HỌC MÁY ĐỐI VỚI MÃ ĐỘC PE CÓ KHẢ NĂNG TRỐN TRÁNH

CK06 - Bài báo: Optimization of code caves in malware binaries to evade machine learning detectors

Nhóm G02
NT522.N21.ATCL
Phương pháp học máy trong An Toàn Thông Tin

Nguyễn Mạnh Cường
20520421
ATCL2020

Hoàng Văn Anh Đức
20520890
ATCL2020

Lê Quang Minh*
20520245
ATCL2020

ABSTRACT

Đề tài về các mẫu mã độc tệp PE trên Windows có khả năng trốn tránh (Adversarial Examples – Mẫu đối kháng). Bài báo [6] đề xuất phương pháp tạo mẫu đối kháng trong điều kiện Black-box, tận dụng những khoảng trống giữa các sections trong tệp PE và ứng dụng thuật toán tiến hóa di truyền (Genetic Algorithm). Thực nghiệm phương pháp tạo mẫu với tập dữ liệu DikeDataset [4] và so sánh với phương pháp GAMMA [5]. Kết quả đưa ra ưu và nhược điểm của cả hai phương pháp, đề xuất cải thiện và hướng phát triển.

KEYWORDS

Học máy, Mẫu đối kháng, Black-box, Tệp PE, Thuật toán tiến hóa di truyền

1 GIỚI THIỆU

Hiện nay, sự gia tăng đáng kể của các mã độc làm cho các trình phát hiện mã độc truyền thống (chủ yếu dựa trên Signature và Behavious) trở nên không còn hiệu quả. Bên cạnh đó, những giải pháp ứng dụng trí tuệ nhân tạo (AI) và học máy (ML), đặc biệt là sự phát triển của các mô hình phân loại dựa trên học máy phân tích tĩnh các tệp thực thi, đã chứng minh được tiềm năng và kết quả khả quan của nó. Tiêu biểu kể đến, trong lĩnh vực học sâu (DL) với các mạng nơ-rôn nhân tạo (ANN) có khả năng tự trích xuất được các thuộc tính giá trị cho việc phân loại, từ đầu vào là tệp thực thi hoàn chỉnh.

Tuy nhiên, những mô hình học máy phân loại tĩnh này cũng có những hạn chế của nó. Được biết đến nhiều nhất là khó khăn khi đối mặt với các mẫu mã độc có khả năng trốn tránh, hay còn được gọi là mẫu đối kháng (Adversarial Examples). Mẫu đối kháng là những mẫu mã độc (trong bài báo tập trung đến mã độc dạng tệp thực thi PE trên hệ điều hành Windows) đã được kẻ tấn công chế tạo bằng cách chèn vào đó những nội dung nhiễu. Mục đích là làm những mô hình học máy phân loại, đặc biệt là phân tích tĩnh, phân loại sai chúng là lành tính, nhưng vẫn có thể giữ được chức năng độc hại khi thực thi.

Từ những nghiên cứu liên quan, bài báo đưa ra 3 yếu tố quan trọng của những nội dung nhiễu để chèn vào mẫu đối kháng. Đó là vị trí chèn trong tệp PE, kích thước nhỏ nhất có thể để đạt hiệu quả và nội dung phù hợp của nó. Với mục tiêu là tối ưu được 3 yếu tố trên, đồng thời tạo cơ sở cho việc đánh giá và cải thiện các mô hình học máy phân loại mã độc dựa trên phân tích tĩnh tệp thực thi trên Windows, bài báo đề xuất phương pháp của mình.

*Thuộc nhóm, nhưng không đóng góp cho đồ án.

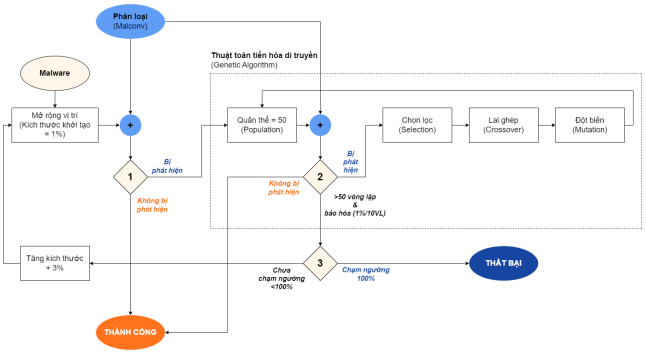

Figure 1: Mô hình phương pháp.

Phương pháp tạo ra các mẫu đối kháng từ những mẫu mã độc có sẵn, phương pháp triển khai trong điều kiện Black-box (tức là không biết gì về thông tin, cấu trúc của mô hình phân loại). Bên cạnh đó, phương pháp tận dụng cơ thế phân trang của hệ điều hành và cơ chế tải tệp thực thi vào bộ nhớ (RAM) để có thể chèn nội dung nhiễu vào vị trí trống giữa các sections trong tệp PE, mà không làm ảnh hưởng đến chức năng độc hại của tệp mã độc gốc. Phương pháp cũng xác định được kích thước phần nội dung nhiễu nhỏ nhất có thể để đạt được kết quả trốn tránh. Ngoài ra, ứng dụng thuật toán tiến hóa di truyền (Genetic Algorithm) để chọn ra nội dung của phần nhiễu phù hợp và đạt kết quả tốt nhất.

2 PHƯƠNG PHÁP

Hình 1 là chi tiết các bước của phương pháp được đề xuất, bao gồm những thông số cấu hình mà bài báo đã rút ra được từ một số thử nghiệm kiểm tra để hoàn thiện phương pháp trước khi thực hiện thí nghiệm chính.

Mô hình học máy phân loại mà phương pháp áp dụng để xem như là mô hình mục tiêu và là mô hình để thử nghiệm đánh giá trong suốt quá trình tạo mẫu đối kháng là MalConv [2]. MalConv là một mô hình học máy phân loại mã độc, mô hình sử dụng mạng nơ-rôn tích chập (CNN) thuộc học sâu và dựa trên các thuộc tính phân tích tĩnh các tệp thực thi. Với điều kiện Black-box, mô hình sẽ chỉ cho ra một chỉ số phân loại (CSPL) có giá trị từ 0 đến 1 (0 đến dưới 0.5 là lành tính, 0.5 đến 1 là độc hại).

Bước 0: malware đầu vào (tệp .exe) trước khi vào quá trình sẽ được kiểm tra phải là một tệp thực thi PE hợp lệ và có tính độc hại (CSPL từ 0.5 trở lên).

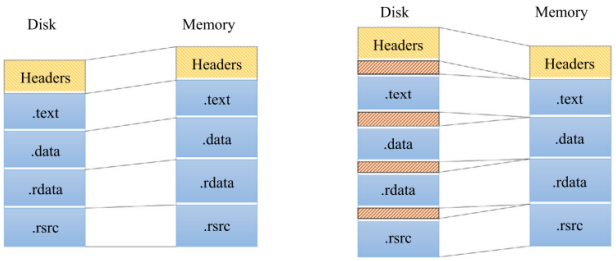

Figure 2: Mở rộng vị trí.

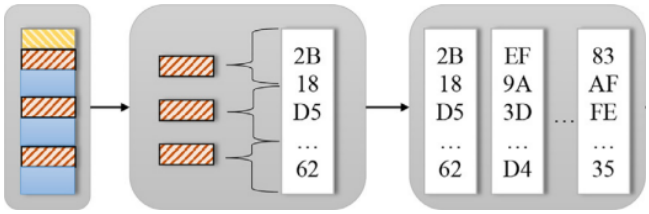

Figure 3: Tạo quần thể.

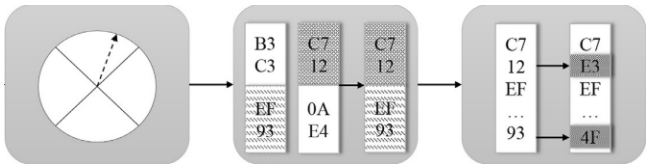

Figure 4: Thuật toán tiến hóa di truyền.

Bước 1: malware hợp lệ được đưa vào quá trình, bắt đầu với bước mở rộng vị trí. Như đã nói, phương pháp tận dụng vị trí giữa các sections trong tệp thực thi PE.

Cụ thể, với cơ chế phân trang của hệ điều hành, bắt đầu mỗi sections phải là bắt đầu của mỗi trang, điều này nhằm đảm bảo khi tải tệp (từng sections theo trang) vào bộ nhớ RAM sẽ không gây mất dữ liệu. Điều đó dẫn đến, ở section trước đó sẽ có khoảng trống cuối trang cuối không có dữ liệu để có thể chèn nội dung nhiễu vào.

Bên cạnh đó, với cơ chế tải các sections vào bộ nhớ RAM dựa theo sections table ở header của tệp PE, tức là chỉ dựa theo địa chỉ bắt đầu section và độ dài kích thước nội dung thực tế của section đó mà không tải phần khoảng trống đã đề cập ở trên. Tận dụng điều đó, có thể mở rộng thêm khoảng trống giữa các sections với ngưỡng kích thước phù hợp, mà khi chạy tệp, những nội dung được chèn vào đó không bị tải lên RAM, điều đó giúp không ảnh hưởng đến chức năng độc hại gốc của tệp mã độc. *(Hình 2)*

Lần lượt các thao tác sẽ được thực hiện để thay đổi mở rộng tệp thực thi PE. Bao gồm, thay đổi thông số địa chỉ bắt đầu của các sections trong sections table ở header của tệp và dịch chuyển các sections theo địa chỉ bắt đầu mới để mở rộng khoảng trống chèn nhiễu.

Bài báo tính toán được kích thước tổng phù hợp của các khoảng trống khi khởi tạo là 1% trên tổng kích thước tệp mã độc gốc. Đồng thời, nội dung nhiễu khởi tạo ở những vị trí đó được thiết lập là ngẫu nhiên.

Điều kiện rẽ nhánh 1: mẫu đã chỉnh sửa mở rộng sẽ được đưa vào mô hình phân loại đánh giá. Nếu mẫu vẫn còn bị phát hiện tính độc hại (CSPL từ 0.5 trở lên) , tiếp tục đưa đến bước tiếp theo. Ngược lại, không bị phát hiện (CSPL nhỏ hơn 0.5), mẫu đối kháng đã tạo THÀNH CÔNG và kết thúc quá trình (đây là trường hợp ít xảy ra ở vòng lặp đầu tiên của quá trình).

Bước 2: mẫu được đưa vào các bước của thuật toán tiến hóa di truyền (GA) để lựa chọn nội dung nhiễu.

Bước 2.1: ở bước 1, nội dung nhiễu được khởi tạo ngẫu nhiên. Bước này sẽ tạo ngẫu nhiên ra thêm 49 bộ nội dung nhiễu khác tương ứng với kích thước nhiễu hiện tại, cộng với bộ gốc, tổng cộng tạo được quần thể ban đầu gồm 50 bộ nội dung nhiễu (50 cá thể). *(Hình 3)*

Điều kiện rẽ nhánh 2: tương tự điều kiện rẽ nhánh 1, từng cá thể trong quần thể (đã chuyển về dạng mẫu tệp thực thi) sẽ được đưa vào mô hình phân loại đánh giá và xét điều kiện CSPL. Bất kỳ cá thể nào đạt điều kiện không bị phát hiện đều sẽ kết thúc quá trình và tạo mẫu THÀNH CÔNG. Nếu không, sẽ đến bước tiếp theo. (Rẽ nhánh để thoát vòng lặp sẽ được đề cập sau) **Bước 2.2:** tiếp tục đi vào 3 bước chính của thuật toán GA. *(Hình 4)*

- Chọn lọc: từ kết quả đánh giá (CSPL) ở điều kiện rẽ nhánh 2, chọn lọc ra một số lượng các cá thể tốt nhất (CSPL nhỏ nhất) để giữ lại cho quần thể.
- Lai ghép: từ những cá thể trên, đôi một các cá thể sẽ được lai ghép các phần với nhau để tạo ra những cá thể con, nhằm bù đắp số lượng cho quần thể (đảm bảo 50 cá thể trong quần thể).
- Đột biến: với quần thể đầy đủ trên, một số lượng ngẫu nhiên cá thể sẽ được đột biến, tức là biến đổi một số bytes trong từng cá thể. Mục đích là nhằm tạo được quần thể sau có sự khác biệt, tiến hóa so với quần thể trước. Quần thể mới trở lại bước 2.1 và bắt đầu lại vòng lặp mới của thuật toán GA.

Sau khi trải qua 50 vòng lặp GA, nếu chưa xảy ra điều kiện không bị phát hiện (tạo mẫu THÀNH CÔNG), điều kiện rẽ nhánh 2 sẽ xét thêm điều kiện bảo hòa. Điều kiện bảo hòa sẽ xét xem ở 10 vòng lặp gần nhất, CSPL của cá thể tốt nhất (CSPL nhỏ nhất) trong quần thể của vòng lặp sau có cải thiện hơn vòng lặp trước 1% hay không. Nếu không, quần thể đã bị bảo hòa và kết thúc thuật toán GA.

Điều kiện rẽ nhánh 3: sau khi thoát khỏi thuật toán GA, quá trình xét xem kích thước hiện tại của nội dung nhiễu (được mở rộng ở bước 1) đã đạt ngưỡng tối đa chưa. Nếu kích thước đã lớn hơn 100% trên tổng kích thước tệp mã độc gốc, quá trình tạo mẫu kết thúc và THẤT BẠI tạo mẫu đối kháng. Ngược lại, nếu chưa chạm ngưỡng trên, quá trình tiếp tục bằng cách tăng kích thước nội dung nhiễu lên, mỗi lần tăng 3% trên kích thước tệp mã độc gốc và trở lại bước 1, bắt đầu lại vòng lặp mới của quá trình tạo mẫu đối kháng.

Quá trình sẽ lặp lại đến khi có điều kiện THÀNH CÔNG hoặc chạm ngưỡng THẤT BẠI.

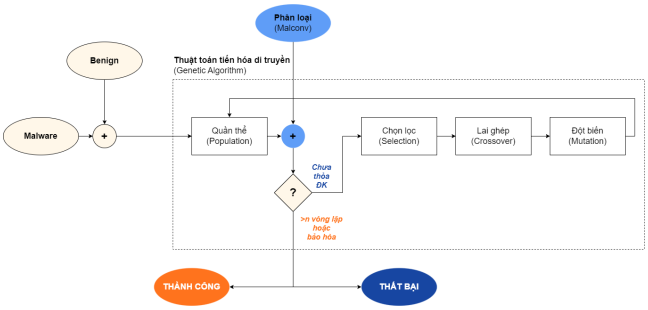

Figure 5: Mô hình phương pháp GAMMA.

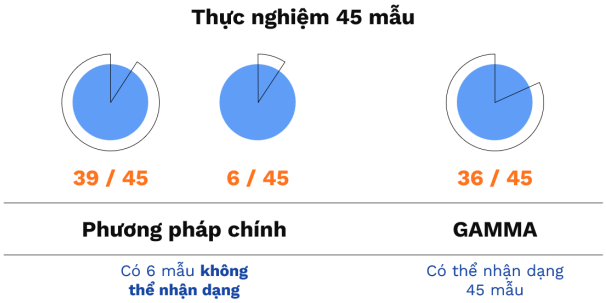

Figure 6: Thí nghiệm 1.

3 THỰC NGHIỆM

Nhóm thực hiện triển khai lại phương pháp tạo mẫu đối kháng tương tự bài báo và triển khai thêm phương pháp GAMMA [5], nhằm so sánh với phương pháp chính.

GAMMA là phương pháp tạo mẫu mã độc đối kháng trong điều kiện Black-box, phương pháp cũng ứng dụng thuật toán tiến hóa di truyền (GA) và cũng lấy MalConv làm mô hình đánh giá. Tuy nhiên, GAMMA không có phần mở rộng vị trí giữa các sections và không có phần tăng kích thước nhiễu để tiếp tục vòng lặp mới (bước 1). Thay vào đó sẽ chèn nội dung nhiễu theo hai kỹ thuật, Sections Injection và Padding, và quần thể ban đầu của thuật toán GA sẽ được tạo bằng cách trích xuất các nội dung từ những tệp thực thi lành tính, thay vì ngẫu nhiên (bước 2.1). Vòng lặp của thuật toán GA sẽ diễn ra đến khi, hoặc là đạt số vòng lặp tối đa, hoặc là đạt điều kiện bảo hòa (CSPL không cải thiện đáng kể), thay vì sẽ kết thúc ngay khi tìm được một mẫu không bị phát hiện (CSPL nhỏ hơn 0.5) (điều kiện rẽ nhánh 2). Tức là GAMMA sẽ có hạn chế về vị trí và kích thước nội dung nhiễu, nhưng lại có lợi thế về quần thể lành tính và tối ưu CSPL. *(Hình 5)*

Để đạt được sự công bằng cho so sánh, thực nghiệm cho 2 phương pháp được thực hiện trong cùng điều kiện và môi trường. Triển khai trên máy ảo Ubuntu 20.04 4GB RAM, tập dữ liệu cho các mẫu mã độc tệp thực thi PE được lấy từ DikeDataset [4] và mô hình MalConv đã training sẵn bằng tập dữ liệu EMBER được lấy từ Github [2]. Mã nguồn của 2 phương pháp được tác giả công bố trên Github [3] [1] và triển khai bằng Python 3.

- **Thí nghiệm thứ 1:** Thí nghiệm thứ nhất, thử nghiệm cả 2 phương pháp với 45 mẫu tệp thực thi PE thuộc tập dữ liệu. Kết quả cho thấy, ở phương pháp chính (PP chính), có 39 mẫu đối kháng được tạo thành công trên 45 mẫu đầu vào. Tuy nhiên, sau khi kiểm tra lại 6 mẫu không thành công đó, kết quả cho thấy PP chính không thể nhận dạng được 6 mẫu đó là tệp PE hợp lệ. Tức là với 39 mẫu PP chính nhận dạng được, nó có thể tạo thành công cho toàn bộ các mẫu. Trong khi đó, thí nghiệm với GAMMA, có 36 mẫu đối kháng được tạo thành công và GAMMA lại có thể nhận dạng được tất cả 45 mẫu là tệp PE hợp lệ. (Điều này có thể là do điều kiện kiểm tra hợp lệ của 2 phương pháp khác nhau, nhóm chưa thể nghiên cứu sâu vào mã nguồn để đưa ra kết luận chi tiết) *(Hình 6)*
  Vì thế, để có thể so sánh công bằng, nhóm thực hiện thêm thí nghiệm thứ 2.

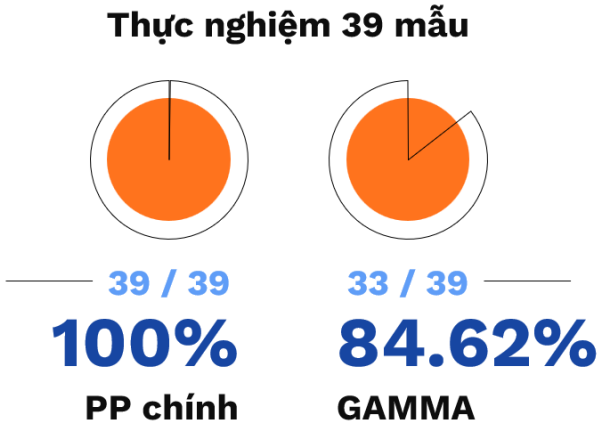

Figure 7: Thí nghiệm 2.

- **Thí nghiệm thứ 2:** Thí nghiệm thứ 2, thử nghiệm lại GAMMA chỉ với 39 mẫu mà PP chính có thể nhận dạng. Kết quả cho thấy, có 33 mẫu đối kháng được tạo thành công trên 39 mẫu đầu vào. *(Hình 7)*
  Tổng kết, PP chính tạo thành công 39/39 mẫu (100%) trong thời gian 6 giờ 20 phút và CSPL trung bình của các mẫu 0.4. GAMMA tạo thành công 33/39 mẫu (84.62%) trong thời gian 23 phút và CSPL trung bình của các mẫu 0.1. *(Hình 8)*

4 ĐỀ XUẤT CẢI THIỆN VÀ HƯỚNG PHÁT TRIỂN

Từ Hình 6, bảng so sánh giữa thực nghiệm của 2 phương pháp cho thấy: tỉ lệ tạo mẫu đối kháng thành công của PP chính gần như tuyệt đối và cao hơn GAMMA. Tuy nhiên, với thời gian thực hiện quá lâu (gấp 21 lần so với GAMMA), sự đánh đổi về tỉ lệ thành công trở nên không tối ưu. Hơn nữa, CSPL trung bình của GAMMA được cải thiện hơn nhờ vào trích xuất nhiễu từ tệp lành tính và cơ chế thoát vòng lặp. Nhưng sự tối ưu về vị trí chèn giữa các sections và kích thước nhỏ nhất đạt được trở thành một ưu thế khó phủ định của PP chính.

Thực nghiệm	PP chính	GAMMA
Số mẫu tạo được	**39 / 39**	**33 / 39**
Tỉ lệ thành công	**100%**	**84.62%**
Thời gian	**6 giờ 20 phút**	**23 phút**
Chỉ số phân loại	**~0.4**	**~0.1**

Figure 8: So sánh thực nghiệm.

Cả 2 phương pháp đều có điểm mạnh, yếu khác nhau, cộng với mô hình thiết kế quy trình khá tương đồng. Vì thế, nhóm đề xuất kết hợp cả 2 mô hình lại với nhau cho hướng phát triển trong tương lai. Điều đó tận dụng được lợi thế tối ưu hóa về vị trí, kích thước và tỉ lệ thành công của PP chính, đồng thời, lợi thế từ việc trích xuất nội dung nhiễu lành tính và thời gian xử lý của GAMMA.

REFERENCES

[1] [n. d.]. *GAMMA.* https://github.com/pralab/secml_malware

[2] [n. d.]. *Mô hình học máy phân loại MalConv.* https://github.com/elastic/ember/blob/master/malconv/malconv.h5

[3] [n. d.]. *Phương pháp chính.* https://github.com/JavierYuste/Optimization-of-code-caves-in-malware-binaries-to-evade-Machine-Learning-detectors

[4] [n. d.]. *Tập dữ liệu DikeDataset.* https://github.com/iosifache/DikeDataset

[5] Luca Demetrio, Battista Biggio, Giovanni Lagorio, Fabio Roli, and Alessandro Armando. 2021. Functionality-Preserving Black-Box Optimization of Adversarial Windows Malware. *IEEE Transactions on Information Forensics and Security* 16 (2021), 3469–3478. https://doi.org/10.1109/TIFS.2021.3082330

[6] Javier Yuste, Eduardo G. Pardo, and Juan Tapiador. 2022. Optimization of code caves in malware binaries to evade machine learning detectors. *Computers Security* 116 (2022), 102643. https://doi.org/10.1016/j.cose.2022.102643

A PHÂN CÔNG CÔNG VIỆC

- Đọc bài báo: tất cả các thành viên
- Tổng hợp, tóm tắt nội dung báo cáo: Nguyễn Mạnh Cường
- Làm Proposal: Hoàng Văn Anh Đức
- Triển khai thực nghiệm: Hoàng Văn Anh Đức
- Làm slides, thuyết trình: Nguyễn Mạnh Cường
- Làm Poster: Nguyễn Mạnh Cường
- Viết báo cáo: Nguyễn Mạnh Cường, Hoàng Văn Anh Đức