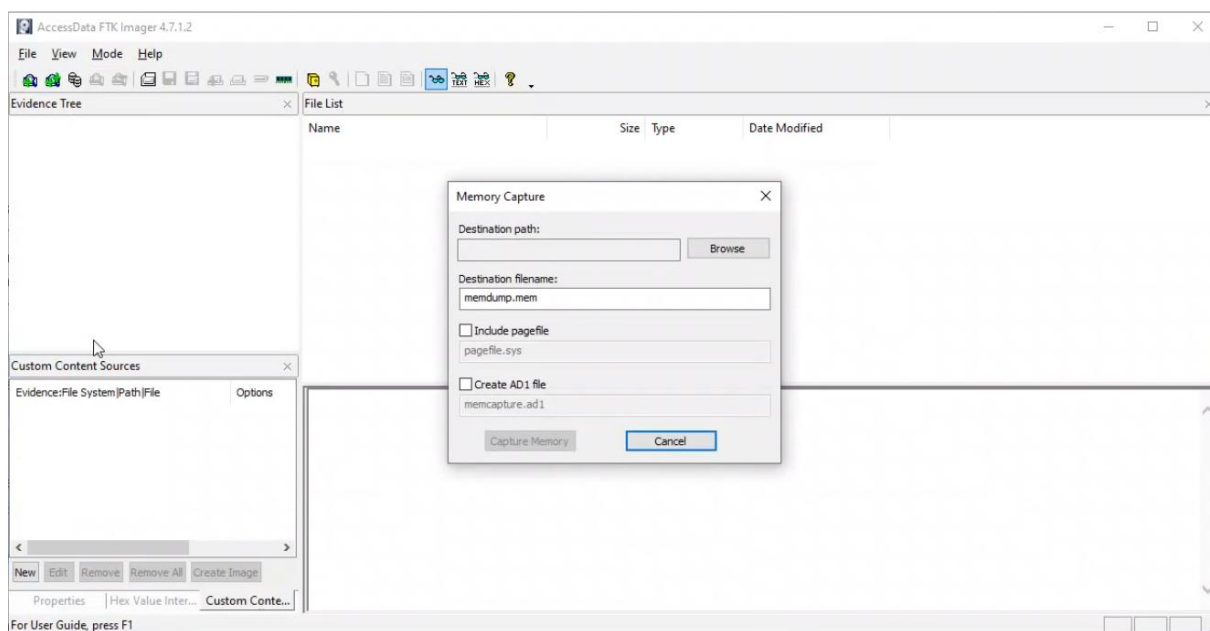
## Virus khi được thực thi











Sử dụng FTK Imager để capture bộ nhớ:

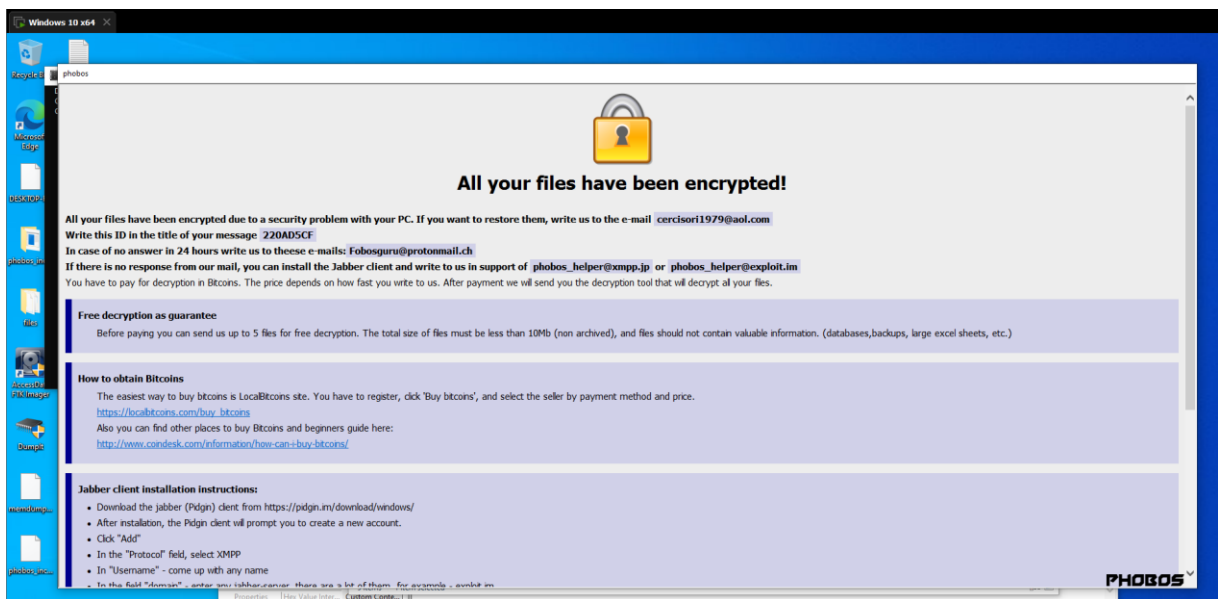


## II. Tạo dòng thời gian của khóa

Ta sẽ dump nhiều lần với mỗi thời điểm khác nhau để đảm bảo rằng key sẽ không bị bỏ sót, trong phần thí nghiệm này sẽ sử dụng tổng cộng lần lượt 4 bản dump để tiến hành tìm kiếm key:

Name	Date modified	Type	Size
 memdump1.mem	11/7/2024 4:42 AM	MEM File	2,097,152 KB
 memdump2.mem	11/7/2024 4:44 AM	MEM File	2,097,152 KB
 memdump3.mem	11/7/2024 4:46 AM	MEM File	2,097,152 KB
 memdump4.mem	11/7/2024 4:47 AM	MEM File	2,097,152 KB

Phobos quét hệ thống, tìm kiếm các tệp dữ liệu quan trọng (tài liệu, hình ảnh, video, cơ sở dữ liệu) và tiến hành mã hóa từng tệp bằng mã hóa AES. Sau khi mã hóa sẽ hiển thị thông báo sau:



### III. Xác thực các khóa tìm thấy

Sử dụng công cụ Findaes để tiến hành tìm các khóa AES có thể tồn tại trong file dump:

```

(kali@kali)-[~/Desktop/Phobos_encrypted/findaes-1.2]
$ ./findaes memdump1.mem memdump2.mem memdump3.mem memdump4.mem
Searching memdump1.mem
Found AES-128 key schedule at offset 0x4389aa0:
7c a6 e8 af b8 d6 91 62 6d b5 8f 29 3e cb f1 bc
Found AES-128 key schedule at offset 0xe4019e0:
e8 ec 42 eb 1c b5 b7 97 a3 ea e9 5a 1f e0 e4 e6
Found AES-128 key schedule at offset 0x2d9482b0:
90 e9 df f3 8e 75 fc 02 eb 57 ba f3 c0 ae fa 62
Found AES-128 key schedule at offset 0x2e9bf2b0:
90 e9 df f3 8e 75 fc 02 eb 57 ba f3 c0 ae fa 62
Found AES-128 key schedule at offset 0x3584f830:
ca 01 03 eb 77 3d b1 38 09 69 53 1f 98 bf 63 b7
Found AES-128 key schedule at offset 0x430492b0:
90 e9 df f3 8e 75 fc 02 eb 57 ba f3 c0 ae fa 62
Found AES-128 key schedule at offset 0x44714340:
48 7a 5a db fa 43 7f 87 b2 e6 95 3d 42 62 c1 d1
Found AES-256 key schedule at offset 0x47e35d20:
93 f1 71 b1 f4 ec a5 ec f0 36 9e 40 17 af 8b b2 be c2 ef bf 6e 52 0f 11 a6 69 2b f1 d1 ea 78 7e
Found AES-256 key schedule at offset 0x49346a60:
91 bf 5a 25 d6 60 8b 8e 57 a6 50 61 c9 37 3d 8d 9e a1 a3 3a e5 49 6a 35 6b ee 2e 32 92 d9 96 57
Found AES-256 key schedule at offset 0x64358a60:
91 bf 5a 25 d6 60 8b 8e 57 a6 50 61 c9 37 3d 8d 9e a1 a3 3a e5 49 6a 35 6b ee 2e 32 92 d9 96 57
Found AES-256 key schedule at offset 0x69754c88:
54 22 5f d0 17 24 5a 1a cf 05 f0 46 10 dd b3 0c ee 59 a3 7d 87 a6 8c 8f c1 c2 21 e0 d7 aa 50 c8
Found AES-128 key schedule at offset 0x6b167340:
48 7a 5a db fa 43 7f 87 b2 e6 95 3d 42 62 c1 d1
Found AES-256 key schedule at offset 0x6e4d5c88:
54 22 5f d0 17 24 5a 1a cf 05 f0 46 10 dd b3 0c ee 59 a3 7d 87 a6 8c 8f c1 c2 21 e0 d7 aa 50 c8
Found AES-128 key schedule at offset 0x7053d2b0:
90 e9 df f3 8e 75 fc 02 eb 57 ba f3 c0 ae fa 62
Found AES-128 key schedule at offset 0x73b012b0:
31 78 e3 10 01 2b d3 dc 1f 44 b3 69 92 36 4a 2a
Found AES-128 key schedule at offset 0x73b017d0:
30 85 98 7c bf b4 c5 9f aa 62 b9 73 5b 84 15 54
Searching memdump2.mem
Found AES-128 key schedule at offset 0x4389aa0:
7c a6 e8 af b8 d6 91 62 6d b5 8f 29 3e cb f1 bc
Found AES-256 key schedule at offset 0x78110c0:
44 97 41 82 ff e1 a6 26 03 48 fb 13 21 80 fb 35 bb 2d 67 08 78 a8 42 6b 6d 90 a6 da 2b 85 3d 25

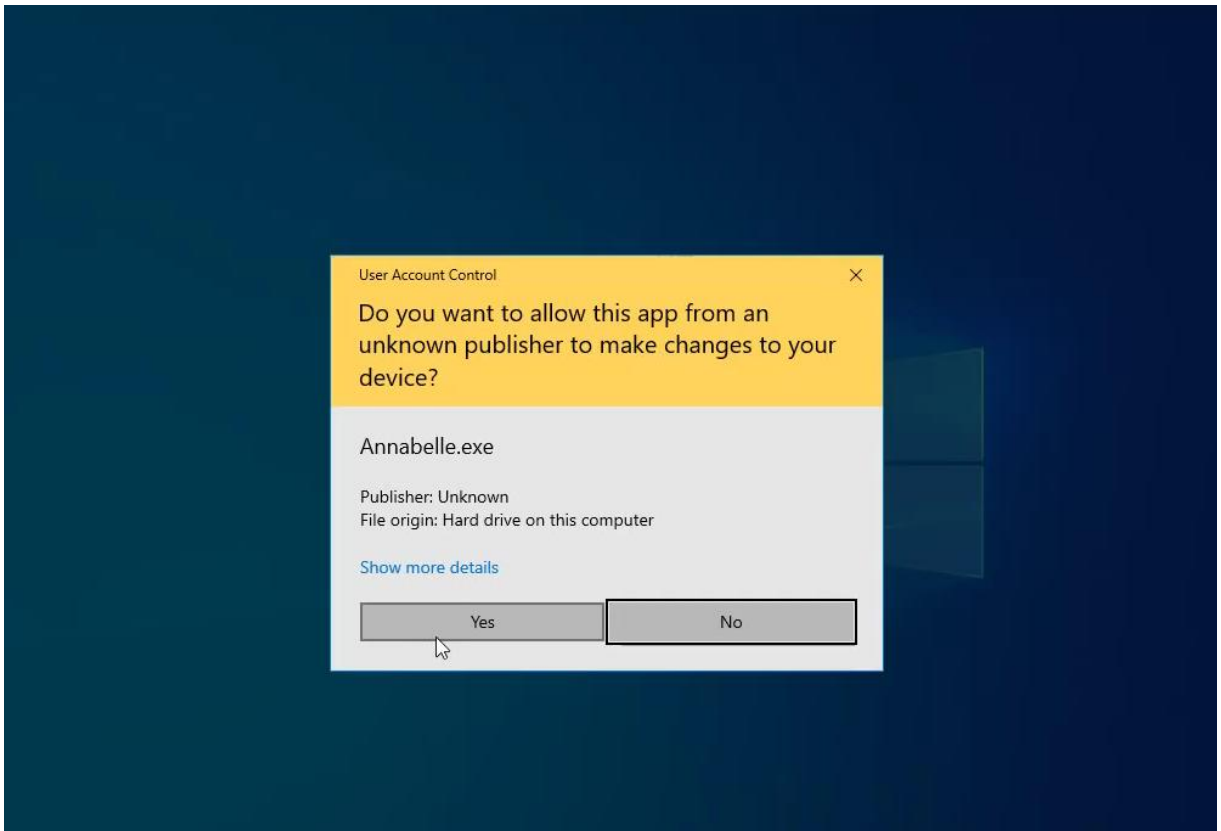
```

Các key này có thể đưa vào để thử giải mã file

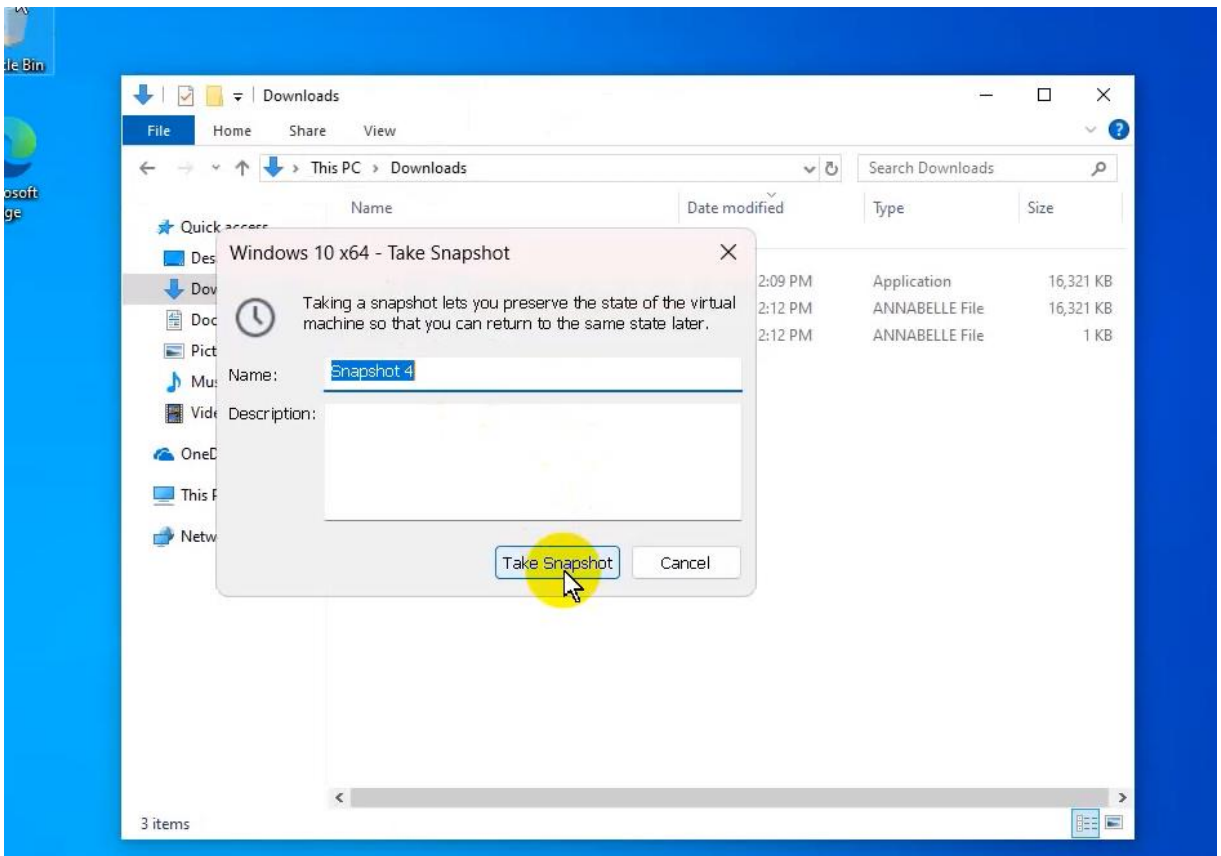
## 5.4. Annabelle

### I. Xác định khóa trong bộ nhớ

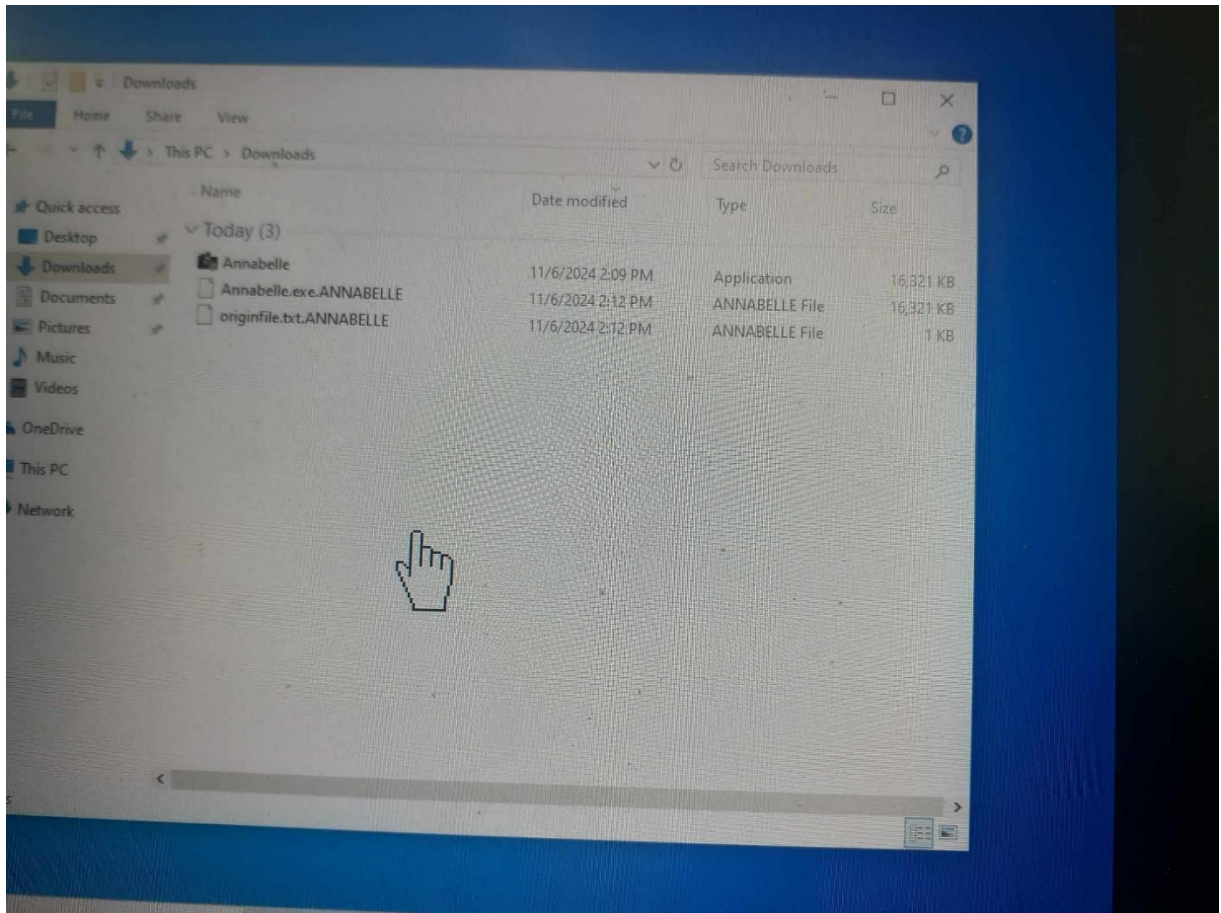
Thực thi virus Annabelle trên Windows10



Sử dụng chức năng snapshot của vmware để dump file dưới dạng vmem(tạo ra các file snapshot lưu ở folder window10 trên máy chính):



Sau khi chạy Annabelle 1 thời gian, các tệp sẽ được mã hóa và gắn đuôi vào:



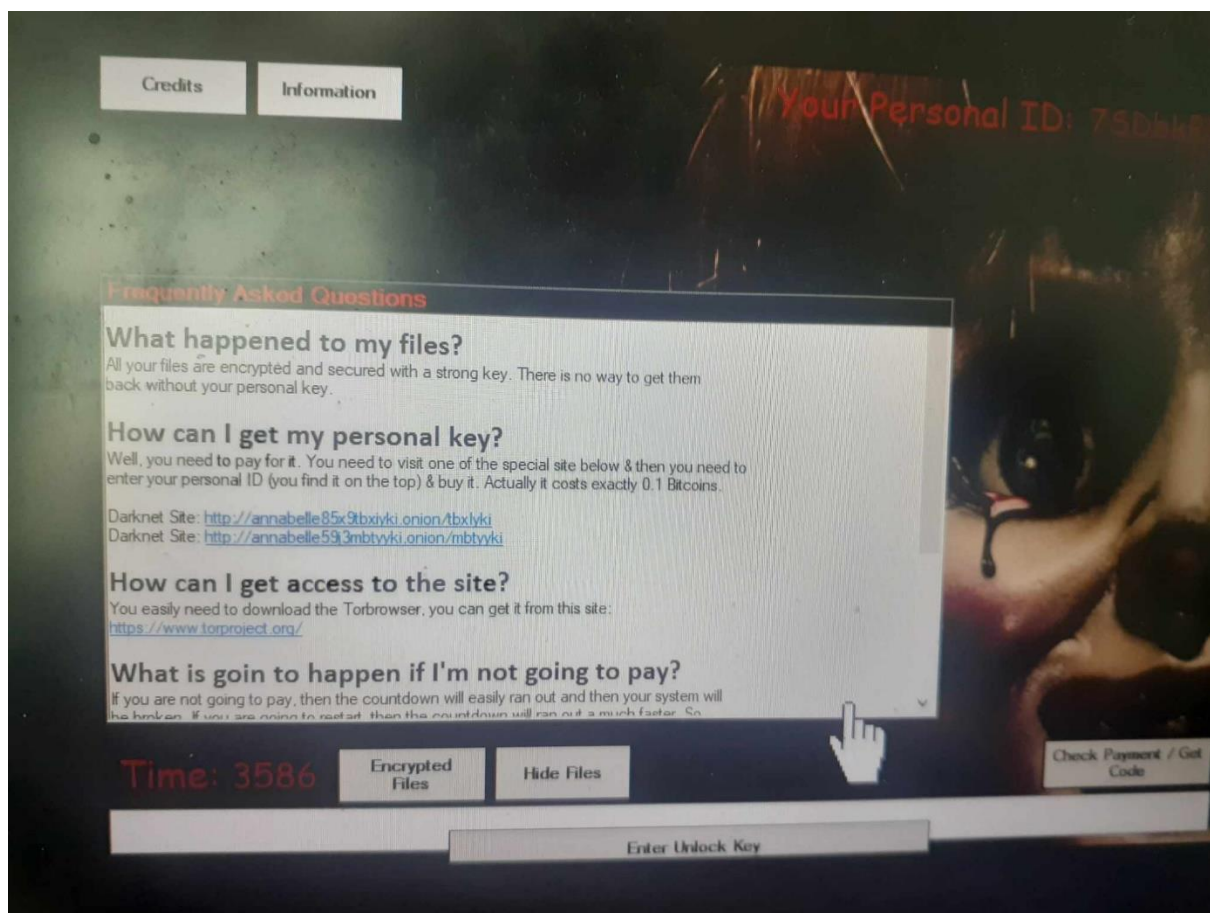
## II. Tạo dòng thời gian của khóa

Ta sẽ dump nhiều lần với mỗi thời điểm khác nhau để đảm bảo rằng key sẽ không bị bỏ sót, trong phần thí nghiệm này sẽ sử dụng tổng cộng lần lượt 10 bản dump để tiến hành tìm kiếm key.

Name	Date modified	Type	Size
Yesterday			
Windows 10 x64-Snapshot1.vmem	11/6/2024 2:11 PM	VMEM File	2,097,152 ...
Windows 10 x64-Snapshot2.vmem	11/6/2024 2:12 PM	VMEM File	2,097,152 ...
Windows 10 x64-Snapshot3.vmem	11/6/2024 2:12 PM	VMEM File	2,097,152 ...
Windows 10 x64-Snapshot4.vmem	11/6/2024 2:12 PM	VMEM File	2,097,152 ...
Windows 10 x64-Snapshot5.vmem	11/6/2024 2:12 PM	VMEM File	2,097,152 ...
Windows 10 x64-Snapshot6.vmem	11/6/2024 2:13 PM	VMEM File	2,097,152 ...
Windows 10 x64-Snapshot7.vmem	11/6/2024 2:13 PM	VMEM File	2,097,152 ...
Windows 10 x64-Snapshot8.vmem	11/6/2024 2:13 PM	VMEM File	2,097,152 ...
Windows 10 x64-Snapshot9.vmem	11/6/2024 2:14 PM	VMEM File	2,097,152 ...
Windows 10 x64-Snapshot10.vmem	11/6/2024 2:14 PM	VMEM File	2,097,152 ...



Trong quá trình khởi động lại win, ransom Annabelle thường sẽ giả checkdisk hoặc giả việc khởi động win với thời gian dài nhưng nó thực chất là quá trình mã hóa MBR (chương trình khởi động máy tính), sau khi quá trình mã hóa thành công màn hình sẽ hiển thị như sau:



### III. Xác thực các khóa tìm thấy

Sử dụng công cụ Findaes để tiến hành tìm các khóa AES có thể tồn tại trong file dump, cần đổi tên snapshot trước vì tên gốc rất dài và có các khoảng trống nếu không cẩn thận sẽ không nhận diện được file:

```

d@d-virtual-machine:~/Downloads/findaes-1.2$ ./findaes /home/d/Snapshot1.vmem
Searching /home/d/Snapshot1.vmem
d@d-virtual-machine:~/Downloads/findaes-1.2$ ./findaes /home/d/Downloads/findaes-1.2/Snapshot2.vmem
Searching /home/d/Downloads/findaes-1.2/Snapshot2.vmem
Found AES-256 key schedule at offset 0x9817e0:
36 84 e1 ba ef 84 f8 ac c3 35 26 01 7e d0 64 94 b3 7c fa 62 46 2a 7e ac fb 00 c0 87 b7 a0 f1 94
Found AES-128 key schedule at offset 0x4199aa0:
f2 fe bf 94 2e 39 c8 38 c8 46 49 64 c2 c2 d0 2f
Found AES-256 key schedule at offset 0x8fecc00:
3e e5 2e 10 12 ff 62 b7 85 c8 22 29 55 2b 7c c9 98 e4 aa 78 32 5b ca 9c a1 e3 fc e2 be af 41 18
Found AES-128 key schedule at offset 0xf694170:
2f 2c 55 81 71 6e dc cb 0f d3 d9 0e 99 3a ad c7
Found AES-128 key schedule at offset 0xf694400:
e2 06 54 1a d7 a3 f4 66 25 00 9e 55 70 01 df 49
Found AES-256 key schedule at offset 0xf694bb0:
fa ee 86 50 2c d7 e9 a6 51 83 aa 92 5a 71 d4 9d 31 2f e8 4b e7 07 d4 27 37 de 8f 56 6a 84 41 97
Found AES-256 key schedule at offset 0x1ab92730:
69 28 b1 9f 2b c6 db d0 8f c7 19 6e 44 c9 c0 2b d5 c3 68 d4 e9 04 26 6d 2b c7 1a 44 f7 5f 3a 89
Found AES-256 key schedule at offset 0x398de730:
eb 4a 5d 51 38 e0 6e a2 46 ec e9 87 93 ef fa 01 a9 d8 9c 2f f7 ce 07 78 63 be 00 f7 a9 22 2f ff
Found AES-128 key schedule at offset 0x3d2e35f0:
af be 61 e4 c0 c0 3f 2c 44 7c 92 49 a5 dd c3 4b
Found AES-128 key schedule at offset 0x3df64250:
a6 78 66 9d 91 29 90 1f 52 83 c3 fd e6 80 2f 1e
Found AES-256 key schedule at offset 0x41895730:
33 d6 1f 6e 63 eb 62 7d 92 4c 26 ce 4b 91 0e 79 63 c5 38 e6 f7 a4 c4 37 51 05 50 b8 b9 b9 18 8f
Found AES-128 key schedule at offset 0x418959c0:
14 8c 3e 9c ef 71 8a a5 0d d8 14 61 73 11 ec 90
Found AES-128 key schedule at offset 0x47a24a60:
fe 9c 84 74 bd be 33 01 5c ea 66 75 e2 b0 d5 09
Found AES-128 key schedule at offset 0x47a24cf0:
d1 81 e2 3c 57 e4 7c c3 90 17 91 aa b4 fa 1a bd
Found AES-128 key schedule at offset 0x48a8d8a0:
99 16 b4 7f af 81 62 24 6b c0 dc 8a db c3 2a fc
Found AES-128 key schedule at offset 0x4b3962b0:
f4 07 37 65 4d e9 31 85 5e fa a5 5f e3 47 cf e7
Found AES-128 key schedule at offset 0x4de72aa0:
e4 33 a7 bb 36 9d 47 12 a9 e3 e3 fa 2f b4 8d ca
Found AES-256 key schedule at offset 0x5b902d30:
69 28 b1 9f 2b c6 db d0 8f c7 19 6e 44 c9 c0 2b d5 c3 68 d4 e9 04 26 6d 2b c7 1a 44 f7 5f 3a 89
Found AES-256 key schedule at offset 0x76dcc400:
3e 9f c0 f4 ac 96 a9 fc 65 90 c7 03 2e 53 f5 ff a2 b6 d0 56 cf d3 8d 46 ca 28 66 f4 6b a9 49 cb
Found AES-256 key schedule at offset 0x7b0e74b0:
a2 e7 89 fb 35 cc 21 26 f0 ad 81 46 29 96 fa c2 be 72 3a 2f f1 ff 54 d8 0f f0 6f 31 22 2f aa 21
Found AES-256 key schedule at offset 0x7b0e7eb0:
a2 e7 89 fb 35 cc 21 26 f0 ad 81 46 29 96 fa c2 be 72 3a 2f f1 ff 54 d8 0f f0 6f 31 22 2f aa 21
d@d-virtual-machine:~/Downloads/findaes-1.2$

```

Thực hiện filescan tìm file bị mã hóa trước đó:

```

Open  [+] scan [Read-Only] Save [Menu] [Close] [Exit]
~/volatility3

1 Volatility 3 Framework 2.11.0
2 Volatility 3 Framework 2.11.0
3
4 Offset Name
5
6 0xa78ff068f4f0 \Windows\System32\drivers\storqosflt.sys
7 0xa78ff0c5e0b0 \Windows\System32\drivers\watchdog.sys
8 0xa78ff0c5e390 \Windows\System32\drivers\dxgkrnl.sys
9 0xa78ff0c5e500 \Directory
10 0xa78ff0c5e670 \Windows\System32\drivers\npfs.sys
11 0xa78ff0c5e950 \Windows\System32\CatRoot\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\Microsoft-
    Windows-Client-Desktop-Required-Package0517~31bf3856ad364e35~amd64~~10.0.19041.3803.cat
12 0xa78ff0c5eac0 \Windows\System32\drivers\msfs.sys
13 0xa78ff0c5ec30 \Windows\System32\drivers\cimfs.sys
14 0xa78ff0c5f080 \Windows\System32\DriverStore\FileRepository\basicdisplay.inf_amd64_19e58b6267591a82\BasicDispl
15 0xa78ff0c5f4d0 \Windows\System32\CatRoot\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\Microsoft-
    OneCore-Containers-merged-Package~31bf3856ad364e35~amd64~~10.0.19041.3636.cat
16 0xa78ff0c5fd70 \Windows\System32\drivers\tdx.sys
17 0xa78ff0c5fee0 \Windows\System32\drivers\tdi.sys
18 0xa78ff0c60330 \Windows\System32\drivers\beep.sys
19 0xa78ff0c608f0 \Windows\System32\DriverStore\FileRepository\basicrender.inf_amd64_d3f5994a67770b50\BasicRender
20 0xa78ff0c60bd0 \Windows\System32\CatRoot\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\Microsoft-
    Windows-Client-Desktop-Required-Package0516~31bf3856ad364e35~amd64~~10.0.19041.3803.cat
21 0xa78ff0c60d40 \Windows\System32\drivers\null.sys
22 0xa78ff0c60eb0 \Windows\System32\CatRoot\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\Microsoft-
    Windows-Client-Desktop-Required-Package05112~31bf3856ad364e35~amd64~~10.0.19041.3803.cat
23 0xa78ff0c61190 \Windows\System32\drivers\netbt.sys
24 0xa78ff0c618c0 \Windows\System32\drivers\afunix.sys
25 0xa78ff0c61ba0 \Windows\System32\drivers\afd.sys
26 0xa78ff0c61d10 \Windows\System32\drivers\vwifflt.sys
27 0xa78ff0efc570 \Users\dung\AppData\Roaming\Microsoft\Protect\S-1-5-21-1393527948-3461380206-204072956-1001\Pre
28 0xa78ff0f683e0 \Windows\System32\normaliz.dll
29 0xa78ff0f68570 \Windows\System32\coml2.dll
30 0xa78ff0f68700 \Windows\System32\user32.dll

```

Tiến hành dump file:

```

scan      setup.py
root@d-virtual-machine:/home/d/volatility3# python3 vol.py -f Snapshot10.vmem wi
ndows.dumpfiles --virtaddr 0xa78ff4f11830
Volatility 3 Framework 2.11.0
WARNING volatility3.framework.layers.vmware: No metadata file found alongside V
MEM file. A VMSS or VMSN file may be required to correctly process a VMEM file.
These should be placed in the same directory with the same file name, e.g. Snaps
hot10.vmem and Snapshot10.vmss.
Progress: 100.00      PDB scanning finished
Cache  FileObject      FileName      Result
DataSectionObject      0xa78ff4f11830  osver.txt      file.0xa78ff4f11830.0xa7
8ff51ef7f0.DataSectionObject.osver.txt.dat

```

Tiến hành giải mã cho đến khi khôi phục thành công được file.

## Tham khảo

Davies, S. R., Macfarlane, R., & Buchanan, W. J. (2020, April 5). Evaluation of live forensic techniques in ransomware attack mitigation. *School of Computing, Edinburgh Napier University, Edinburgh, UK.*