

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



**MÔN HỌC: PHÁP CHỨNG KỸ THUẬT SỐ**

**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

*Evaluation of Live Forensic Techniques  
in Ransomware Attack Mitigation*  
*Đánh giá kỹ thuật pháp chứng theo thời gian thực  
trong việc giảm thiểu cuộc tấn công của Ransomware*

**Giảng viên hướng dẫn: Lê Đức Thịnh**

**NT334.O21.ATCL**

**Sinh viên thực hiện :**

**21522492 - Ngô Minh Quân**

**21522312 - Phùng Đức Lương**

**21522620 – Hồ Ngọc Thiện**

**21522483 - Chu Nguyễn Hoàng Phương**

**TP.HCM, Ngày 22 tháng 4 năm 2024**

## Mục Lục

1. Mở đầu .....	4
2. Các công việc liên quan .....	4
2.1. Pháp chứng thời gian thực (Live Forensics) và thu thập bộ nhớ .....	4
2.2. Phân tích pháp chứng thời gian thực .....	5
2.2.1. Kiểm tra phương pháp ghi nhớ .....	5
2.2.2. Nhận dạng các khóa trong bộ nhớ .....	5
2.2.3. Xác định khóa AES .....	6
2.2.4. Phương pháp .....	7
3. Thiết kế .....	7
3.1. Thiết kế môi trường .....	7
3.2. Thiết kế thử nghiệm .....	8
3.2.1. Thử nghiệm phần 1 – Xác định khóa trong bộ nhớ .....	8
3.2.2. Thử nghiệm phần 2 - Tạo biểu đồ thời gian cho khóa .....	8
3.2.3. Thử nghiệm phần 3 – Xác thực các khóa tìm thấy .....	9
3.2.4. Thử nghiệm tổng hợp .....	10
4. Triển khai và Kết quả .....	11
4.1. Lựa chọn mẫu ransomware .....	11
4.1.1. Một số loại ransomware khác .....	11
4.2. Cấu hình máy thử nghiệm .....	12
4.3. Các thử nghiệm .....	12
4.3.1. Thử nghiệm phần 1 - Xác định khóa trong bộ nhớ .....	12
4.3.2. Thử nghiệm phần 2 - Tạo biểu đồ thời gian cho khóa .....	13
4.3.3. Thử nghiệm 3 - Xác thực khóa được tìm thấy .....	13
5. Kết quả và thảo luận .....	13
5.1. NotPetya .....	13
5.1.1. Thí nghiệm phần 1 – Xác định khóa trong bộ nhớ .....	14
5.1.2 Thí nghiệm phần 2 – Tạo biểu đồ thời gian khóa .....	14
5.1.3. Thí nghiệm phần 3 - Xác thực khóa đã tìm thấy .....	15
5.2. Bad Rabbit .....	15
5.2.1 Thí nghiệm phần 1 – Xác định khóa trong bộ nhớ .....	15
5.2.2. Thí nghiệm phần 2 - Tạo dòng thời gian khóa .....	16

5.2.3. Thí nghiệm phần 3 - Xác thực khóa đã tìm thấy .....	16
5.3. Phobos .....	16
5.3.1. Thí nghiệm phần 1 - Xác định khóa trong bộ nhớ .....	17
5.3.2. Thí nghiệm phần 2 - Tạo dòng thời gian khóa .....	17
5.3.3. Thí nghiệm phần 3 - Xác thực khóa đã tìm thấy .....	18
6. Kết quả Demo .....	18
6.1. Notpetya .....	18
6.2. Bad Rabbit .....	23
7. Những điều kiện cần thiết để thí nghiệm thực hiện thành công .....	26

# 1. Mở đầu

Mục tiêu của bài báo này là khám phá xem các kỹ thuật thường được sử dụng trong pháp chứng theo thời gian thực có thể áp dụng vào phân tích ransomware và từ đó cho phép nhà điều tra khám phá các đoạn mã hóa hữu ích từ phần mềm độc hại để có thể đảo ngược hiệu quả của việc thực thi ransomware.

Tồn tại một lĩnh vực nghiên cứu riêng liên quan đến việc nghiên cứu và phát triển các kỹ thuật pháp chứng theo thời gian thực và cụ thể hơn là phân tích bộ nhớ thời gian thực. Một số nghiên cứu trong lĩnh vực này đã được thực hiện để khôi phục các đoạn mã hóa, đặc biệt là các khóa mã hóa, từ nội dung của bộ nhớ hệ thống nơi các quá trình mật mã hóa đang hoạt động. Nhiều nghiên cứu này đã chứng minh rất thành công (Balogh and Pondelik, 2011; Maartmann-Moe et al., 2009), cho phép các nhà nghiên cứu xác định các khóa mã hóa được sử dụng bởi các chương trình mật mã hóa và sau đó sử dụng chúng để giải mã các tệp. Tuy nhiên, không tìm thấy nghiên cứu cụ thể nào mà các kỹ thuật này đã được áp dụng vào các máy đang chạy ransomware.

Đóng góp chính của nghiên cứu này là:

- Đầu tiên cung cấp xác nhận rằng các kỹ thuật pháp chứng có thể được áp dụng để điều tra bộ nhớ tạm thời của các máy bị nhiễm ransomware và có thể được sử dụng để trích xuất các đoạn mã hóa hữu ích có thể được sử dụng để giảm thiểu tác động của cuộc tấn công.
- Thứ hai, xác định thời gian và cách mà hành vi của cuộc tấn công ransomware diễn ra và chỉ ra khi nào và trong bao lâu các 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 hại nhất mà người dùng internet đang phải đối mặt ngày nay và thường được phân loại theo loại mã hóa được sử dụng :

- Symmetric Crypto-Ransomware (SCR): Sử dụng một khóa cho cả quá trình mã hóa và giải mã, giúp cuộc tấn công hoàn thành trong thời gian ngắn hơn và giảm khả năng bị phát hiện.
- Asymmetric Crypto-Ransomware (ACR): Sử dụng các khóa khác nhau cho quá trình mã hóa và giải mã.
- Hybrid Key Crypto-Ransomware (HCR): Đầu tiên sử dụng mã hóa đối xứng để mã hóa các tệp của người dùng càng nhanh càng tốt. Sau đó, khóa đối xứng được mã hóa bằng mã hóa không đối xứng.

### 2.1. Pháp chứng thời gian thực (Live Forensics) và thu thập bộ nhớ

Phương pháp phân tích pháp y tính được sử dụng để phân tích bằng chứng từ một hệ thống máy tính đã được tắt nguồn. Vấn đề của phương pháp này là các thông tin quan trọng được lưu trữ trong bộ nhớ tạm thời của máy tính bị mất khi máy tính được tắt. Các ví dụ về thông tin có thể có trong bộ nhớ là các khóa mã hóa, chi tiết kết nối đang mở, quá trình đang chạy, người dùng đã đăng nhập. Để giải quyết vấn đề này, một phương pháp pháp y bổ sung được gọi là **Live forensics** đã được phát triển. **Live forensics** chủ yếu nhằm vào dữ liệu tạm thời của máy tính chỉ có thể được thu thập từ một hệ thống đang chạy. Khi áp dụng vào phân tích ransomware, các kỹ thuật **Live forensics** có thể được kết hợp với kỹ thuật phân tích malware.

Một khía cạnh quan trọng của **Live forensics** là kiểm tra bộ nhớ hệ thống nơi mã độc đang chạy. Việc kiểm tra này thường được thực hiện ngoại tuyến để nội dung của bộ nhớ không bị ảnh hưởng bởi quá trình kiểm tra hoặc các công cụ thu thập bộ nhớ được sử dụng. Để làm được điều này, bộ nhớ của hệ thống cần được thu thập và lưu trữ:

### 1. Phương pháp dựa trên phần mềm:

Thường liên quan đến việc thực thi các chương trình trích xuất. Vấn đề của phương pháp này là việc thực thi phần mềm sẽ ảnh hưởng đến nội dung của bộ nhớ hệ thống đã được ghi lại.

### 2. Phương pháp dựa trên phần cứng:

Thường liên quan đến việc kết nối các thiết bị như thẻ PCMCIA hoặc USB, và không luôn luôn thực tế trong các tình huống thực tế khi cần truy cập vật lý vào máy tính.

### 3. Các kỹ thuật dựa trên công nghệ ảo hóa:

Sử dụng phương pháp ảo hóa, một bản chụp của bộ nhớ tạm thời của hệ thống được trích xuất bằng các công cụ được cung cấp bởi phần mềm ảo hóa. Bản chụp này sau đó được kiểm tra bởi một nhà phân tích bằng cách sử dụng các công cụ pháp y chuyên dụng. Đương nhiên, để sử dụng phương pháp này, hệ thống phải đang chạy trong một môi trường ảo hóa.

**Ưu điểm của phương pháp này** là không có dấu vết của bất kỳ chương trình trích xuất nào tồn tại trong bộ nhớ đã được ghi lại và bất kỳ chương trình độc hại đang chạy cũng không nhận biết rằng chúng đang bị phân tích hoặc rằng việc trích xuất bộ nhớ đã được thực hiện. Thách thức của việc thu thập bộ nhớ trong ngữ cảnh này là tìm ra các dấu vết mật mã hóa, chẳng hạn như các khóa mã hóa, một cách cho phép thiết bị mục tiêu tiếp tục hoạt động bình thường trong quá trình thu thập bộ nhớ.

## 2.2. Phân tích pháp chứng thời gian thực

Một số nghiên cứu đã được tiến hành trước đây để khảo sát khả năng sử dụng kỹ thuật **Live forensics** để khám phá các khóa mã hóa có thể tồn tại trong bộ nhớ của máy tính. Tuy không tìm thấy bất kỳ tài liệu nào tập trung đặc biệt vào ransomware, nhưng đã có những nghiên cứu tương tự về xác định khóa trong bộ nhớ tạm thời cho các SSH tunnels, ổ đĩa được mã hóa, WinRAR, WinZip và Skype.

Một số bài báo nghiên cứu xác nhận giả định rằng để hệ thống có thể mã hóa và giải mã dữ liệu, thuật toán mật mã cần truy cập vào các khóa mã hóa và thông thường chúng được lưu trữ trong bộ nhớ tạm thời. **Balogh** cho biết rằng việc mã hóa trong thời gian thực chỉ diễn ra trong bộ nhớ, điều đó có nghĩa là các khóa mã hóa cũng phải hiện diện ở đó để giải mã. Đối với mã hóa đối xứng, điều này có nghĩa là các khóa cần thiết cho việc giải mã cũng có thể được khôi phục từ bộ nhớ. Liên quan đến quản lý khóa, **Maartmann-Moe** cho biết rõ ràng các khóa mã hóa cần phải có mặt trong bộ nhớ trong quá trình mã hóa khi sử dụng phần cứng máy tính tiêu chuẩn.

Trong các thử nghiệm tỉ mỉ được tiến hành trên 10 hệ thống mật mã khác nhau, các nhà nghiên cứu luôn có thể khôi phục lại tất cả các khóa mật mã từ bộ nhớ cho mọi ứng dụng được kiểm tra bằng công cụ được phát triển riêng gọi là "**Interrogate**". Mặc dù các nhà nghiên cứu này chưa điều tra cụ thể về ransomware, nhưng kết quả của họ mạnh mẽ cho thấy việc trích xuất các khóa mã hóa ransomware bằng cách sử dụng các kỹ thuật tương tự là khả thi.

#### 2.2.1. Kiểm tra phương pháp ghi nhớ

Quy trình thông thường để tìm kiếm một cái gì đó là cố gắng xác định một đặc điểm của cái đang được tìm kiếm 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 các mã và dữ liệu không được chọn ngẫu nhiên và rõ ràng sự phân biệt này có ý nghĩa. Khi dữ liệu là ngẫu nhiên, nó có entropy cao hơn so với thông tin có mô hình. Điều này có nghĩa là có thể tìm thấy các khóa mật mã trong các phần dữ liệu khác bằng cách xác định các phần có entropy không bình thường cao. Các tác giả đã phát hiện ra rằng khối bộ nhớ chứa các khóa AES chính và phụ có cấu trúc dễ nhận biết và entropy cao.

Trên thực tế, các khóa mật mã đối xứng chỉ là các chuỗi ngắn của dữ liệu có dạng ngẫu nhiên, thường có độ dài 16-32 byte, nằm giữa các mảnh dữ liệu khác có entropy thấp hơn nhiều.

#### 2.2.2. Nhận dạng các khóa trong bộ nhớ

Để trích xuất các khóa mã hóa từ bộ nhớ, trước tiên chúng phải được xác định. Một số nhà nghiên cứu đã khám phá ra rằng các khóa mã hóa trong bộ nhớ có cấu trúc phức tạp hơn so với những gì được tin tưởng trước đây và đã đề xuất vài chiến lược để xác định các khóa này, các chiến lược này được thảo luận dưới đây.

Sử dụng phương pháp tìm kiếm với entropy cao như được đề xuất bởi **Shamir**. Vì đã biết rằng dữ liệu khóa có entropy cao hơn so với dữ liệu không phải khóa, một cách để xác định 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 đã biết trong bộ nhớ như lịch trình khóa (key schedule) được đặc thù cho một số loại mã hóa. Những mẫu này cũng có thể bao gồm các vị trí bộ nhớ đã biết, độ 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

Dựa trên việc xem xét các mẫu ransomware hiện tại, đã được xác định rằng đa số các ransomware mã hóa hiện đại đều có tính chất lai (**hybrid**). Khóa công khai của mã hóa không đối xứng được gửi kèm với ransomware, trong khi khóa bí mật được giữ bởi kẻ tấn công. Vì khóa bí mật của mã hóa không đối xứng không bao giờ có mặt trên máy tính, bài báo này sẽ tập trung vào việc xác định khóa được sử dụng trong giai đoạn mã hóa đối xứng của quá trình thực thi của ransomware, trong đó trong đa số các trường hợp đó là khóa *Advanced Encryption Standard (AES)*.

AES là một thuật toán mật mã dựa trên mạng thay thế-permuted (*Substitution-Permutation network*) hoạt động trên các khối 128 bit và có thể sử dụng các khóa có độ 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 được mà không có khóa chính xác.

Hệ thống mật mã khóa đối xứng hiện đại được xây dựng bằng cách áp dụng lặp đi lặp lại một hàm đơn giản hơn, trong đó thực hiện một số "vòng" hoặc lần lặp. Từ khóa chính, một hàm phái sinh sẽ tạo ra các khóa con khác được sử dụng trong mỗi vòng.

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	

**Fig. 1.** AES Key and Key Schedule [26].

Điều này được gọi là thuật toán lập lịch khóa và đã được nhiều nhà nghiên cứu chỉ ra rằng thông tin này cần có trong bộ nhớ. Kiến thức về hệ thống mật mã này có thể được sử dụng để tìm kiếm mẫu khóa này trong bộ nhớ. Tiêu chí tìm kiếm là có một mối quan hệ toán học giữa khóa chính và các khóa con. Lịch trình khóa thường được tính toán trước, trong một sự đánh đổi giữa bảo mật và hiệu suất, và được lưu trong bộ nhớ trong quá trình mã hóa/giải mã. Một ví dụ về một khóa AES 128 bit trống (toàn bộ giá trị là 0) và lịch trình khóa tương ứng, được trích xuất từ bộ nhớ, được hiển thị trong



Hình 1.

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

Đã phát triển một số công cụ sử dụng hiện tượng này để xác định khóa AES trong bộ nhớ như AESFinder, Volatools, interrogate và Findaes, tuy nhiên không tìm thấy nghiên cứu nào cho thấy chúng đã được sử dụng để phân tích ransomware mật mã cụ thể.

#### **2.2.4. Phương pháp**

Một số công trình nghiên cứu trong lĩnh vực này đã sử dụng cùng phương pháp thực nghiệm chung. Đầu tiên, chạy chương trình đang được điều tra trong một môi trường ảo, sau đó sử dụng các công cụ từ môi trường ảo để ghi lại bộ nhớ hệ thống một cách an toàn và đáng tin cậy. Sau khi hoàn thành việc ghi lại bộ nhớ đủ số lần cần thiết, chúng được phân tích.

Nhiều nhà nghiên cứu đã thành công trong việc trích xuất các khóa mã hóa thông qua việc khám phá thông tin mật mã trong bộ nhớ tạm thời.

Một điều cần nhớ khi áp dụng phương pháp này vào các mẫu ransomware được phân tích trong báo cáo này là các mẫu này thực hiện một số bước trước khi thực sự bắt đầu mã hóa dữ liệu trên hệ thống của nạn nhân. Vì vậy, khóa mã hóa của ransomware sẽ không có mặt ngay lập tức trong bộ nhớ khi chương trình bắt đầu thực thi và xác định thời điểm ghi lại bộ nhớ trở nên quan trọng đối với thành công của các thí nghiệm.

Lý tưởng nhất là ghi lại bộ nhớ trong quá trình ransomware đang mã hóa các tệp tin. Quá trình mã hóa có thể kéo dài từ vài phút đến một vài giờ, và chương trình có thể thực hiện quản lý khóa tốt khi hoàn thành mã hóa bằng cách xóa khóa khỏi bộ nhớ, vì vậy xác định thời điểm ghi lại bộ nhớ là rất quan trọng. Phương pháp mô tả sử dụng phân tích bộ nhớ tạm thời kết hợp với quan sát kinh nghiệm thay vì tập trung vào các cuộc gọi API cụ thể mà ransomware có thể sử dụng.

### **3. Thiết kế**

#### **3.1. Thiết kế môi trường**

Để thực hiện các thử nghiệm ransomware có giá trị và thực tế, máy nạn nhân cần giống một máy tính thực tế nhất có thể, với thông tin riêng tư và bí mật đã được loại bỏ, và lý tưởng nhất là cách ly hoàn toàn khỏi internet. Vậy nên ta sẽ quyết định sử dụng phần mềm ảo hóa VirtualBox của Oracle để tạo môi trường thử nghiệm cho dự án này. Khi thiết kế môi trường thử nghiệm, một yêu cầu quan trọng là phải tạo một môi trường thực tế nhất có thể. Môi trường thử nghiệm được thiết kế với ba máy ảo. Hai trong số các máy ảo này là các máy nạn nhân thử nghiệm. Chỉ có một máy hoạt động tại một thời điểm và chỉ trên đó ransomware được thực thi.

Trong hầu hết các thử nghiệm, cần ít nhất một hệ thống khác để cung cấp các dịch vụ mạng hỗ trợ cho máy nạn nhân. Việc từ chối truy cập mạng cho mẫu đang được phân tích sẽ dẫn đến quan sát không đầy đủ về hành vi của malware. Do đó, một kỹ thuật phổ biến là sử dụng một máy ảo thứ ba trên mạng ảo để cung cấp các dịch vụ mạng cho các máy nạn nhân như DNS, IRC, HTTP và xử lý các yêu cầu có thể được gửi từ phần malware về máy chủ điều khiển (C&C).

Các máy này được kết nối thông qua mạng ảo 'host-only' và chỉ có các máy này trên mạng ảo đó. Cấu hình này đảm bảo sự cách ly hoàn toàn của các máy khách này khỏi máy chủ và do đó cách ly các kết nối mạng của máy chủ. Máy chủ vật lý là một laptop không kết nối với mạng bên ngoài, cung cấp một lớp cách ly thứ hai. Việc có cả truy cập mạng và nội dung trên máy ảo nạn nhân đóng vai trò

quan trọng trong việc làm cho phần malware nhìn như một máy tính thực sự và khiến phần malware hoạt động bình thường. Ngoài ra, các chính sách kiểm soát như tường lửa và phần mềm chống virus cũng được triển khai trên máy chủ.

### 3.2. Thiết kế thử nghiệm

Ở mức trừu tượng nhất, các thí nghiệm được thiết kế như sau: Một mẫu ransomware được thực thi trong một môi trường ảo. Trong quá trình thực thi này, các bản sao của bộ nhớ khả biến của máy tính được lấy. Các công cụ pháp chứng sử dụng để phân tích các tệp bộ nhớ này nhằm tìm ra khóa mã hóa. Các khóa được tìm thấy sau đó được sử dụng để giải mã các tệp bị mã hóa trong quá trình thực thi của ransomware, từ đó 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 con riêng biệt. Kết quả của từng thí nghiệm này sẽ được sử dụng để xác nhận hoặc phủ định giả thuyết rằng các kỹ thuật pháp chứng thời gian thực có thể được sử dụng để giảm thiểu một cuộc tấn công ransomware. Mỗi vòng thí nghiệm bắt đầu bằng phiên bản mới của một máy ảo phản ánh một máy trạm thực tế được khởi động và một ví dụ về ransomware đang được điều tra được thực thi. Ba thử nghiệm sau được thực hiện:

#### 3.2.1. Thử nghiệm phần 1 – Xác định khóa trong bộ nhớ

Một bản sao bộ nhớ từ máy tính đang thực thi ransomware đã được thu thập. Trong quá trình chọn mẫu ransomware cho các thí nghiệm này thì AES được sử dụng cho phần đối xứng của quá trình mã hóa. Vì vậy, sau khi hoàn thành việc thu thập bản sao bộ nhớ, nó đã được phân tích bằng các công cụ pháp y thời gian thực để xác định xem khóa AES có thể được khám phá hay không. Đặc biệt chú ý đến các vùng bộ nhớ có độ ngẫu nhiên cao. Một biểu đồ đồ họa về điều này và thí nghiệm tiếp theo được hiển thị trong Hình 2.

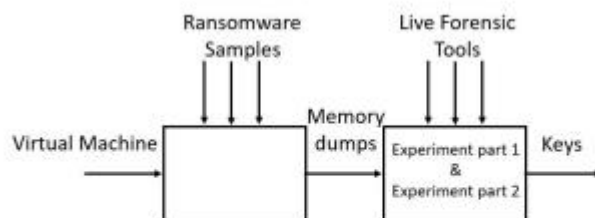
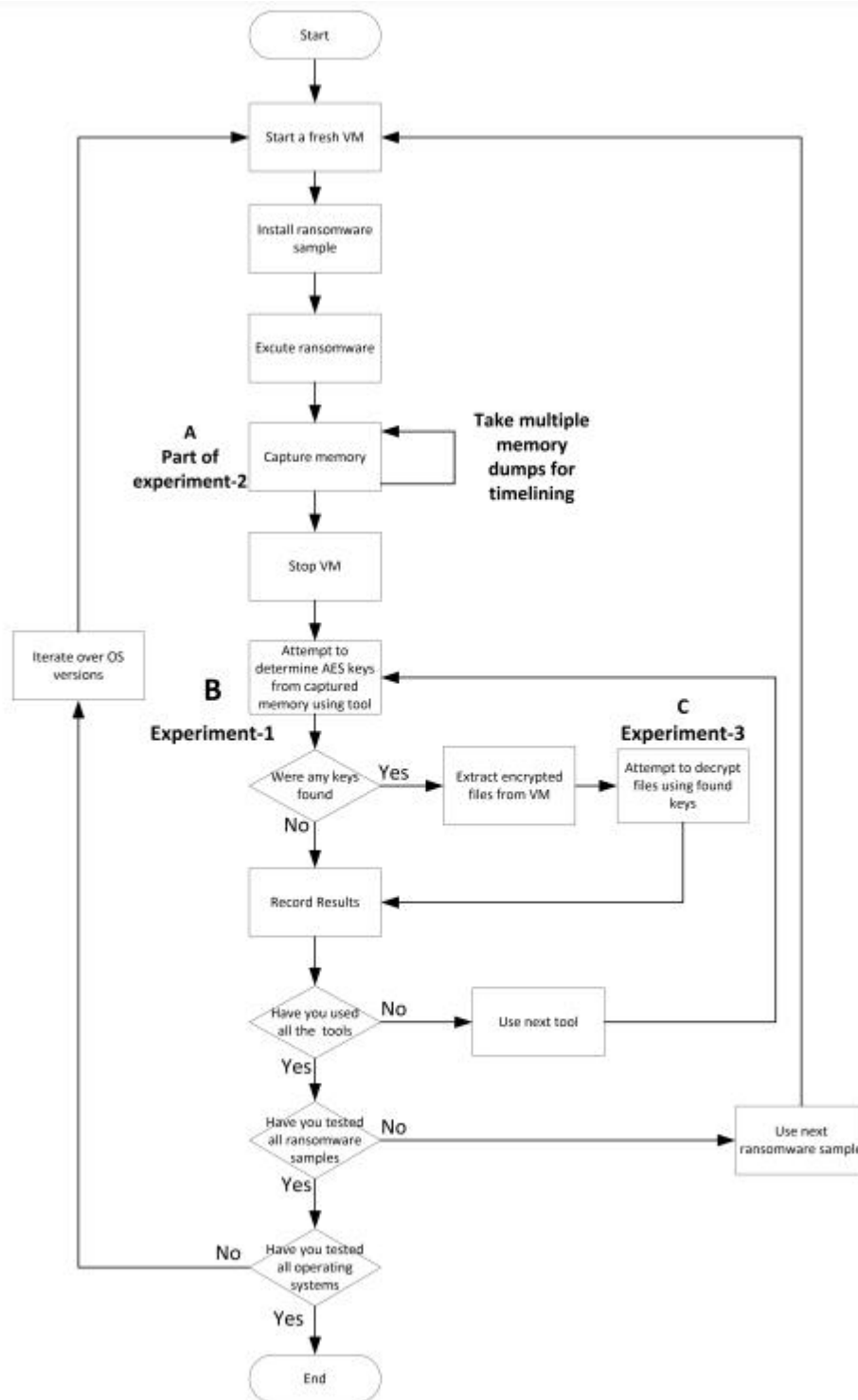


Fig. 2. Overview of Experiment part 1 & Experiment part 2.

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

Một bản sao bộ nhớ được lấy vào các khoảng thời gian đều đặn trong suốt vòng đời thực thi hoàn chỉnh của ransomware để xác định thời điểm khóa được tải vào bộ nhớ và trong bao lâu nó ở đó. Điều này hỗ trợ việc thực hiện phần thử nghiệm 1 bằng cách xác định thời điểm thực thi điểm A trong Hình 4. Kết quả của thí nghiệm này là một biểu đồ thời gian xấp xỉ cho việc thực thi của ransomware.





**Fig. 4. Experiment flow.**

### 3.2.3. Thử nghiệm phần 3 – Xác thực các khóa tìm thấy

Nếu có bất kỳ khóa nào được tìm thấy trong phần thử nghiệm 1, chúng sẽ được kiểm tra để xác định xem chúng có thể giải mã bất kỳ tệp điều khiển nào đã bị mã hóa bởi ransomware hay không. Như đã 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 đồ về điều này được hiển thị trong

Hình 3.

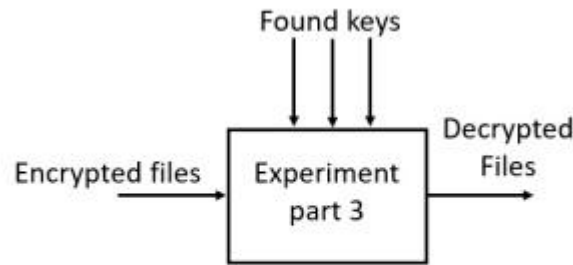


Fig. 3. Overview of Experiment part 3.

#### 3.2.4. Thử nghiệm tổng hợp

Mặc dù các thí nghiệm được thiết kế trong cuộc điều tra này có những điểm tương đồng đáng kể với công việc trước đó 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 đã thử nghiệm nhiều công cụ trích xuất khóa trên nhiều hệ điều hành và kiểm tra các khóa đã tìm thấy để xác nhận rằng chúng giải mã thành công các tệp điều khiển. Ngoài ra, nhiều bản sao bộ nhớ đã được lấy trong suốt thí nghiệm để tạo ra một biểu đồ thời gian cho việc thực thi của ransomware. Một biểu đồ về toàn bộ thí nghiệm được thể hiện trong Hình 4 và một số bước chính được mô tả như sau:

**Lặp lại các phiên bản hệ điều hành (Iterate over Operating Systems versions):** Khi thảo luận về tính thực tế trong thí nghiệm, nên tránh thực hiện thí nghiệm chỉ trên một hệ điều hành và rút ra những kết luận tổng quát. Vì vậy, các thí nghiệm được thực hiện trong nghiên cứu này sẽ được tiến hành trên hai phiên bản hệ điều hành Windows phổ biến nhất hiện nay.

**Khởi động một máy ảo mới (Start a fresh VM):** Kết quả chỉ có thể so sánh được nếu mỗi mẫu được thực thi trong một môi trường giống nhau, vì vậy một máy ảo mới được khởi động vào đầu mỗi thí nghiệm.

**Cài đặt Ransomware (Install Ransomware):** Mẫu được kiểm tra được trích xuất từ file nén và được chuẩn bị để thực thi.

**Thực thi Ransomware (Execute Ransomware):** Mẫu ransomware được chọn được thực thi từ dòng lệnh.

**Thu thập bộ nhớ (Capture Memory):** Thời điểm lấy bản sao bộ nhớ làm việc của máy ảo được xác định dựa trên kết quả của phần thử nghiệm 2. 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 và các kỹ thuật tương tự khác.

**Dừng máy ảo (Stop VM):** Sau khi đã thực hiện đủ số lần thu thập bộ nhớ yêu cầu, hoặc nếu ransomware đã hoàn thành, máy ảo của nạn nhân sẽ được dừng lại.

**Cố gắng xác định các khóa AES từ bộ nhớ đã thu (Attempt to determine AES keys from captured memory):** Phân tích được thực hiện trên các mẫu bộ nhớ đã thu với mục tiêu xác định các khóa AES đề xuất, sử dụng ba công cụ dưới đây:

1. findaes - Công cụ được phát triển bởi Kornblum dựa trên nghiên cứu của Trenholme. Công cụ này sẽ cố gắng tìm các khóa sử dụng cấu trúc lịch trình khóa AES.
2. interrogate- Công cụ được phát triển bởi Maartmann-Moe cũng dựa trên nghiên cứu của Trenholme.
3. RansomAES - Công cụ lại được phát triển bởi tác giả kết hợp logic từ Volatility Framework cùng

với logic từ findaes, mục đích cải thiện độ chính xác và hiệu suất của việc phát hiện khóa.

**Trích xuất các tệp tin đã mã hóa ( Extract Encrypted Files ):** Một tập hợp các tệp tin điều khiển có nội dung đã biết được đặt trên máy nạn nhân trước khi ransomware được thực thi. Sau khi ransomware đã thực thi, những tệp tin này được phân tích để xác định xem chúng đã bị mã hóa hay chưa.

**Cố gắng giải mã tệp tin ( Attempt to decrypt files ):** Nếu có bất kỳ khóa đề cử nào được phát hiện trong giai đoạn phân tích của thí nghiệm, chúng sẽ được sử dụng để thử giải mã các tệp tin điều khiển đã được trích xuất từ máy ảo nạn nhân. Vector khởi tạo AES (IV) cần thiết để giải mã được chứa trong tệp tin đã mã hóa và được sử dụng kết hợp với các khóa đề cử để giải mã tệp tin.

## 4. Triển khai và Kết quả

### 4.1. Lựa chọn mẫu ransomware

Dựa trên kết quả phát hiện trong việc nghiên cứu các cuộc tấn công ransomware gần đây, người ta quyết định chọn ba mẫu ransomware sau đây để phân tích. Tất cả đều sử dụng AES cho phần đối xứng của mã hóa.

**NotPetya:** Cuộc tấn công ransomware này được coi là cuộc tấn công gây thiệt hại lớn nhất từ trước đến nay.

**BadRabbit:** Một bản chuyển thể của dòng ransomware NotPetya xuất hiện vào năm 2018.

**Phobos:** Một trong những mẫu ransomware gần đây nhất và cũng là một trong những mẫu phổ biến nhất trong năm 2019.

**Table 1.** Ransomware Samples

Name	SHA256 Checksum
NotPetya	027cc450ef5f8c5f653329641ec1fed91f694e0d229928963b30f6b0d7d3a745
Bad Rabbit	630325cac09ac3fab908f903e3b00d0dadd5fdaa0875ed8496fcb97a558d0da
Phobos	a91491f45b851a07f91ba5a200967921bf796d38677786de51a4a8fe5ddeafd2

Chi tiết cụ thể về các mẫu ransomware được sử dụng được mô tả trong Bảng 1. Ba mẫu ransomware được chọn giống 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 mã hóa.

#### 4.1.1. Một số loại ransomware khác

Các mẫu ransomware sau đây ban đầu được cân nhắc để làm mẫu cho thử nghiệm, tuy nhiên sau đó bị loại bỏ.

- WannaCry. Chúng này sử dụng các khóa AES duy nhất cho mỗi tệp được mã hóa. Sau khi tiến hành nhiều cuộc kiểm tra bộ nhớ có được trong quá trình thực thi phần mềm độc hại này bằng tất cả các công cụ điều tra trực tiếp, không tìm thấy khóa AES nào có thể khôi phục được.

- Cerber. Phần mềm ransomware này dường như không sử dụng mã hóa AES.

- Lucky/nmar. Mẫu của phần mềm ransomware này yêu cầu quyền truy cập vào Internet để có thể tải xuống các mô-đun mã hóa tệp vì chúng không được phân phối cùng với mẫu ban đầu. Các dịch vụ do máy ảo Debian cung cấp không thể làm cho phần mềm ransomware thực thi bình thường và việc cho phép truy cập mạng bên ngoài này được coi là quá rủi ro.

- Satan, SamSam và GrandCab. Không thể kích hoạt các mẫu ransomware này để mã hóa bất kỳ

quyền kiểm soát nào của tệp hoặc hiển thị thông báo đòi tiền chuộc.

## 4.2. Cấu hình máy thử nghiệm

Laptop được sử dụng để thử nghiệm không có kết nối mạng bên ngoài và để phòng ngừa thêm, máy đã được đặt ở chế độ trên máy bay và đã tắt wifi card. Máy vật lý này được gọi là máy chủ vì nó lưu trữ môi trường ảo trong VirtualBox. Ba máy khách được xác định trong môi trường ảo chạy trên máy laptop. Hai máy nạn nhân có hệ điều hành khác nhau, được sử dụng để kiểm tra hành vi 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 kỹ thuật kết nối ‘host-only’ được đề xuất, tạo ra một mạng LAN ảo riêng biệt cung cấp sự cách ly và ngăn chặn các máy khách. Mạng LAN ảo và máy khách được chạy trong hai cấu hình riêng biệt tùy thuộc vào phiên bản Windows nào đang được thử nghiệm.

**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

Các công cụ sau đã được sử dụng trong quá trình cấu hình của máy Debian. Đầu tiên, fakedns.py được sử dụng để cung cấp DNS dịch vụ vào mạng và INetSim - Được coi là công cụ miễn phí tốt nhất để cung cấp mô phỏng dịch vụ mạng giả tạo ảo tưởng về một mạng giả thực tế phần mềm độc hại có thể tương tác nên được yêu cầu.

## 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\lsaas.bin.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**

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
```

```
interrogate -a aes -k 128 <filename>.elf
```

```
ransomaes -p <ransomeware pid> -t Win7SP0x86 <filename>.elf
```

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 dòng 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. Kết quả và thảo luận**

Chương này được chia thành các phần riêng biệt, một phần cho mỗi mẫu ransomware được thử nghiệm. Mỗi phần này thảo luận về kết quả của ba thí nghiệm được tiến hành.

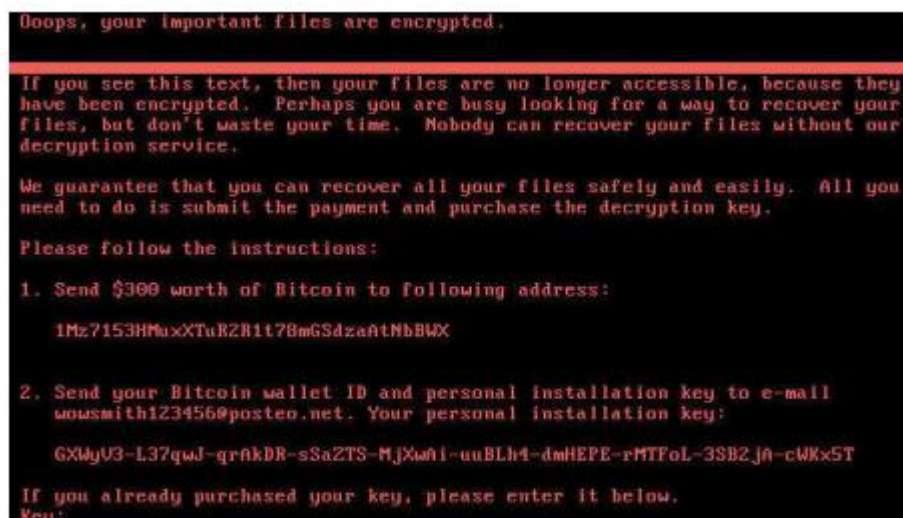
### **5.1. NotPetya**

Việc thực thi ransomware tuân theo mô tả được cung cấp bởi (Berry et al., 2017; Vipre Security, 2017). Các bước chính được mô tả như sau:

1. Thêm tính duy trì lâu dài và thu thập thông tin đăng nhập của người dùng.
2. Quét máy tính để tìm các tệp cần mã hóa. Mẫu này bỏ qua các tệp điều khiển có phần mở rộng '.txt'.
3. Mã hóa các tệp được xác định bằng cách sử dụng mã hóa AES với cùng một khóa AES được sử dụng cho tất cả các tệp. Điều thú vị là tên tệp hoặc siêu dữ liệu tệp không thay đổi khi tệp được mã hóa.
4. Cố gắng di chuyển ngang sang các máy khác, tuy nhiên không có bằng chứng thực tế nào về điều này được phát hiện.
5. Sau 1 giờ, tự động khởi động lại máy.
6. Hiện thị đầu ra lệnh chkdsk giả, trong khi mã hóa Master Boot Record.



7. Sau khi chkdsk giả hoàn tất, máy tự động khởi động lại một lần nữa.
8. Sau khi khởi động lại, thông báo ransomware được hiển thị trong Hình 5.



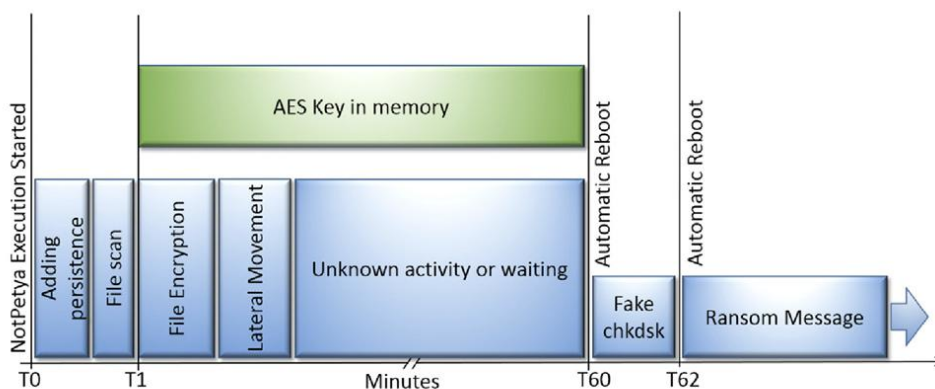
**Fig. 5. NotPetya Ransom Message.**

#### 5.1.1. Thí nghiệm phần 1 – Xác định khóa trong bộ nhớ

Cả ba công cụ pháp chứng thời gian thực được sử dụng để kiểm tra bộ nhớ đều có thể xác định thành công khóa AES trong bộ nhớ của quy trình ransomware.

#### 5.1.2 Thí nghiệm phần 2 – Tạo biểu đồ thời gian khóa

Tổng cộng 15 bản dump bộ nhớ đã được thu thập và phân tích để xác định xem chúng có chứa khóa AES hay không. Kết quả cho thấy khóa có sẵn trong vòng 2 phút sau khi ransomware bắt đầu hoạt động và duy trì trong bộ nhớ cho đến khi máy tự động khởi động lại bởi ransomware sau 60 phút. Khóa không tồn tại sau khi khởi động lại và không thể khôi phục từ bộ nhớ sau đó. Hình 6 dưới đây minh họa một số hoạt động của ransomware và khả năng truy cập khóa.



**Fig. 6. NotPetya timeline.**

Biểu đồ thời gian này tương quan tốt với các phát hiện của các nhà nghiên cứu khác (Berry et al., 2017; Vipre Security, 2017). Tuy nhiên, không có nghiên cứu nào phân tích chính xác thời điểm và thời gian khóa thực tế tồn tại trong bộ nhớ. Sử dụng sơ đồ này, có thể thấy rõ ràng rằng khóa tồn tại trong tổng số 59 phút, chiếm phần lớn thời gian thực thi của ransomware. Ngoài ra, không có biểu diễn đồ họa tương tự nào về dòng thời gian ransomware được tìm thấy trong tài liệu.



### 5.1.3. Thí nghiệm phần 3 - Xác thực khóa đã tìm thấy

Khi ransomware NotPetya mã hóa một tệp, 16 byte đầu tiên của tệp sẽ được ghi đè bằng giá trị Vector Khởi tạo (IV) AES (Sood and Hurley, 2017). Một chương trình đã được phát triển để đầu tiên đọc giá trị IV từ tệp được mã hóa, sau đó sử dụng nó cùng với khóa được tìm thấy trong thí nghiệm phần 1 để giải mã nội dung tệp.

Sử dụng kỹ thuật này, các tệp điều khiển được trích xuất pdf, doc, docx, xls vàxlsx đã được khôi phục thành công bằng cùng một khóa AES. Mỗi loại tệp này yêu cầu các tiêu đề khác nhau được chèn vào và một số tệp yêu cầu một số byte được xóa khỏi cuối tệp.

## 5.2. Bad Rabbit

Việc thực thi ransomware này tuân theo mô tả được cung cấp bởi (Malwarebytes LABS, 2017; Mamedov et al., 2018; Perekalin, 2018) và tương tự như các bước được sử dụng bởi ransomware NotPetya. Các bước chính là:

1. Thêm tính duy trì lâu dài.
2. Quét máy tính để tìm các tệp cần mã hóa. Mẫu này dường như bỏ qua các tệp điều khiển có phần mở rộng tệp 'txt' và 'jpg'.
3. Mã hóa các tệp được xác định bằng cách sử dụng mã hóa AES.
4. Sau 14 phút, một tệp ghi chú tiền chuộc được tạo trên ổ đĩa C.
5. Một phút sau, máy tự động khởi động lại.
6. Máy khởi động lại với màn hình nền Windows có vẻ bình thường. Tuy nhiên, ở chế độ nền, MBR đang được mã hóa.
7. 22 phút sau khi ransomware bắt đầu thực thi, máy tự động khởi động lại một lần nữa.
8. Sau khi khởi động lại, thông báo ransomware được hiển thị trong Hình 7 và người dùng bị ngăn truy cập vào cài đặt Windows của họ.

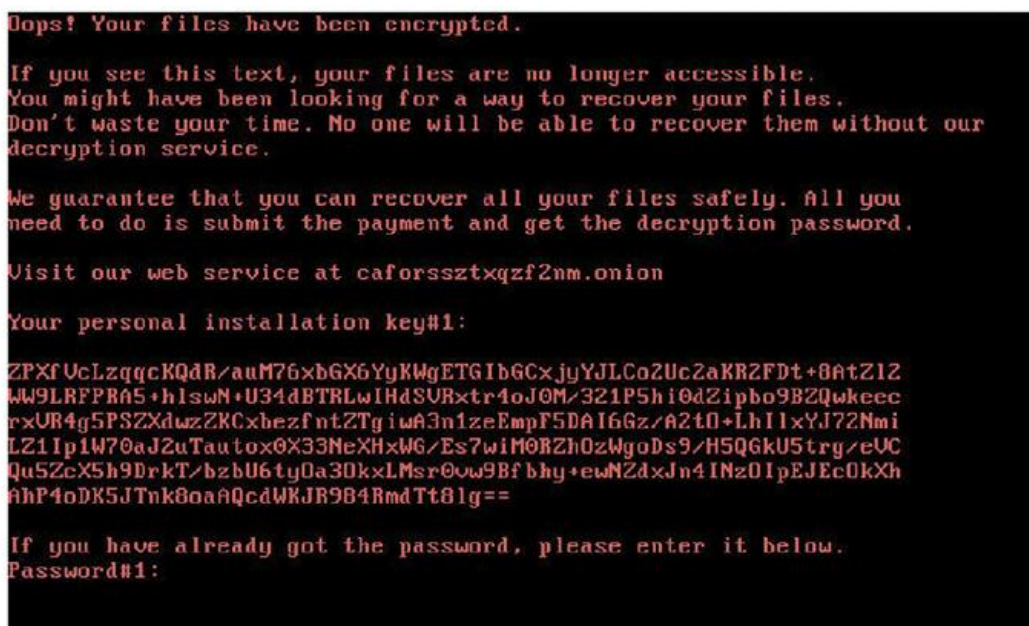


Fig. 7. Bad rabbit ransom note.

### 5.2.1 Thí nghiệm phần 1 – Xác định khóa trong bộ nhớ

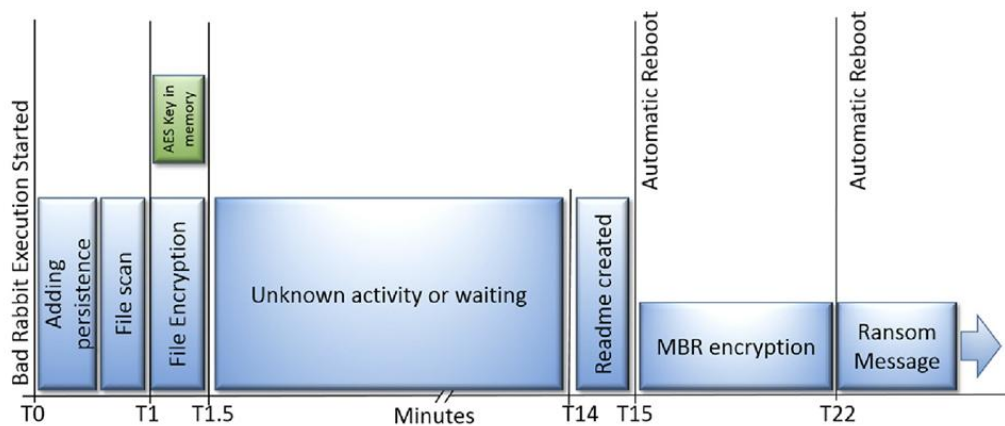
Cả ba công cụ pháp chứng thời gian thực được sử dụng để kiểm tra bộ nhớ đều có thể xác định khóa AES trong bộ nhớ của quy trình ransomware.

### 5.2.2. Thí nghiệm phần 2 - Tạo dòng thời gian khóa

Lại một lần nữa, tổng cộng 15 bản dump bộ nhớ đã được thu thập và phân tích để xác định xem chúng có chứa khóa AES hay không. Kết quả cho thấy khóa có sẵn trong vòng 1 phút sau khi ransomware bắt đầu thực thi và chỉ tồn tại trong bộ nhớ trong khi mã hóa đang được thực hiện. Khoảng thời gian này xấp xỉ 30 giây. Khóa không xuất hiện lại ngay cả sau khi máy khởi động lại. Hình 8 dưới đây minh họa một số hoạt động của ransomware và khả năng truy cập khóa.

Dòng thời gian này khớp với mô tả được đưa ra bởi (Malwarebytes LABS, 2017). Sử dụng sơ đồ dòng thời gian, có thể thấy rõ ràng rằng khóa chỉ tồn tại trong 30 giây, ngắn hơn nhiều so với thời gian thực thi tổng thể và ngắn hơn nhiều so với ransomware NotPetya. Khóa có thể có mặt vào những thời điểm khác, nhưng bị bỏ lỡ do thời gian lấy mẫu.

*S.R. Davies et al. / Forensic Science International: Digital Investigation 33 (2020) 300979*



**Fig. 8.** Bad rabbit timeline.

### 5.2.3. Thí nghiệm phần 3 - Xác thực khóa đã tìm thấy

Tập được mã hóa sử dụng định dạng giống như ransomware NotPetya và các bước được mô tả trong phần 5.1.3 có thể được sử dụng để giải mã các tập được mã hóa bằng ransomware Bad Rabbit. Sử dụng kỹ thuật này, các tệp pdf, doc, docx, xls vàxlsx đã được khôi phục thành công bằng cùng một khóa AES.

## 5.3. Phobos

Việc thực thi ransomware tuân theo mô tả được cung cấp bởi (Issa, 2019). Các bước chính là:

1. Thêm tính duy trì lâu dài và thu thập thông tin đăng nhập.
2. Quét máy tính để tìm các tệp cần mã hóa. Mẫu này đã mã hóa tất cả các tệp điều khiển.
3. Mã hóa các tệp được xác định bằng cách sử dụng mã hóa AES với khóa AES có vẻ như được sử dụng cho tất cả các tệp được mã hóa ban đầu. Tên tệp cũng đã được thay đổi.
4. Sau khoảng 2 phút, ghi chú tiền chuộc được hiển thị trong Hình 9.
5. Thật thú vị, máy vẫn hoạt động ở một mức độ nào đó và bất kỳ tệp mới nào được tạo đều được mã hóa bằng một khóa AES khác. Nhà nghiên cứu không thể khám phá ra khóa AES thứ cấp này. Đây có thể là một lĩnh vực nghiên cứu tiếp theo, người ta tin rằng các khóa này cũng nên có mặt.
6. Máy không tự động khởi động lại. Nếu người dùng khởi động lại máy, thì cùng một thông báo ransomware sẽ được hiển thị.



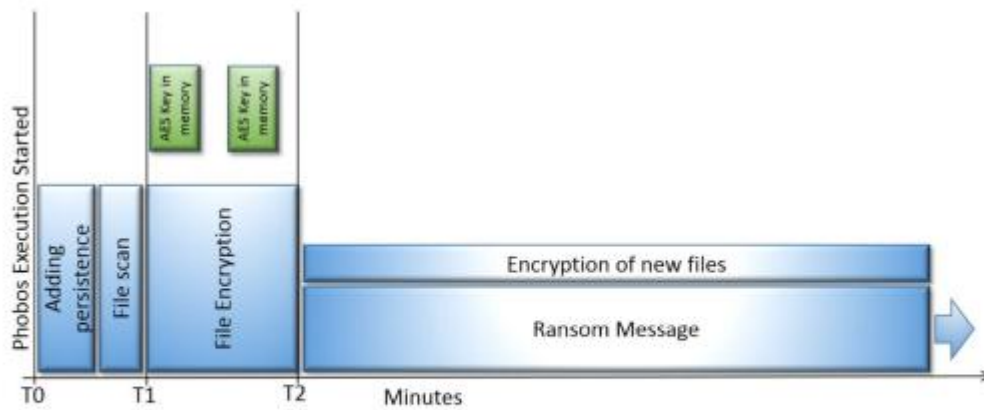
**Fig. 9. Phobos Ransom Note.**

### 5.3.1. Thí nghiệm phần 1 - Xác định khóa trong bộ nhớ

Cả ba công cụ pháp chứng thời gian thực được sử dụng để kiểm tra bộ nhớ của quy trình ransomware đều có thể xác định khóa AES 256 bit được ransomware sử dụng để mã hóa các tệp.

### 5.3.2. Thí nghiệm phần 2 - Tạo dòng thời gian khóa

Tương tự, tổng cộng 15 bản dump bộ nhớ đã được thu thập và phân tích để xác định xem chúng có chứa khóa AES hay không. Kết quả cho thấy khóa ban đầu có sẵn trong vòng 1 phút sau khi ransomware bắt đầu thực thi. Cùng một khóa AES đã được tải vào bộ nhớ và xóa nhiều lần trong quá trình mã hóa ban đầu của máy. Khóa đã bị xóa khi thông báo tiền chuộc được hiển thị. Ransomware tiếp tục mã hóa bất kỳ tệp mới nào được tạo, sử dụng một khóa AES khác. Một số nỗ lực không thành công đã được thực hiện để cố gắng và lấy khóa thứ cấp này từ bộ nhớ. Cũng như với hai mẫu ransomware trước đó, không có khóa AES nào tồn tại sau khi máy khởi động lại. Hình 10 dưới đây minh họa một số hoạt động của ransomware và khả năng truy cập khóa. Dòng thời gian này tương quan tốt với mô tả được đưa ra bởi (Issa, 2019; Panda, 2017). Sử dụng sơ đồ này, có thể dễ dàng nhận thấy rằng cùng một khóa AES có mặt trong bộ nhớ vào một số thời điểm khác nhau. Cũng cần lưu ý rằng có thể có những trường hợp sự hiện diện của khóa bị bỏ lỡ do thời gian lấy mẫu. Không có biểu diễn đồ họa tương tự nào về dòng thời gian ransomware được tìm thấy trong tài liệu, hình minh họa trong Hình 10 được tạo bởi tác giả.



**Fig. 10. Phobos Timeline.**

### 5.3.3. Thí nghiệm phần 3 - Xác thực khóa đã tìm thấy

Thông tin bổ sung được thêm vào cuối tệp đã được mã hóa bởi ransomware Phobos. Thông tin bổ sung này bao gồm một số phần đệm, theo sau là giá trị IV AES được cho là, sau đó là 128 byte giống nhau cho tất cả các tệp được mã hóa. Mô tả chi tiết về tệp được mã hóa xuất hiện trong công trình của Issa (2019), người đưa ra giả thuyết rằng khối 128 byte này có thể là khóa bất đối xứng được mã hóa. Cũng có một chuỗi cố định ở cuối khối, trong trường hợp này là 'LOCK960' nhưng các phiên bản khác của Phobos đã được quan sát thấy với các từ khóa khác nhau như 'DAT260'. Một chương trình đã được phát triển để đầu tiên đọc giá trị IV từ tệp được mã hóa, sau đó sử dụng nó cùng với khóa được tìm thấy trong thí nghiệm phần 1 để giải mã tệp. Sử dụng kỹ thuật này, các tệp pdf, doc, docx, xls vàxlsx đã được khôi phục thành công bằng cùng một khóa AES.

## 6. Kết quả Demo

Quá trình thử nghiệm được thực hiện trên máy ảo Windows 10.

Các công cụ sử dụng:

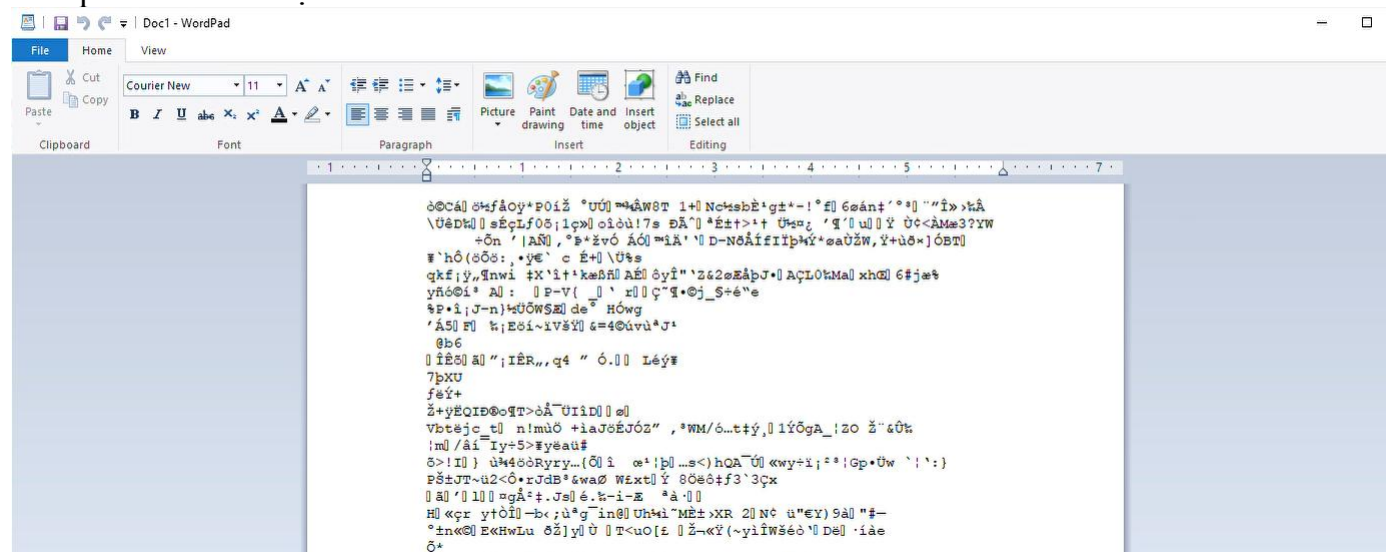
- Công cụ DumpIT để lưu lại bộ nhớ của máy tính khi thực thi virus ransomware.
- Sử dụng volatility để phân tích và khôi phục từ file bị mã hóa.
- Công cụ aesfindkey để tìm ra khóa aes.

### 6.1. Notpetya

Thực thi virus Notpetya trên Windows 10.



Kết quả các file đã bị mã hóa:



Khóa sẽ xuất hiện sau 2 phút kể từ khi virus được thực thi, sau đó ta tiến hành dumpit để lưu lại bộ nhớ và tìm kiếm key.



```
C:\Users\hp\Downloads\Dump\DumpIt.exe
DumpIt - v1.3.2.20110401 - One click memory memory dumper
Copyright (c) 2007 - 2011, Matthieu Suiche <http://www.msuiche.net>
Copyright (c) 2010 - 2011, MoonSols <http://www.moonsols.com>

Address space size:      1073741824 bytes ( 1024 Mb)
Free space size:        4272957440 bytes ( 40750 Mb)

* Destination = \??\C:\Users\hp\Downloads\Dump\DESKTOP-IJK25H1-20240602-210333.raw

--> Are you sure you want to continue? [y/n] y
+ Processing... Success.
```

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.

Notpetya1	6/3/2024 4:02 AM	RAW File	1,048,576 KB
Notpetya2	6/3/2024 4:03 AM	RAW File	1,048,576 KB
Notpetya3	6/3/2024 4:07 AM	RAW File	1,048,576 KB
Notpetya4	6/3/2024 4:10 AM	RAW File	1,048,576 KB
Notpetya5	6/3/2024 4:12 AM	RAW File	1,048,576 KB

Sau đó tiếp theo sẽ reboot lại máy Windows và kiểm tra kết quả.

Repairing file system on C:

The type of the file system is NTFS.

One of your disks contains errors and needs to be repaired. This process may take several hours to complete. It is strongly recommended to let it complete.

WARNING: DO NOT TURN OFF YOUR PC! IF YOU ABORT THIS PROCESS, YOU COULD DESTROY ALL OF YOUR DATA! PLEASE ENSURE THAT YOUR POWER CABLE IS PLUGGED IN!

CHKDSK is repairing sector 102336 of 239072 (42%)

Quá trình này virus Notpetya 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 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](mailto:wowsmith123456@posteo.net). Your personal installation key:

PFD8KN-zPHTL6-3waLZX-5YDfne-y4YF8x-2uEipt-hLnPJw-QW7QRC-9v1W8Y-RfJG6D

If you already purchased your key, please enter it below.

Key:

Bước tiếp theo ra đi tìm key aes từ các file đã dump ra được.

Sử dụng lệnh:

*aeskeyfind <tên file>*

Tổng cộng các key tìm được từ 5 file:

```
1 650a41c25760a52545cb5fd668f9d828
2 1f2a8ec34965f892c7e19cc29a4345db
3 cc210b9173cc8c07ef377c9105b767a5
4 dd42d90101d32b411a1d1fa91b0a14cc47e84340a470c7e307a352e471f42036
5 250a43ac5f702f9fd384d000a34f8051
6 2528f550849b80d293e33a9266b29d80
7 3c9b69152d4c7cb5829d1ffccdbd599d6cb588e756f60872000972f5062b0d4d
8 2d747ad795956d060cfa8960792bd48b39c665ee7484c4fce3e7d5f47f80c71e
9 61e790d6e2c1461c800cb2230f82874f3e55bbb0c71375b054c238caee446e48
10 29e1bc3b55314e6d64defbeecaf61c04
11
12 557b910b18d7b80526e284785b36a27b
13 2a291d0d7358ec12090a06df30bdd3a1
14 650a41c25760a52545cb5fd668f9d828
15
16 5091d8349dc7153a74d79bd84a32d774742252dad7e557222f4e3c9533e95865
```

Sau khi đã có được các key này, ta thử tiến hành tìm kiếm file và khôi phục file.

Sử dụng volatility với plugins filescan để tìm file, câu lệnh như sau:

*python2 vol.py -f <tên file> --profile=<thông tin máy tính> filescan*

Lưu lại kết quả vào tệp txt sau đó tìm kiếm file cần tìm.



```

715 0x0000960f52f83700 16 0 RW— \Device\HarddiskVolume2\Users\hp\Documents\CNXHKH.docx
716 0x0000960f52f83a20 11 0 RW— \Device\HarddiskVolume2\Users\hp\Documents\Doc1.docx
717 0x0000960f52f83bb0 19 0 RW-rwd \Device\HarddiskVolume2\$_Directory
718 0x0000960f52f83d40 19 0 RW-rwd \Device\HarddiskVolume2\$_Directory
719 0x0000960f52f83ed0 19 0 RW-rwd \Device\HarddiskVolume2\$_Directory
720 0x0000960f52f84380 19 0 RW-rwd \Device\HarddiskVolume2\$_Directory
721 0x0000960f52f84510 19 0 RW-rwd \Device\HarddiskVolume2\$_Directory
722 0x0000960f52f84830 11 0 RW— \Device\HarddiskVolume2\Users\hp\Desktop\Doc1.docx
723 0x0000960f52f84ce0 19 0 RW-rwd \Device\HarddiskVolume2\$_Directory
724 0x0000960f52f84e70 19 0 RW-rwd \Device\HarddiskVolume2\$_Directory
725 0x0000960f52f85190 19 0 RW-rwd \Device\HarddiskVolume2\$_Directory
726 0x0000960f52f85320 19 0 RW-rwd \Device\HarddiskVolume2\$_Directory
727 0x0000960f52f85640 19 0 RW-rwd \Device\HarddiskVolume2\$_Directory
728 0x0000960f52f85960 14 0 RW— \Device\HarddiskVolume2\Users\hp\Desktop\Lựa chọn mẫu ransomware.docx
729 0x0000960f52f85af0 12 0 RW— \Device\HarddiskVolume2\Users\hp\Desktop\mau-bao-cao-do-an-uit-mau-bao-caotieu-luan-cua-truong-dh-cnnt.pdf
730 0x0000960f52f85c80 19 0 RW-rwd \Device\HarddiskVolume2\$_Directory
731 0x0000960f52f85e10 15 0 RW— \Device\HarddiskVolume2\Users\hp\Desktop\Tutorial.docx
732 0x0000960f52f86130 7 0 RW— \Device\HarddiskVolume2\Users\hp\Documents\1-s2.0-S0045790622001884-main.pdf
733 0x0000960f52f862c0 19 0 RW-rwd \Device\HarddiskVolume2\$_Directory
734 0x0000960f52f86900 32768 1 R--r-d \Device\HarddiskVolume2\Windows\System32\en-US\KernelBase.dll.mui
735 0x0000960f52f86a90 11 0 RW— \Device\HarddiskVolume2\Users\hp\Downloads\chapter11.docx
736 0x0000960f52f86c20 11 0 RW— \Device\HarddiskVolume2\Users\hp\Downloads\1-s2.0-S0045790622001884-main.pdf
737 0x0000960f52f86db0 12 0 RW— \Device\HarddiskVolume2\Users\hp\Documents\mau-bao-cao-do-an-uit-mau-bao-caotieu-luan-cua-truong-dh-cnnt.pdf
738 0x0000960f52f870d0 11 0 RW— \Device\HarddiskVolume2\Users\hp\Downloads\Dump\1-s2.0-S0045790622001884-main.pdf
739 0x0000960f52f87260 14 0 RW— \Device\HarddiskVolume2\Users\hp\Downloads\Dump\Lựa chọn mẫu ransomware.docx
740 0x0000960f52f87710 32768 1 R--rw- \Device\HarddiskVolume2\Windows\System32
741 0x0000960f52f878a0 16 0 RW— \Device\HarddiskVolume2\Users\hp\Downloads\Dump\Tutorial.docx

```

Xác định file và trích xuất file cần khôi phục bằng volatility 3, vì nhóm gặp lỗi khi sử dụng vol2.

```

1642 0x0000960f52c31da0 1 1 ——— \Device\DeviceApi\CMap1
1643 0x0000960f52c600c0 1 1 R--rw- \Device\HarddiskVolume2\Windows\WinSxS\amd64_microsoft.wind
1644 0x0000960f52c60250 16 0 R--r-- \Device\HarddiskVolume2\Users\hp\Desktop\chapter11.docx
1645 0x0000960f52c603e0 32763 1 R--r-- \Device\HarddiskVolume2\Windows\Registration\R0000000000006.
1646 0x0000960f52c60570 15 0 R--r-d \Device\HarddiskVolume2\Windows\System32\TaskSchdPS.dll

```

Trích xuất file với câu lệnh sau:

```
python3 vol.py -f <tên file> -o <tên thư mục lưu trữ file dump> windows.dumpfiles.DumpFiles --virtaddr <offset>
```

Kết quả:

```

(kali@kali) ~[~/Ransomware/Win10/volatility3]
$ sudo python3 vol.py -f Notpetya1.raw -o file windows.dumpfiles.DumpFiles --virtaddr 0x960f52c60250
Volatility 3 Framework 2.7.0
Progress: 100.0% PDB scanning finished
Cache FileObject FileName Result
DataSectionObject 0x960f52c60250 chapter11.docx file.0x960f52c60250.0x960f5244f560.DataSectionObject.chapter11.docx.dat

```

Tiếp theo sử dụng các key tìm được và file đã dump ra được ta thử viết một vài dòng code python để thử decrypt file.

```






Open ~\Ransomware\Win10\findaes-12
1 650a41c25760a52545cb5f6d668f9d828
2 1f2abec349b5f892c7e19cc29a4345db
3 cc210b9172cc0e07ef37c910b5b767a5
4 dd42d09101d32b411a1d1fa91b0a14cc47e84340a470c7e307a352e471
5 250a43ac5f702f9fd384d008a34f8051
6 2528f550849b80d293e3a9266b29d80
7 3c9b09152d4c7cb5829d1ffccbd599d6cb588e756f60872000972f50e
8 2d747ad79595d060cfab8960792bd48b39c665ee7484cfce3e7d5f47f
9 61e790d6e2c1461c800cb2230f82874f3e55bb0c71375b054c2389cae
10 29e1bc3b55314e6d64defbeecaf61c04
11
12 557b910b18d7b08526e284785b36a27b
13 2a291d0d7358ec12090a0d6f30bd3a1
14 650a41c25760a52545cb5f6d668f9d828
15
16 5091d8349dc7153a7d79bd84a32d774742252dad7e557222f4e3c9533
17
~\Ransomware\Win10\decrypt_notpetya.py - Mousepad
File Edit Search View Document Help
10 iv = encrypted_file.read(iv_size)
11 # Đọc phần còn lại của file
12 encrypted_data = encrypted_file.read()
13
14 # Tạo đối tượng giải mã AES
15 cipher = AES.new(key, AES.MODE_CBC, iv)
16
17 # Giải mã dữ liệu
18 decrypted_data = cipher.decrypt(encrypted_data)
19
20 # Xóa padding (nếu có)
21 padding_length = decrypted_data[-1]
22 decrypted_data = decrypted_data[:-padding_length]
23
24 # Tạo tên file mới cho file đã được giải mã
25 decrypted_file_path = file_path + ".decrypted"
26
27 with open(decrypted_file_path, 'wb') as decrypted_file:
28     decrypted_file.write(decrypted_data)
29
30 print(f'File đã được giải mã: {decrypted_file_path}')
31
32 # Thay thế 'key' bằng key AES mà bạn đã biết
33 key = b'650a41c25760a52545cb5f6d668f9d828'
34
35 # Thay thế 'file_path' bằng đường dẫn tới file bị mã hóa bởi NotPetya
36 file_path = '/home/kali/Ransomware/Win10/volatility3/file/file.0x960f52c60250.0x960f5244f560.DataSectionObject.chapter11.docx.dat'
37
38 decrypt_file(file_path, key)

```

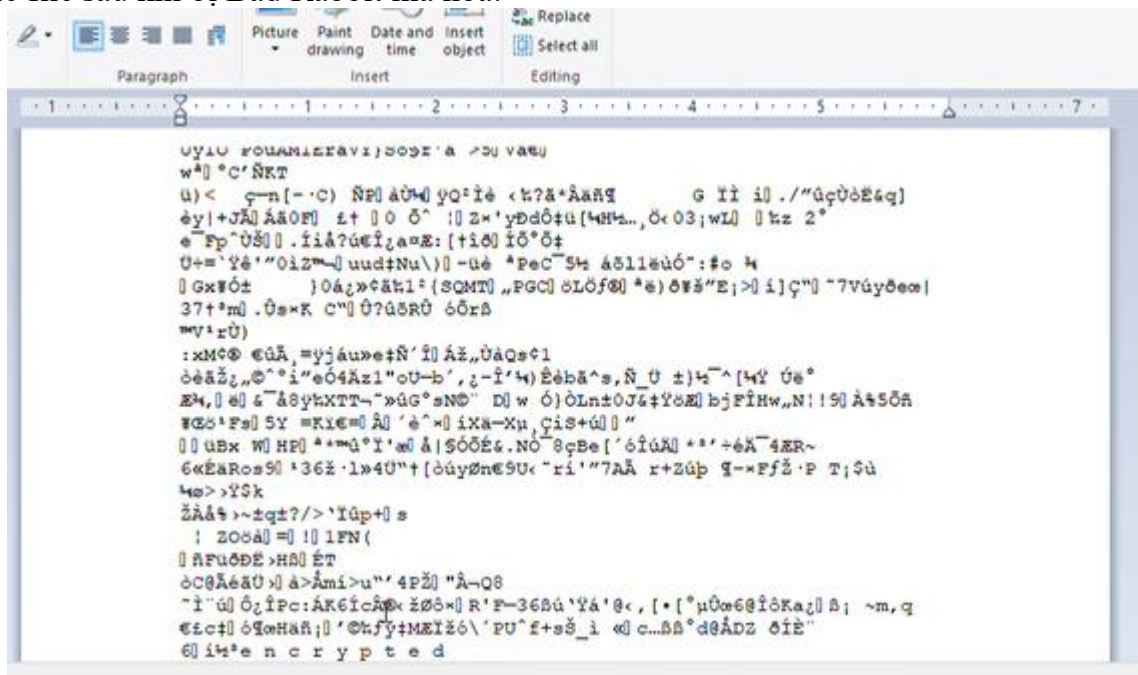
Kết quả decrypt:





	BadRB1	6/3/2024 10:30 AM	RAW File	1,048,576 KB
	BadRB2	6/3/2024 10:32 AM	RAW File	1,048,576 KB
	BadRB3	6/3/2024 10:35 AM	RAW File	1,048,576 KB
	BadRB4	6/3/2024 10:38 AM	RAW File	1,048,576 KB
	BadRB5	6/3/2024 10:39 AM	RAW File	1,048,576 KB

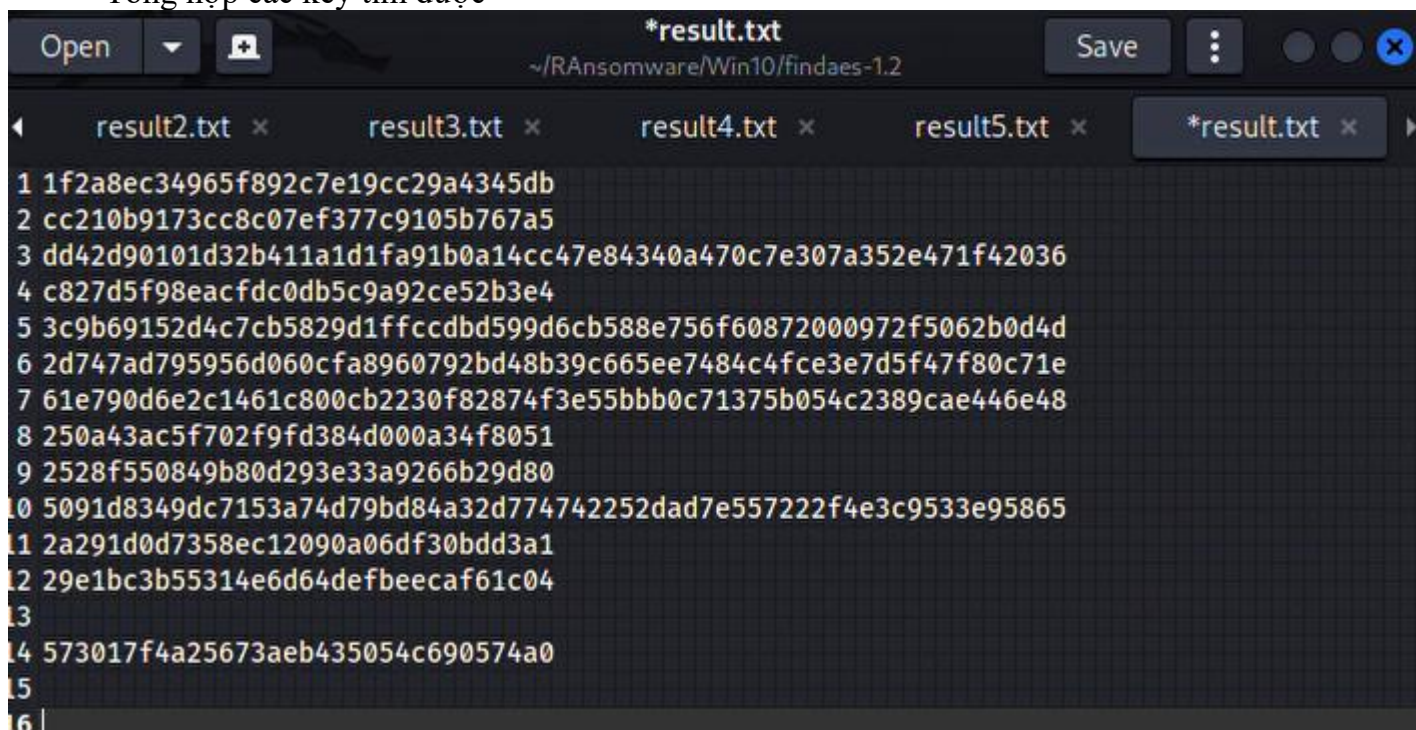
Các file sau khi bị Bad Rabbit mã hóa:



Sử dụng aeskeyfind để tìm key với câu lệnh sau:

*aeskeyfind <tên file>*

Tổng hợp các key tìm được





Tương tự với các bước thực hiện với virus Notpetya, ta tiến hành tìm kiếm file và dump file để phục hồi lại file với các khóa tìm được.

Kết quả scan file:

```
3361 0x960f52f8cd50 \Windows\System32\drivers\DumpIt.sys 216
3362 0x960f52f8f1e0 \Program Files\WindowsApps\Microsoft.Windows.Photos_2019.19071.12548
3363 0x960f52f8f370 \Users\hp\chapter11.docx 216
3364 0x960f52f8f500 \Users\hp\Downloads\Dump\DumpIt.exe 216
3365 0x960f52f8f690 \Windows\System32\Microsoft\Protect\S-1-5-18\User\Preferred 216
3366 0x960f52f8f820 \Program Files\WindowsApps\Microsoft.Windows.Photos_2019.19071.12548
3367 0x960f52f8f9b0 \Program Files\Windows Defender\EppManifest.dll 216
3368 0x960f52f8fb40 \Users\hp\Doc1.docx 216
3369 0x960f52f8fcd0 \Users\hp\AppData\Roaming\Microsoft\Windows\SendTo\Bluetooth File Tr
3370 0x960f52f8fe60 \Users\hp\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinat
3371 0x960f52f90180 \Users\hp\mau-bao-cau-do-an-uit-mau-bao-caotieu-luan-cua-truong-dh-c
3372 0x960f52f90310 \Program Files\WindowsApps\Microsoft.Windows.Photos_2019.19071.12548
3373 0x960f52f904a0 \Windows\Prefetch\TASKHOSTW.EXE-3E0B74C8.pf 216
3374 0x960f52f90630 \CurrentIn 216
3375 0x960f52f907c0 \ProgramData\Microsoft\Windows\AppRepository\Packages\Microsoft.Wind
3376 0x960f52f90950 \Users\hp\Lựa chọn mẫu ransomware.docx 216
3377 0x960f52f91120 \Windows\System32\Windows.Energy.dll 216
3378 0x960f52f912b0 \Windows\Prefetch\SEARCHFILTERHOST.EXE-77482212.pf 216
3379 0x960f52f91440 \Windows\System32\config\systemprofile\AppData\Local\Microsoft\XboxL
```

Tiến hành dump file với câu lệnh:

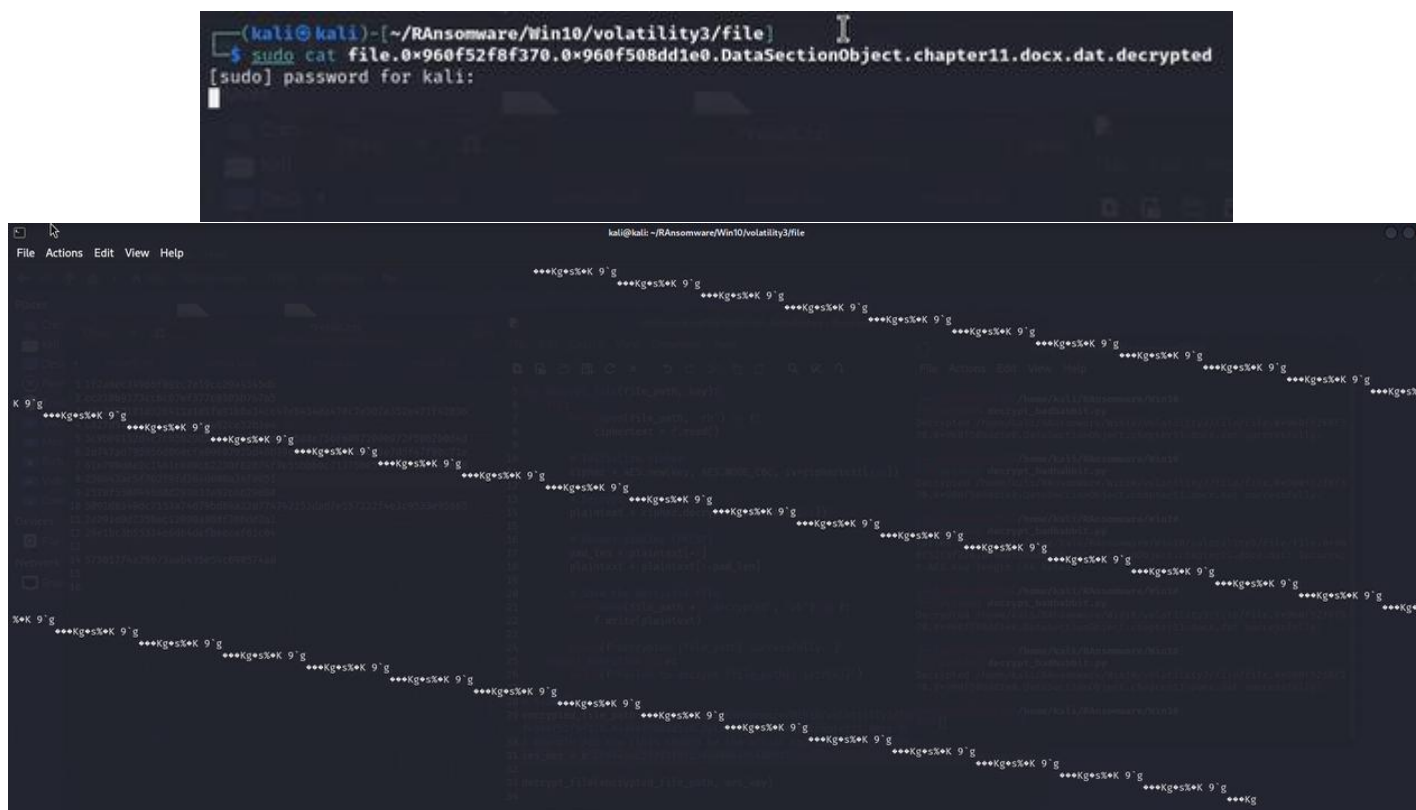
`python3 vol.py -f <tên file> -o <tên thư mục lưu trữ file dump> windows.dumpfiles.DumpFiles -virtaddr <offset>`

```
(kali@kali)~[~/Ransomware/Win10/volatility3]
$ sudo python3 vol.py -f BadRB5.raw -o file windows.dumpfiles.DumpFiles --virtaddr 0x960f52f8f370
Volatility 3 Framework 2.7.1
Progress: 100.00 PDB scanning finished
Cache FileObject FileName Result
DataSectionObject 0x960f52f8f370 chapter11.docx file.0x960f52f8f370.0x960f508dd1e0.DataSectionObject.chapter11.docx.dat
(kali@kali)~[~/Ransomware/Win10/volatility3]
```

Sau đó sẽ thử khôi phục file với tất cả các key tìm được.

```
Open result2.txt result3.txt result4.txt result5.txt *result.txt
1 1f2a8ec34965f892c7e19cc29a4345db
2 cc210b9173cc807ef377c9105b767a5
3 dd42d901d32b411a1d1fa91b0a14cc47e84340a470c307a352e471f42036
4 c827d5f98eacfdcd0b5c9a92ce52b3e4
5 3c9b09152d4c7cb5829d1ffccdbd599dc588e756f60872000972f5062b0d4d
6 2d747ad795956d06cfa896b792b48b39c665ee7484c4fce3e7d5f47f80c71e
7 61e790d6c21461c800cb2238f82b7f3e55bb0c71375b054c2389cae440e48
8 250a43ac5f702f9f4384d000a34f8051
9 2528f508a9b80d29e33a9266b29e80
0 5091d8349dc7153a74d79bd84a32d774742252dad7e557222f4e3c9533e95865
1 2a291d0d7358ec12090a06df30bdd3a1
2 29e1bc3b55314e6d64defbeecaf61c04
3
4 573017f4a25673aeb435054c690574a0
5
6
~/Ransomware/Win10/decrypt_badhabbit.py - Mousepad
View Document Help
le(file_path, key):
    pen(file_path, 'rb') as f:
        phertext = f.read()
    lalize cipher
        = AES.new(key, AES.MODE_CBC, iv=ciphertext[:16])
    ypt the ciphertext
        ext = cipher.decrypt(ciphertext[16:])
    ve padding (PKCS7)
        n = plaintext[:n]
        ext = plaintext[:n-pad_len]
    the decrypted file
        pen(file_path + '.decrypted', 'wb') as f:
            write(plaintext)
    f'Decrypted {file_path} successfully.'
    option as e:
        f'Failed to decrypt {file_path}: {str(e)}'
    e
    _path = '/home/kali/Ransomware/win10/volatility3/fi
        .0x960f508dd1e0.DataSectionObject.chapter11.docx.dat
    key (this should be the actual key used for encrypti
        2a8ec34965f892c7e19cc29a4345db'
    nrypted_file_path, aes_key)
```

Kết quả có được như sau:



Kết quả chưa thành công khôi phục file, quá trình này ta sẽ thực hiện lại từ đầu đến khi tìm ra được key chính xác.

## 7. Những điều kiện cần thiết để thí nghiệm thực hiện thành công

Các điều kiện để thực hiện thành công các phương pháp và các thí nghiệm trong bài báo là:

### 1. Thiết lập môi trường thực tế:

- Môi trường thử nghiệm phải mô phỏng một máy thực càn sát càng tốt, bao gồm việc loại bỏ thông tin riêng tư hoặc bảo mật và cách ly khỏi internet.
- Một mạng giả lập thực tế nên được tạo ra bằng cách sử dụng các công cụ như fakedns.py và INetSim để cung cấp các dịch vụ DNS và mạng khác.
- Máy chủ vật lý nên được "air gapped" để đảm bảo hoàn toàn cách ly.

### 2. Cấu hình máy ảo:

- Cần ba máy ảo: hai máy nạn nhân và một máy cho các dịch vụ hỗ trợ mạng.
- Mỗi thí nghiệm nên bắt đầu với một máy ảo mới để đảm bảo kết quả có thể so sánh.
- Các máy ảo nên được kết nối qua kết nối mạng 'host-only' để cách ly chúng khỏi máy chủ và các kết nối mạng của nó.

### 3. Các bước thực hiện thí nghiệm:

- Mẫu ransomware được thực thi trong môi trường ảo và bộ nhớ khả biến của máy được sao chép trong quá trình thực thi này.
- Các tệp bộ nhớ thu được được phân tích bằng các công cụ pháp chứng để tìm khóa mã hóa.
- Một bản sao bộ nhớ mới nên được thu thập trước khi ransomware thực thi để loại trừ bất kỳ khóa



AES nào có sẵn trước đó.

- Các bản sao bộ nhớ được thực hiện theo các khoảng thời gian đều đặn để tạo ra một biểu đồ thời gian (timeline) về sự hiện diện của khóa trong bộ nhớ.
- Nếu phát hiện bất kỳ khóa nào, thử giải mã các tệp đã được ransomware mã hóa bằng các khóa tìm được cùng với khóa IV, thêm vào đó cần điều chỉnh phần headers và phần ends của file để có thể giải mã.

#### ***4. Các công cụ hỗ trợ thu thập bộ nhớ, tìm khóa AES, giải mã:***

- Nhiều công cụ được sử dụng để sao chép và phân tích bộ nhớ, bao gồm VBoxManage.exe để tạo bản sao bộ nhớ và các công cụ như findaes, interrogate, ransomAES để hỗ trợ tìm khóa AES và một công cụ decrypt.py do tác giả viết để thực hiện giải mã file extract ra được.

#### ***5. Thực thi mẫu ransomware:***

- Các lệnh cụ thể được sử dụng để khởi chạy từng mẫu ransomware

For NotPetya

c:\windows\system32\rundll32.exe

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

For BadRabbit

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

For Phobos

c:\ransomware\lsaas.bin.exe

và thời gian thực thi được xác định thông qua thử và sai nhiều lần.

- Việc thu thập bộ nhớ nên được thực hiện lý tưởng trong khi ransomware đang mã hóa các tệp, vì các khóa không xuất hiện ngay lập tức khi chương trình bắt đầu.
- Các mẫu Ransomware nên được mã hóa AES, key được lưu trữ local. Không nên chọn các mẫu ransomware có mã hóa phức tạp (như ECC, Lattice-based...), hoặc sử dụng khóa có độ dài lớn, trên 256 bit (ở trong bài báo nên là 128 hoặc 256 bit).
- Các file do ransomware mã hóa không bị thay đổi tên, do đó dễ nhận định để giải mã