# ÁP DỤNG NGUYÊN LÝ MÁY HỌC ĐỂ PHÂN TÍCH VÀ PHÁT HIỆN MÃ ĐỘC TRONG PHẦN MỀM MÁY TÍNH

Lê Mạnh*


## TÓM TẮT

*An ninh mạng máy tính trong môi trường internet rất cần thiết đối với các nhà quản trị mạng. Việc phát hiện và loại trừ mã độc (malware) trong các phần mềm máy tính trong mạng là công việc hàng ngày của nhà quản trị hệ thống mạng. Bài báo đề nghị phương pháp dùng nguyên lý máy học (trong chuyên ngành trí tuệ nhân tạo) để thực hiện phát hiện mã độc trong phần mềm máy tính.*


*Từ khóa: Phần mềm mã độc, Malware, Hành vi phần mềm, máy ảo, CSDL mã độc, SandBox, An ninh mạng, Máy học.*


## ABSTRACT

**Applying machine learning principles for the analysis and detection of malicious codes in computer software**

*In the internet environment computer network security is essential for network administrators. The detection and elimination of malware in computer software in the network is the daily work of network administrators. This paper proposes methods using machine learning principles (in specialized artificial intelligence) to perform malware detection in computer software.*


*Key words: Malicious software, malware, behavioral software, virtual machine, malware database, SandBox, network security, machine learning.*

An ninh mạng máy tính là một việc vô cùng cần thiết, đối với các nhà quản trị mạng. Mới đây nhóm chuyên gia bảo mật của Kaspersky Lab vừa phát hiện virút máy tính The Mars (hay còn được gọi Careto) đã là mối đe dọa các hệ thống mạng hàng đầu, virút này đã tham gia vào hoạt động gián điệp mạng toàn cầu từ năm 2007. Loại virút này (The Mars) được thiết kế vô cùng tinh vi, trong nó bao gồm một đoạn mã độc, một rootkit (phá hoại thư mục), một bootkit (phá hoại tệp tin khởi động). Hệ virút The Mars có phiên bản Windows, Mac OS, Linux (Android) và IOS (cho các điện thoại Iphone) [22]. Do vậy việc phát hiện mã độc và loại trừ nó là công việc hàng ngày các nhà quản trị hệ thống tin học. Chúng tôi đề xuất một phương pháp dùng tri thức của ngành trí tuệ nhân tạo để phát hiện phần mềm mã độc.

## 1. Phân loại các loại phần mềm mã độc và các phương pháp phát hiện chúng

Những công cụ cần thiết giúp cho các nhà phân tích phần mềm độc hại có được cái nhìn thấu đáo về hành vi của mã độc. Tuy nhiên, số lượng khổng lồ các mẫu malware mới đến tay các nhà cung cấp giải pháp chống virút hàng ngày đòi hỏi các phương pháp phân tích phải được tự động hóa hơn nữa để hạn chế bớt các thao tác bằng tay của người dùng. Chính vì lẽ đó, các hướng tiếp cận khác nhau đã được đề ra nhằm xếp các mẫu thử chưa biết vào các họ mã độc đã biết hoặc đánh dấu hành vi lạ của những mẫu thử đó để yêu cầu các phân tích bằng tay kỹ hơn. Sẽ có một cái nhìn tổng quan về các phương pháp xếp lớp malware [1,2].

* TS, Trường ĐH Văn Hiến

### 1.1. Gộp nhóm mã độc dựa trên Anubis

Như một phản ứng trước hàng chục ngàn mẫu malware mới xuất hiện mỗi ngày, nhà nghiên cứu U. Bayer đã giới thiệu một hệ thống có khả năng gộp nhóm hiệu quả và tự động các tập mã độc lớn dựa trên hành vi của chúng. Bằng cách so sánh các hành vi mã độc, nhà phân tích có thể tập trung vào các mối đe dọa mới và tiết kiệm thời gian khi không phải phân tích những mẫu thử có những hành vi đã biết.

Kỹ thuật được đưa ra dựa trên Anubis để xuất ra dấu vết thực thi của tất cả các mẫu thử. Hơn thế nữa, Anubis còn được mở rộng với khả năng lan truyền dấu vết, để tận dụng những nguồn thông tin bổ sung. Chẳng hạn, nếu một mẫu malware tạo một file với tên file phụ thuộc vào giờ hệ thống hiện hành, sự phụ thuộc này sẽ bị phát hiện bởi hệ thống. Ngoài ra, khả năng lan truyền dấu vết giúp phát hiện những đặc trưng của sâu mạng như thao tác đọc mã thực thi của một tiến trình và truyền chúng đi trên mạng. Cuối cùng, tập quyết định dòng điều khiển phụ thuộc vào dữ liệu bị đánh dấu cũng bị ghi lại.

Khi các dấu vết thực thi cùng với thông tin bị đánh dấu được tạo ra, một profile hành vi được trích xuất ra cho mỗi dấu vết. Profile bao gồm các đối tượng hệ điều hành (như file, tiến trình...) và các thao tác tương ứng (đọc, ghi, tạo...). So với chuỗi lời gọi hệ thống được lưu trong dấu vết thực thi, cách thể hiện này giàu ngữ nghĩa hơn, do đó cho phép hệ thống hợp nhất những hành vi tương đương ngữ nghĩa một cách tốt hơn. Ví dụ, đọc 256 byte cùng lúc từ một file tương đương ngữ nghĩa với đọc 1 byte, 256 lần. Tuy nhiên, cách thể hiện trong dấu vết thực thi có khác biệt đáng kể. Trong quá trình tạo ra profile hành vi, thông tin ghi nhận được thông qua lan truyền dấu vết được sử dụng để mô tả xem liệu các đối tượng có phụ thuộc lẫn nhau như đã mô tả hay không.

Profile hành vi thu được sẽ được sử dụng như là đầu vào cho một thuật toán gộp nhóm, kết hợp nhiều profile cùng mô tả những hành vi giống nhau thành một cluster liền mạch. Để chứng minh cho khả năng mở rộng phương pháp của mình, các tác giả đã đánh giá công cụ của họ trên một tập 75000 mẫu thử được gộp nhóm trong vòng 3 tiếng đồng hồ.

### 1.2. Các hệ để chia mẫu thử mã độc

Các nhà khoa học Lee và Mody cũng đề xuất một hệ thống để chia các mẫu thử malware thành các cluster dựa trên profile hành vi bằng cách áp dụng kỹ thuật máy học. Việc thực thi của những mẫu thử này thực hiện trong một môi trường ảo được giám sát chặt chẽ. Một công cụ theo dõi ở kernel-mode sẽ ghi lại tất cả các lời gọi hệ thống cùng với đối số của chúng. Thông tin thu được về tương tác của mẫu thử với hệ thống sẽ được ghi vào trong profile hành vi. Profile này gồm có thông tin liên quan đến tương tác của mẫu thử với tài nguyên hệ thống như ghi file, thay đổi khóa registry hay các hoạt động mạng. Để đo lường được sự tương đồng giữa hai profile, khoảng cách hiệu chỉnh giữa chúng sẽ được tính toán, chi phí của sự biến đổi được định nghĩa trong một ma trận chi phí thao tác. Các tác giả sau đó sẽ áp dụng phương pháp gộp nhóm k-medoids để chia các mẫu thử mã độc thành các cluster có profile hành vi tương đồng [3,4]. Khi quá trình huấn luyện đã hoàn thành, mẫu thử mới và chưa được biết đến sẽ được gán vào cluster có medoid gần với mẫu thử nhất.

### 1.3. Áp dụng máy học để xếp loại hành vi mã độc

Các chương trình đóng gói giúp tin tặc dễ dàng hơn trong việc tạo ra các thể hiện của mã độc vượt qua được sự kiểm tra của các sản phẩm quét virút truyền thống. Vì vậy, cần thiết phải tìm ra những phương pháp khác để xếp loại các mã độc chưa biết. Để làm được điều này, Konrad Rieck đã giới thiệu một hệ thống sử dụng thông tin hành vi chứa trong các báo cáo từ công cụ CWSandbox. Đầu tiên, profile hành vi của mỗi họ mã độc đã được biết đến sẽ được trích xuất. Tiếp theo, kỹ thuật máy học được áp dụng để tạo ra bộ xếp loại từ những profile đó, cho phép nhóm những thể hiện mã độc có hành vi tương tự nhau [6,7,8,21].

Để củng cố cho phương pháp này, các tác giả sử dụng một phần mềm chống virút thương mại quét một số lượng lớn các mẫu thử để gán nhãn cho từng mẫu thử. Với những mẫu có thể nhận diện được, những nhãn này tương ứng với những họ mã độc có chứa mẫu thử này. Dựa vào những nhãn này và profile hành vi trích xuất từ các bản báo cáo của CWSandbox, các tác giả huấn luyện các bộ máy vector hỗ trợ (SVM) để xây dựng nên những bộ xếp loại cho từng họ mã độc. Mỗi SVM chỉ đưa ra được khả năng mà mẫu thử có thể thuộc về một họ nào đó. Hệ

thống có thể gặp những mẫu thử không thuộc về một họ đã biết nào hoặc thể hiện các hành vi đặc trưng của nhiều họ khác nhau. Quá trình quyết định được cấu trúc sao cho nó đưa ra được duy nhất một họ mà mẫu thử thuộc về. Nếu mẫu thử thể hiện các hành vi thuộc về nhiều họ hoặc không thuộc bất cứ họ nào, mẫu thử sẽ được xếp loại là chưa biết, cần phải phân tích kỹ hơn.

Dựa vào việc nhúng các báo cáo vào một không gian vector, các tác giả đã áp dụng kỹ thuật máy học để phân tích hành vi. Đặc biệt, các tác giả nghiên cứu hai khái niệm máy học để phân tích: Gộp nhóm hành vi, cho phép xác định các lớp mới của malware có hành vi tương tự và phân loại các hành vi, cho phép gán phần mềm mã độc vào các lớp hành vi được biết đến. Để bắt kịp tốc độ ngày càng tăng của số lượng phần mềm độc hại trên mạng, các phương pháp phân nhóm và phân loại đòi hỏi phải xử lý được hàng ngàn báo cáo hàng ngày. Thật không may, hầu hết các phương pháp máy học lại có chi phí tính toán tỷ lệ siêu tuyến tính với số lượng dữ liệu đầu vào do đó không thể áp dụng trực tiếp cho phân tích phần mềm mã độc.

Để giải quyết vấn đề này, các tác giả đề xuất một kỹ thuật phân nhóm xếp lớp lấy cảm hứng từ thành quả của U. Bayer và các đồng sự. Một tập hợp các mã nhị phân độc hại thường bao gồm các biến thể tương tự của cùng một họ, thể hiện các hành vi gần như giống hệt nhau. Kết quả là, các báo cáo nhúng tạo thành những đám mây dày đặc trong không gian vector. Các tác giả khai thác sự thể hiện dày đặc này bằng cách gom các hành vi tương tự thành từng nhóm sử dụng khái niệm nguyên mẫu - prototype. Bằng cách hạn chế các tính toán của phương pháp học tập vào nguyên mẫu và sau đó lan truyền kết quả cho tất cả các dữ liệu nhúng, các tác giả có thể đẩy nhanh quá trình gộp nhóm cũng như phân loại. Các nguyên mẫu chiết xuất tương ứng với với các báo cáo thông thường và do đó có thể dễ dàng kiểm tra bởi sự phân tích của con người, trong khi phương pháp của Bayer gần như không cung cấp được những nhận thức về nhóm các hành vi.

**Gộp nhóm** đề cập đến một kỹ thuật cơ bản của máy học nhằm mục đích phân vùng một tập dữ liệu thành các nhóm có ý nghĩa, được gọi là những cluster. Phân vùng được xác định sao cho các đối tượng trong cùng một cluster là tương tự như nhau, trong khi các đối tượng trong các cluster khác nhau không giống nhau. Gộp nhóm cho phép khám phá ra cấu trúc trong dữ liệu chưa biết và do đó đã được sử dụng trong một lượng lớn các ứng dụng khác nhau.

Gộp nhóm để phân tích hành vi của phần mềm độc hại đã được đề xuất bởi Bailey và các đồng sự và sau đó được hoàn thiện bởi nhóm của U. Bayer. Các tác giả đã theo đuổi hướng nghiên cứu này và khảo sát kỹ thuật gom nhóm phân cấp tiêu chuẩn để xác định nhóm các phần mềm mã độc có hành vi tương tự. Ngược lại với công trình trước đó, các tác giả đã triển khai phân tích dựa trên khái niệm nguyên mẫu - prototype. Đó là, các cluster đầu tiên của nguyên mẫu được xác định và sau đó được phát tán tới dữ liệu ban đầu.

Tiếp theo các tác giả tiến hành phân loại, cho phép tìm hiểu sự phân biệt các lớp đối tượng khác nhau. Phương pháp phân loại đòi hỏi một giai đoạn học tập trước khi áp dụng, một mô hình cho việc phân biệt được suy ra từ tập dữ liệu của các đối tượng dán nhãn. Mô hình này sau đó có thể được áp dụng để dự đoán nhãn lớp dữ liệu chưa từng được biết. Cũng như nhiều ứng dụng thực tế dựa vào khái niệm về học tập, phần lớn nghiên cứu này tồn tại dựa trên việc thiết kế và áp dụng các phương pháp phân loại.

Việc áp dụng phân loại để phân tích hành vi phần mềm độc hại đã được nghiên cứu bởi Lee và Mody (2006) cùng với Konrad Rieck và các cộng sự (2008). Trong cả hai phương pháp tiếp cận, hành vi của các phần mềm mã độc chưa được biết đến được phân loại thành các lớp hành vi đã được biết đến, nơi các dữ liệu đào tạo ban đầu được dán nhãn bằng cách sử dụng các chương trình quét virút. Thật không may, hầu hết các sản phẩm chống virút đều chịu tác động từ các nhãn không phù hợp và không đầy đủ do đó đã không cung cấp nhãn đầy đủ chính xác cho quá trình huấn luyện. Như một biện pháp khắc phục, các tác giả sử dụng các lớp phần mềm mã độc được phát hiện bởi quá trình gộp nhóm làm nhãn cho việc huấn luyện từ đó hiểu được sự phân biệt giữa các cluster đã biết của hành vi mã độc [7,9.10].

**Phân tích bổ sung:** Dựa trên một công thức chung của nhóm và phân loại, các tác giả nghĩ ra một phương pháp phân tích bổ sung để phân tích hành vi của phần mềm độc hại. Trong khi

công trình trước đó đã được thiết kế để phân tích hàng loạt, các tác giả đề xuất xử lý các báo cáo hành vi của phần mềm mã độc trong từng khối nhỏ, chẳng hạn trên cơ sở hàng ngày. Để thực hiện phân tích gia tăng, chúng ta cần phải theo dõi các kết quả trung gian, chẳng hạn như cluster được xác định trong quá trình chạy các thuật toán. May mắn thay, các khái niệm về nguyên mẫu cho phép các tác giả lưu trữ các cluster phát hiện trong một cách thể hiện súc tích, và hơn nữa, tăng tốc đáng kể nếu sử dụng cho việc phân loại.

Các báo cáo được phân tích nhận được từ một nguồn dữ liệu như một tập hợp các honeypots hoặc cơ sở dữ liệu của phần mềm độc hại được thu thập trong khoảng thời gian thường xuyên. Trong giai đoạn xử lý đầu tiên, các báo cáo được phân loại bằng cách sử dụng nguyên mẫu của cluster được biết đến. Qua đó, các biến thể của phần mềm độc hại đã biết được nhận diện hiệu quả và được lọc từ các phân tích sâu hơn. Trong giai đoạn tiếp theo, nguyên mẫu được chiết xuất từ các báo cáo còn lại và sau đó được sử dụng cho việc gom nhóm hành vi. Các nguyên mẫu của cụm mới được lưu giữ cùng với các thiết lập ban đầu của nguyên mẫu, như vậy chúng có thể được áp dụng trong lượt phân loại tiếp theo. Quá trình này bao gồm hai công đoạn phân loại và gộp nhóm nhóm luân phiên xen kẽ được lặp đi lặp lại từng bước, khiến cho số lượng malware chưa biết liên tục giảm và các lớp phổ biến của phần mềm mã độc sẽ tự động phát hiện ra.

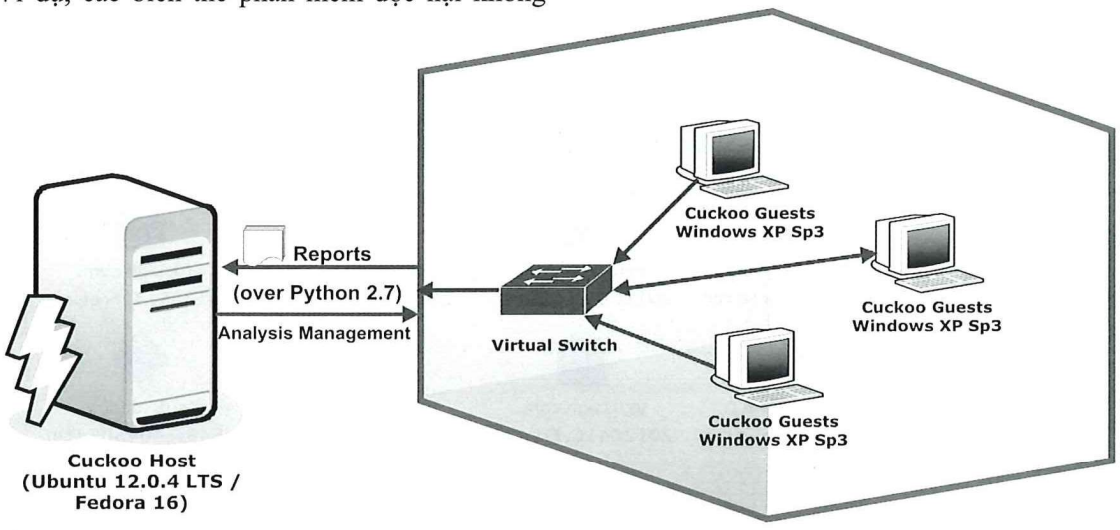
Số lượng các báo cáo có sẵn trong một lượt phân tích bổ sung có thể là không đủ để xác định tất cả các cluster hành vi của phần mềm độc hại. Ví dụ, các biến thể phần mềm độc hại không xuất hiện thường xuyên có thể chỉ được đại diện bởi vài mẫu trong không gian vector nhúng. Để bù đắp cho sự thiếu thông tin, các tác giả bác bỏ cluster có ít hơn m thành viên và trả các báo cáo tương ứng về lại nguồn dữ liệu. Do đó, các malware không xuất hiện thường xuyên được tích lũy dần cho đến khi đủ dữ liệu cho việc gộp nhóm. Thủ tục này đảm bảo phát hiện chính xác các lớp phần mềm mã độc, thậm chí khi các thông tin có liên quan không có sẵn trong quá trình chạy gia tăng đầu tiên.

Thời gian chạy của thuật toán gia tăng là O (nm + k2logk) cho một đoạn n báo cáo trong đó m là số nguyên mẫu được lưu trữ từ lượt chạy trước và k là số nguyên mẫu được chiết xuất trong lượt chạy hiện tại. Mặc dù sự phức tạp thời gian chạy là bậc hai với k, số lượng mẫu thử nghiệm chiết xuất trong mỗi quá trình chạy vẫn không đổi cho từng khối có kích thước bằng nhau. Vì vậy, sự phức tạp của phân tích bổ sung được xác định bởi m, số lượng prototype cho các lớp phần mềm mã độc đã biết, tương tự như sự phức tạp tuyến tính của việc khớp dấu hiệu trong các sản phẩm anti-virút [6,7].

**2. Thử nghiệm chương trình cài đặt trong mạng ảo các lý để phát hiện mã độc**

**2.1. Cấu trúc mô hình**

Mô hình được xây dựng với cấu trúc gồm có một máy Host chạy hệ điều hành Linux Ubuntu 12.04 LTS hoặc Fedora 16 và nhiều máy Guest cài hệ điều hành Windows XP Service Pack 3. Gói điều khiển Cuckoo Sandbox 0.3.2 được triển khai trên máy Host. Phần mềm máy ảo Virtualbox 4.1.12 của hãng Oracle được sử dụng để cài các máy Guest.




Cấu trúc Cuckoo Sandbox. Fedora 16

**Các thư viện Python**

Các thành phần máy chủ của Cuckoo được viết hoàn toàn bằng ngôn ngữ Python, do đó, ngôn ngữ Python phiên bản mới nhất 2.7 là một phần không thể thiếu trên máy Host. Ngoài ra, Cuckoo còn cần đến một số thư viện khác trên Python như: Magic (để phát hiện các loại tập tin), Dpkt (chiết xuất thông tin liên quan từ các tập tin pcap), Mako (để kết xuất các báo cáo HTML và giao diện web).

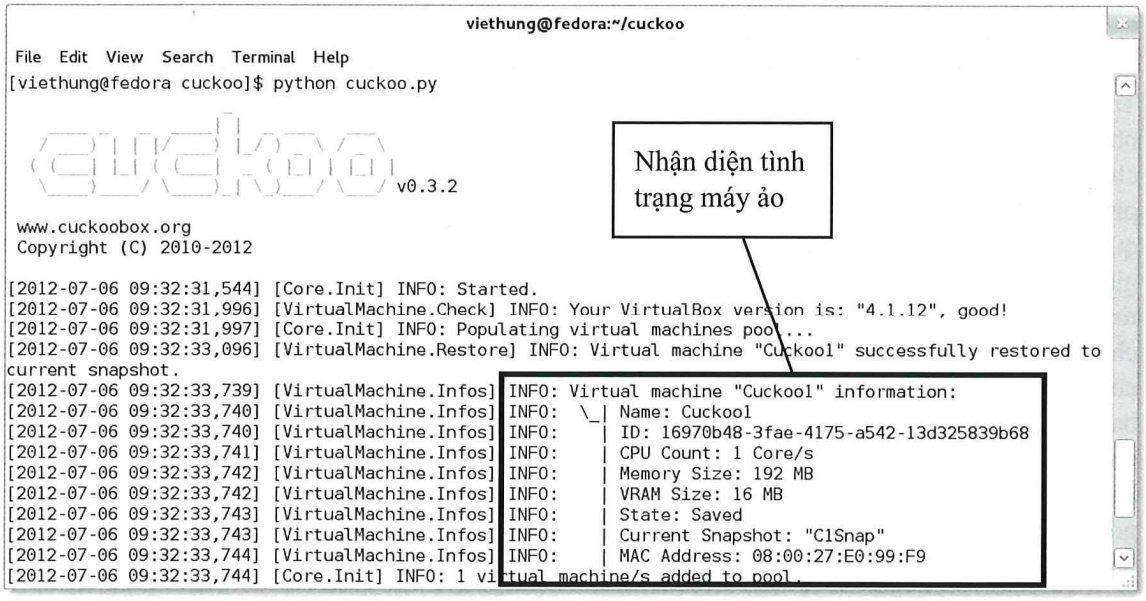
**Virtualbox**

Mặc dù Virtualbox thường được đóng gói sẵn trong các bản phân phối GNU/Linux nhưng tốt nhất chúng ta nên sử dụng bản mới nhất tải về từ trang chủ. Lý do của sự lựa chọn này là các bản đóng gói của Virtualbox (được gọi là OSE) thường có một số hạn chế và đã được điều chỉnh để đáp ứng yêu cầu của giấy phép GNU GPL.

Mặc định, Cuckoo sử dụng các chức năng theo dõi hành vi mạng được nhúng sẵn trong Virtualbox, tuy nhiên để tăng cường khả năng này, chúng ta có thể sử dụng gói cài đặt ngoài Tcpdump hoặc iNetsim.

**Tạo user**

Mặc dù hoàn toàn có thể chạy Cuckoo với user hiện tại, trong thực tế vẫn nên tạo một user mới.

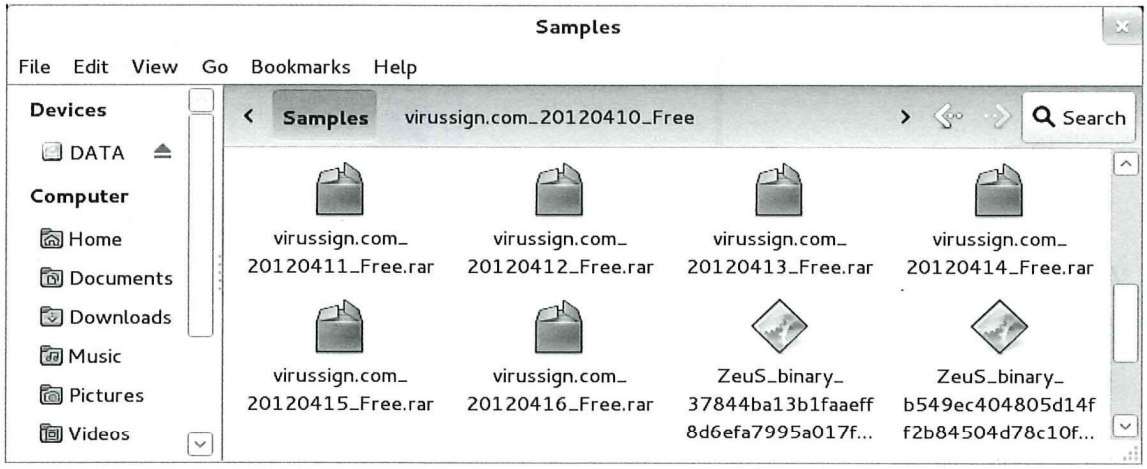



*Khởi động Cuckoo Sandbox*

Lúc này, Cuckoo đã sẵn sàng hoạt động, chờ mẫu thử được nạp vào để phân tích.

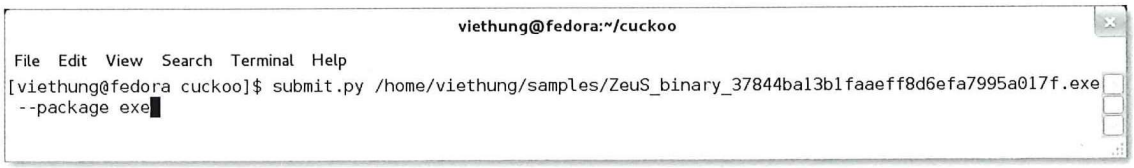
**Nạp mẫu thử**

Các mẫu thử malware được thu thập từ các trang web virussign.com, virustotal.com và các trang web cung cấp dịch vụ honeypot khác.




*Mẫu thử malware*

Để nạp mẫu thử, chúng ta có thể sử dụng chức năng submit.py :




*Sử dụng lệnh submit.py để nạp mẫu thử*

Cuckoo còn được thiết kế để dễ dàng tích hợp trong hệ thống giải pháp lớn hơn với cách hoạt động hoàn toàn tự động. Để làm được điều này, chúng ta có thể thao tác trực tiếp trên cơ sở dữ liệu SQLite tại db/cuckoo.db.

**Các gói phân tích**

Các gói phân tích là thành phần chính trong Cuckoo Sandbox. Chúng bao gồm các script cấu trúc Python được thực thi bên trong máy ảo, xác định cách mà Cuckoo thực hiện việc phân tích. Nếu không có gói phân tích nào được gọi ra trong câu lệnh, Cuckoo sẽ cố gắng phát hiện mẫu thử thuộc loại file nào để chọn được gói phân tích phù hợp. Nếu định dạng file không được hỗ trợ và không có gói phân tích nào được gọi ra, việc phân tích sẽ bị hủy bỏ và được đánh dấu thất bại trong cơ sở dữ liệu.

Cuckoo cung cấp một số gói phân tích mặc định sau:

• exe: gói phân tích mặc định được sử dụng để phân tích chung các file thực thi của Windows.

• Dll: được sử dụng để phân tích thư viện liên kết động.

• pdf: được sử dụng để phân tích tập tin PDF Adobe Reader.

• doc: được sử dụng để phân tích các tài liệu Microsoft Office.

• php: được sử dụng để phân tích các script PHP.

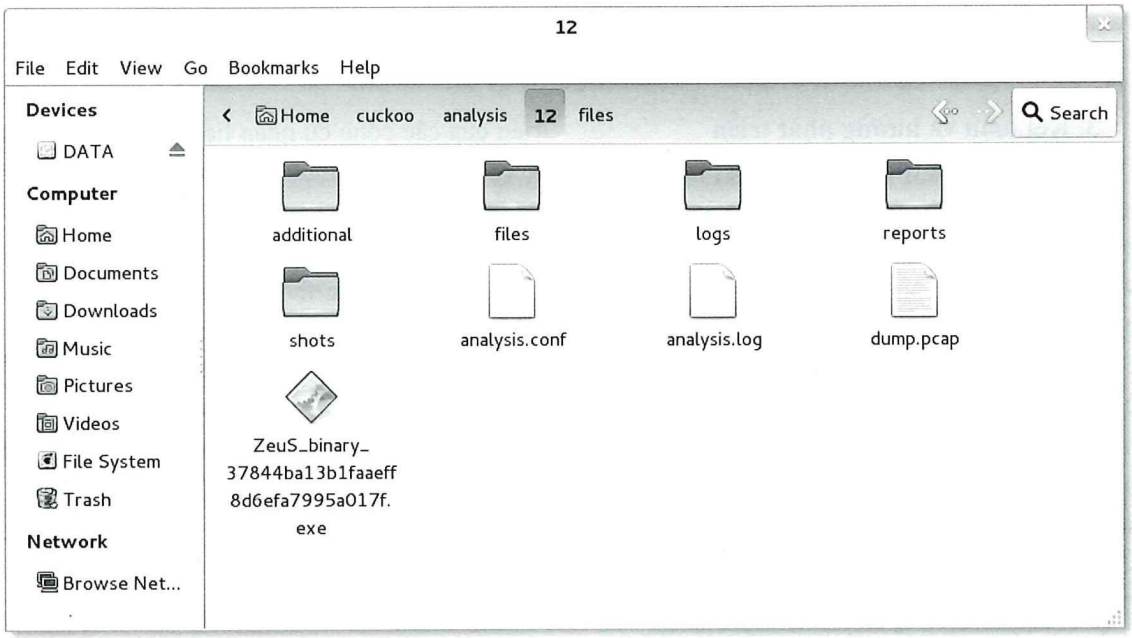
• ie: sử dụng Internet Explorer để phân tích các URL được cung cấp.

• firefox: sử dụng Firefox để phân tích các URL được cung cấp.

Kết quả phân tích

Sau khi việc phân tích một mẫu được hoàn thành, các tập tin kết quả sẽ được lưu trong một thư mục chuyên dụng. Trừ khi chúng ta cấu hình lại, tất cả các kết quả phân tích được lưu vào analysis với một thư mục con được đặt tên theo số ID được chỉ định để phân tích trong cơ sở dữ liệu.

Sau đây là một ví dụ về kết quả phân tích:



Behavior Analysis | Network Analysis | Static Analysis | Dropped Files

## Network Summary

No signatures matched.

## Network Stats

| | |
|---|---|
| **Nr. of TCP packets:** | 0 |
| **Nr. of UDP packets:** | 46 |
| **Nr. of DNS requests:** | 0 |
| **Nr. of HTTP requests:** | 0 |

## Involved Hosts

| **IP Address** |
|---|
| 0.0.0.0 |
| 255.255.255.255 |
| 10.0.2.2 |
| 10.0.2.15 |
| 239.255.255.250 |
| 224.0.0.22 |
| 10.0.2.255 |

## Network Analysis

Nothing to display.

*Kết quả phân tích hoạt động mạng*

**3. Kết luận và hướng phát triển**

Trước khi phát triển các phương pháp chống lại các phần mềm mã độc, điều quan trọng là phải hiểu được cơ chế hành động của malware và các kỹ thuật ngụy trang của chúng. Các phương pháp trên đưa một cái nhìn tổng quan về các kỹ thuật phân tích động tại thời điểm hiện tại cũng như các công cụ hỗ trợ các nhà phân tích trong việc thu thập thông tin về hành vi mã độc một cách nhanh chóng và chi tiết. Hệ thống dã sử dụng các công cụ ngụy trang (như đóng gói, làm nhiễu, tự động thay đổi code...), malware đã có thể vượt qua được các phương pháp phân tích tĩnh, điều này dẫn đến sự xuất hiện và phát triển của các công cụ phân tích động. Phân tích động là quá trình thực thi một mẫu phần mềm mã độc và theo dõi hành vi của chúng.

Hầu hết các công cụ phân tích động đều được trang bị chức năng theo dõi API và lời gọi hệ thống được sử dụng bởi mẫu thử. Việc sử dụng các API và các lời gọi hệ thống là cần thiết để malware có thể tương tác với môi trường của nó. Phân tích các tham số được truyền tới các API và hàm hệ thống này giúp chúng ta có thể gom các lời gọi hàm khác nhau thành các nhóm theo logic ngữ nghĩa. Những việc này được thực hiện bằng các kỹ năng chuyên dụng như cấy hàm ký sinh, biên tập lại mã nhị phân, nhúng thư viện

động... Ngoài ra, nhiều công cụ phân tích còn cung cấp chức năng giám sát cơ chế xử lý và tự lây nhiễm của các loại dữ liệu nhạy cảm trong hệ thống. Những thông tin này đóng vai trò là một đầu mối giúp các nhà phân tích có thể hiểu được malware đang xử lý loại dữ liệu nào. Từ đó các nhà phân tích có thể xác định được các hoạt động đã được thực thi để hoàn thành nhiệm vụ bất hợp pháp của phần mềm mã độc.

Thông tin mà các nhà phân tích thu được từ những công cụ này giúp họ có được nhận thức về hành vi của các phần mềm mã độc, từ đó giúp cơ quan tổ chức của mình triển khai các phương pháp đối phó với malware một cách kịp thời và phù hợp. Những thông tin hữu ích này không chỉ giúp họ loại bỏ được phần mềm mã độc đó mà còn cho những nhà quản trị an ninh một cái nhìn thấu đáo về những tác động của malware đó đến hệ thống của mình. Họ có thể định lượng một cách chính xác thiệt hại để có thể khắc phục hậu quả một cách triệt để, kịp thời cũng như lên được phương án vá lỗ hổng và xây dựng kịch bản phòng vệ nhằm đối phó hiệu quả các đợt tấn công mới. Bài nghiên cứu của chúng tôi, hy vọng sẽ thay đổi được quan niệm của các nhà quản trị hệ thống và quản trị mạng, từ đó hình thành nên thói quen làm việc mới, chủ động hơn, hiệu quả hơn trong lĩnh vực an ninh mạng.

## TÀI LIỆU THAM KHẢO


1. Daisuke Inoue, Katsunari Yoshioka, Masashi Eto, Yuji Hoshizawa, Koji Nakao (2010), *"Malware Behavior Analysis in Isolated Miniature Network for Revealing Malware's Network Activity"*, IEEE Xplore.

2. Dennis Distler (2011), *Perfoming Behavioral Analysis of Malware*, SANS Training.

3. Egele, M., Theodoor, S., Engin, K., & Christopher, K. (2010), "A Survey on Automated Dynamic Malware Analysis Techniques and Tools", *ACM Computing Surveys*, 1-49.

4. Jim Clausing (2009), "Building an Automated Behavioral Malware Analysis Environment using Open Source Software", *SANS Institute InfoSec Reading Room*, 18/6/2009.

5. Katsunari Yoshioka, Takahiro Kasama, and Tsutomu Matsumoto (2010), *Sandbox Analysis with Controlled Internet Connection for Observing Temporal Changes of Malware Behavior*, Yokohama National University, Yokohama, Japan.

6. Michael Sikorski, Andrew Honig (2011), *Practical Malware Analysis*, No Starch Press.

7. Roberto Sponchioni (2011), "Run-time Malware Analysis System", *IT Security for the Next Generation International Cup 2011*, Munich, Germany.

8. Konrad Rieck, Philipp Trinius, Carsten Willems and Thorsten Holz, 2011, "Automatic Analysis of Malware Behavior using Machine Learning", *Journal of Computer Security*, Volume 13, Number 4/2011, 639-668.

9. Philipp Trinius, Thorsten Holz, Konrad Rieck and Carsten Willems, 2010, "A malware Instruction Set for Behavior-Based Analysis", Sicherheit 2010 (Sicherheit, Schutz und Verlässlichkeit), 205–216, Berlin, Germany.

10. Konrad Rieck, Thorsten Holz, Carsten Willems, Patrick Dussel, and Pavel Laskov, 2008, "Learning and Classification of Malware Behavior", *Detection of Intrusions and Malware, and Vulnerability*

*Assessment (DIMVA)*, 108–125, Berlin, Germany.

11. Nguyễn Việt Hùng (2012), *Phương pháp phân tích và theo dõi hành vi mã độc trong môi trường mạng ảo cách ly*, Luận án cao học tại Học viên Bưu chính – Viễn thông Khu vực phía Nam.

12. Lê Mạnh, Nguyễn Việt Hùng (2012), "Theo dõi hành vi mã độc trong môi trường mạng ảo cách ly", *Kỷ yếu hội thảo quốc gia lần thứ XV về CNTT: Một số vấn đề chọn lọc của CNTT và truyền thông*, trang 443-446.

13. Báo SGGP ngày 19/02/2014, *Các phát hiện virut máy tính The Mars của Kaspersky Lab*.

14. http://malwr.com/

15. http://virustotal.com/

16. http://www.joesecurity.org/

17. http://www.threattrack.com

18. http://www.norman.com/security_center/security_tools/

19. http://www.bkav.com.vn/tin_tuc_noi_bat/-/view_content/content/tong-ket-tinh-hinh-virus-va-an-ninh-mang-nam-2011, Truy cập: 13/01/2012.

20. http://hanoimoi.com.vn/newsdetail/Cong-nghe/534610/an-ninh-mang-nam-2011-nhieu-dien-bien-bat-thuong.htm, Truy cập: 30/12/2011.

21. http://anubis.iseclab.org/

22. http://cuckoobox.org/