Phân loại hành vi phần mềm độc hại trên các thiết bị IoT dựa vào system call

Hoàng Đăng Kiên, Nguyễn Đại Thọ, Vũ Duy Lợi

Đại học Công nghệ, Đại học quốc gia Hà Nội

Email: {15021363, nguyendaitho}@vnu.edu.vn, vuduyloi55@gmail.com

Tóm tắt-- Phần mềm độc hại từ lâu đã là một trong các mối nguy hại lớn nhất đối với các hệ thống máy tính, nay đã chuyển sang cả các thiết bị IoT. Trong bài báo này chúng tôi đề xuất sử dụng thuật toán FRNN để thay thế cho thuật toán 1-NN trong quy trình đề xuất bởi Rieck và các cộng sự [2]. Kết quả thực nghiệm cho thấy phương pháp của chúng tôi đề xuất mang lại hiệu quả phân lớp cao đối với các mã độc đã biết với độ đo micro-F_1 lên tới trên 98% trong khi vẫn duy trì khả năng phát hiện các loại mã độc mới lên tới 97%, khả năng chống lại ảnh hưởng của dữ liệu nhiễu so với phương pháp gốc được tăng lên đáng kể.

***Từ khóa*-- Phần mềm độc hại, IoT, system call.**

I. GIỚI THIỆU

Cuộc cách mạng công nghiệp lần thứ tư có thể được hiểu như là sự phát triển vượt bậc của một loạt các công nghệ mới, xóa nhòa ranh giới giữa lĩnh vực vật lý, kỹ thuật số, sinh học, và ảnh hưởng đến kinh tế, xã hội, giáo dục và cả chính trị. Các lĩnh vực chính đang được quan tâm trong cuộc cách mạng 4.0 bao gồm: Robotics, trí thông minh nhân tạo, công nghệ nano, công nghệ sinh học, Internet vạn vật (IOT), in 3D, và xe tự hành. Trong đó các thiết bị IoT đóng vai trò vô cùng quan trọng. Theo thống kê từ trang web https://iot-analytics.com, trên thế giới hiện nay đã có hơn 17 tỉ thiết bị kết nối Internet trong đó có hơn 7 tỉ thiết bị IoT trong đó điển hình là CameraIP, VoIp, smart TV, thiết bị định tuyến, … Các thiết bị IoT có sự đa dạng về kiến trúc vi xử lý như MIPS, MIPSEL, ARM, PowerPC, SuperH, SPARC…, và hệ điều hành phổ biến nhất là Linux [6]. Các thiết bị này thường có tài nguyên hạn chế và được thiết kế cho một chức năng chuyên biệt.

Do việc cạnh tranh về sự phát triển công nghệ, các nhà sản xuất đã không chú trọng tới an ninh của các thiết bị IoT, khiến cho các thiết bị dễ dàng bị khai thác thông qua các lỗ hổng bảo mật [3]. [9] thực hiện phân tích 32 356 firmware trên các thiết bị nhúng và đã phát hiện ra 38 lỗ hổng bảo mật mới. Theo thống kê của Kaspersky, số lượng mã độc thu thập được trên các thiết bị IoT trong nửa đầu năm 2018 gấp 3 lần năm 2017 và 10 lần so với năm 2016. Cách tấn công và lây nhiễm chủ yếu là bẻ khóa mật khẩu Telnet và SSH đối với các thiết bị cấu hình mật khẩu yếu [1]. Năm 2016, mã độc Mirai đã lợi dụng lỗ hổng này để gây ra cuộc tấn công từ chối dịch vụ phân tán lớn nhất lịch sử với lưu lượng mạng lên tới 620Gbps với hơn 380000 thiết bị lây nhiễm tham gia.

Nghiên cứu về mã độc trên các thiết bị IoT đang được quan tâm sâu sắc trong lĩnh vực an toàn thông tin. Các chủ đề nghiên cứu trước đây hầu như tập trung vào mã độc trên hệ điều hành Windows và kiến trúc vi xử lý của Intel và gần đây là các thiết bị di động. Các đề tài nghiên cứu về mã độc trên các thiết bị IoT còn hạn chế. Detux [5] cung cấp môi trường phân tích động phục vụ cho thu thập mã độc tuy nhiên các kết quả thu được mới chỉ có dữ liệu mạng (tập tin pcap) và thông tin phân tích tĩnh cơ bản. [7] phân tích hành vi mã độc trên IoT tuy nhiên chỉ tập trung vào mã độc Mirai và hành vi đặc trưng đó là quét cổng. [8] phân tích mã độc trên kiến trúc ARM và cách tiếp cận phân lớp không có khả năng phát hiện ra các lớp phần mềm độc hại mới.

Trước thực trạng đó, chúng tôi tiến hành nghiên cứu xây dựng và đánh giá quy trình phân tích hành vi mã độc trên các thiết bị IoT dựa trên các system call. Báo cáo của chúng tôi được xây dựng dựa trên các phương pháp được đưa ra bởi bài báo [2]. Hai bước chính trong quy trình phân tích của đó là: phân lớp nhằm dự báo nhãn lớp đối với các mã độc thuộc một họ đã biết , phân cụm

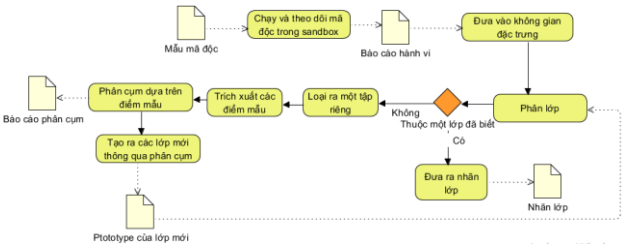
giúp gom nhóm hành vi các mã độc chưa biết giống nhau tạo thành lớp mới và cập nhật tự động vào bộ phân lớp. Chúng tôi đề xuất sử dụng thuật toán FRNN để thay thế cho thuật toán 1-NN nhằm giảm thiểu ảnh hưởng của các dữ liệu nhiễu. Kết quả thực nghiệm cho thấy phương pháp của chúng tôi đề xuất mang lại hiệu quả phân lớp cao đối với các mã độc đã biết với độ đo micro-F_1 lên tới trên 98% trong khi vẫn duy trì khả năng phát hiện các loại mã độc mới lên tới 97%, khả năng chống lại ảnh hưởng của dữ liệu nhiễu so với phương pháp gốc được tăng lên đáng kể

II. CÔNG TRÌNH LIÊN QUAN

Năm 2011, Rieck và các cộng sự đã áp dụng quy trình tự động phân tích hành vi mã độc dựa trên các system call sử dụng kết hợp phân cụm và phân lớp. Trong đó phân lớp giúp gán nhãn cho các mã độc đã biết và phát hiện ra các mã độc chưa biết. Phân cụm giúp gom những mã độc chưa biết có hành vi giống nhau tạo thành một cụm. Các cụm đủ về mặt số lượng sẽ đươc coi như một lớp mã độc mới và được tự động cập nhật vào bộ phân lớp. Các tham số tối ưu cho phân lớp và phân cụm sẽ được tìm ra thông qua bộ dữ liệu có nhãn. Sau khi có được các tham số, việc phân tích một mã độc mới được thực hiện theo quy trình như ***hình 1***.

Đầu tiên các mã độc sẽ được chạy trong môi trường sandbox để thu thập các system call trong thời gian thực thi, sau đó mỗi mã độc sẽ được biểu diễn bởi 1 điểm trong không gian vector. Các điểm sẽ được đưa qua bộ phân lớp để phân loại, nếu mã độc thuộc một lớp đã biết thì sẽ được đưa ra nhãn lớp, nếu mã độc thuộc một lớp chưa biết thì sẽ đưa vào ra một tập riêng. Việc phân cụm sẽ được thực hiện trên tập mã độc chưa biết để tìm ra các mã độc có hành vi giống nhau và đưa chúng vào 1 cụm. Các cụm đủ về mặt số lượng sẽ được coi như một họ mã độc chưa biết tới và sẽ được cập nhật tự động vào bộ phân lớp. Trong bài báo cáo này chúng tôi tập chung vào cải tiến giải thuật phân lớp của tác giả nên sẽ chỉ tập chung giới thiệu vào giải thuật này.

Các báo cáo về system call sẽ được thu thập thông



Hình 1: Quy trình phân loại các mã độc mới

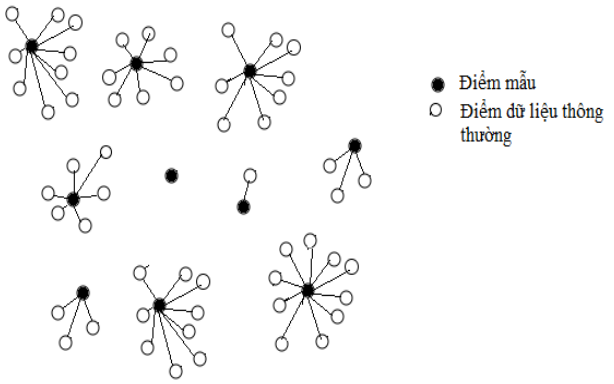
qua CWSandbox. Các báo cáo này sau đó sẽ được biểu diễn trong không gian vector bằng cách sau: Giả sử có tất cả n system call có thể thì vector sẽ có n^2 chiều tương ứng với số lượng tất cả 2-gram system call có thể, mỗi chiều sẽ đại diện cho một 2-gram. Nếu 2-gram có xuất hiện trong chuỗi system call của mã độc thì chiều tương ứng với 2-gram đó sẽ là 1, ngược lại thì sẽ là 0. Vector sau đó sẽ được chuẩn hóa bằng cách chia tất cả các chiều cho độ dài của chính vector đó. Kết quả sẽ thu được các vector có độ dài là 1, chúng ta sẽ kéo gốc vector về gốc tọa độ và lấy điểm đầu của vector làm điểm đại diện cho mã độc.

Sau khi đã có điểm biểu diễn mã độc trong không gian vector, việc phân lớp mã độc sẽ được thực hiện theo thuật toán 1-NN. Nhược điểm của thuật toán k-NN nói chung và 1-NN nói riêng đó là thời gian phân lớp với một dữ liệu mới là lớn. Để giảm thời gian phân lớp dữ liệu mới, kỹ thuật trích xuất điểm nguyên mẫu được sử dụng để chọn ra một số điểm đại diện cho tập dữ liệu đã biết nhãn. Việc tính toán để phân lớp cho dữ liệu mới sẽ được thực hiện trên các điểm nguyên mẫu chứ không phải cả tập dữ liệu. Giải thuật này được đề xuất bởi Gonzalez. 1985 [13]. Chi tiết thuật toán được trình bày tại **thuật toán 1.**

Thuật toán có một tham số ngưỡng d_p là khoảng cách tối đa giữa một điểm dữ liệu bất kì và một điểm nguyên mẫu gần nhất với nó. Các điểm nguyên mẫu sẽ được lần lượt chọn ra từ tập dữ liệu cho đến khi khoảng cách từ một dữ liệu điểm bất kì với điểm mẫu gần nhất với nó nhỏ hơn d_p. Các điểm dữ liệu sẽ được đại diện bởi điểm nguyên mẫu gần nó nhất. Minh họa về việc trích xuất điểm mẫu được thể hiện tại **hình 2**.

Thuật toán 1: Trích xuất điểm nguyên mẫu

1: *prototypes* ← ∅
2: *distance*[x]← ∞ for all x ∈ *reports*
3: **while** (1) **do**
4: find z that *distance*[z]=max(*distance*)
5: **if** (distance[z] <=d_p) **then break**
6: **for** x∈ *reports* and x!=z **do**
7: **if** *distance*[x] > d(x,z) **then**
8: *distance*[x]=d(x,z)
9: add z to *prototypes*



Hình 2: Minh họa việc trích xuất điểm mẫu

Toàn bộ các chấm trên *hình 2* biểu diễn cho cả tập dữ liệu, các điểm đen là các điểm nguyên mẫu và các điểm trắng là các điểm dữ liệu thông thường. Các điểm trắng được nối với điểm đen gần nó nhất, và sẽ được đại diện bởi điểm đen đó. Sau khi trích chọn được các điểm đen, các điểm trắng sẽ được loại bỏ và không có ý nghĩa gì đối với phân lớp một dữ liệu mới. Sau quá trình lựa chọn, việc thực hiện thuật toán phân lớp một dữ liệu mới được thực hiện như **thuật toán 2.**

Thuật toán có một tham số đó là d_r là khoảng cách tối đa cho phép tới điểm mẫu gần nhất để mã độc được coi là thuộc một lớp đã biết. Với mỗi mã độc mới, thuật toán sẽ tìm ra điểm nguyên mẫu gần nhất, nếu khoảng cách tới điểm gần nhất lớn hơn ngưỡng d_r thì chứng tỏ độ khác biệt với các mã độc đã biết là lớn do đó mã độc sẽ được gán nhãn là mã độc chưa biết, ngược lại dữ liệu sẽ được gán nhãn bằng với nhãn của điểm nguyên mẫu gần nhất. Nếu d_r lớn thì nhiều mã độc chưa biết sẽ được gán vào một họ mã độc đã biết, nếu d_r quá nhỏ thì nhiều mã độc thuộc các họ đã biết sẽ được coi như là mã độc chưa biết. Thuật toán này dựa trên k-NN với k=1 và có thêm một chút chỉnh sửa đó là bổ xung thêm ngưỡng d_r để phát hiện ra các mã độc mới. Độ phức tạp cho mỗi lần phân lớp mã độc mới là O(N) với N là số điểm nguyên mẫu, nếu áp dụng cách lưu trữ các điểm nguyên mẫu dưới dạng K-D tree thì độ phức tạp sẽ giảm xuống còn O(log N) [10].

Thuật toán 2: Phân lớp một mã độc mới

1: **for** x ∈ *reports* **do**
2: z← nearest prototype to x
3: **if** d(z, x) > d_r **then**
4: reject x as unknown class
5: **else**
6: assign label of x equal to label of z

III. ĐÓNG GÓP CỦA CHÚNG TÔI

Chúng tôi sẽ tiến hành áp dụng và đưa ra cải tiến thuật toán phân lớp được sử dụng bởi tác giả [2] trên dữ liệu về mã độc trên các thiết bị IoT. Một nhược điểm thuật toán của tác giả [2] đó là việc nhạy cảm với dữ liệu nhiễu về nhãn lớp. Sự nhạy cảm với dữ liệu nhiễu của thuật toán k-NN với k=1 đã được thử nghiệm bởi bài báo [12] trên nhiều bộ dữ liệu khác nhau. Các nhà nghiên cứu luôn cố gắng gán nhãn mã độc một cách chính xác để có được dữ liệu tốt cho việc học tuy nhiên việc đó gần như không thể. Bailey và các cộng sự đã chỉ ra rằng hầu hết tất cả các antivirus đều cung cấp nhãn lớp không hoàn hảo cho huấn luyện [11], vì vậy chúng tôi xem xét trường hợp các nhãn lớp dữ liệu bị nhiễu về nhãn lớp. Để giải quyết vấn đề đó, chúng tôi xem xét nhiều điểm nguyên mẫu chứ không phải riêng 1 điểm nguyên mẫu, chúng tôi phân lớp dựa trên tất cả các điểm nguyên mẫu, chúng tôi phân lớp dựa trên tất cả các điểm nguyên mẫu đã biết trong khu vực bán kính d_r, thuật toán này có tên là **Fixed Radius Nearest Neighbor (FRNN)** [15], chi tiết giải thuật xem tại **thuật toán 3.**

Thuật toán 3: Fixed Radius Nearest Neighbor

1: **for x** ∈ *reports* **do**
2: *neighbors* ←{ z : d(x, z) < d_r }
3: **if** neighbors is ∅ **then**
4: reject x as unknown class
5: **else**
6: assign x to the label that biggest number of neighbors belong to

IV. THỰC NGHIỆM VÀ ĐÁNH GIÁ

A. Thu thập và tiền xử lý dữ liệu

- Các mẫu mã độc trên các thiết bị IoT được chúng tôi thu thập ở nhiều nguồn khác nhau bao gồm Detux.com, VirusShare.com và IoTPot [4]. Sau khi loại bỏ các dữ liệu trùng lặp các dữ liệu còn lại hơn 3900 mẫu.
- Các dữ liệu system call được thu thập trong quá trình thực thi thông qua sandbox xây dựng bởi chính chúng tôi. Các system call phát sinh được thu thập bằng công cụ Strace, môi trường để chạy các mã độc trên kiến trúc vi xử lý của các thiết bị IoT được giả lập bằng QEMU, môi trường Internet được mô phỏng bằng INetSim. Các kết quả đã được chúng tôi giới thiệu tại [14]. Tổng cộng chúng tôi thu được hơn 2200 các báo cáo system call của 6 lớp mã độc trên các thiết bị IoT từ 3900 mẫu thu thập được ở trên.
- Các dữ liệu sẽ được gán nhãn bằng 4 hãng antivirus nổi tiếng đó là: Kaspersky, Avast, Avira, Symantec.
- Cách trích xuất điểm biểu diễn mã độc trong không gian vector được thực hiện dựa trên 2-gram các system call theo [2] như đã trình bày ở trên.
- Các điểm ngoại lai sẽ được phát hiện và loại bỏ khỏi tập dữ liệu bằng thuật toán DBSCAN [15].
- Các lớp có số lượng dữ liệu lớn sẽ được loại bỏ bớt để đảm bảo nhãn lớp không lệch nhau quá nhiều. Lớp ít nhất có 13 dữ liệu, các lớp nhiều nhất sẽ được lấy tối đa 300 mẫu dữ liệu. Sau khi loại bỏ bớt, bộ dữ liệu của chúng tôi còn 929 mẫu với 6 lớp mã độc.

B. Đánh giá

a. Các Độ đo cho đánh giá phân lớp

Độ đo sử dụng cho đánh giá phân lớp là độ đo $\boldsymbol{F1_{micro}}$. Độ đo $\boldsymbol{F1_{micro}}$ được định nghĩa dựa trên các khái niệm sau:

- $\boldsymbol{TP_i}$: Các điểm dữ liệu thuộc lớp i và được gán đúng vào lớp i
- $\boldsymbol{TN_i}$: Các điểm dữ liệu không phải nhãn i và được gán nhãn không phải nhãn i
- $\boldsymbol{FP_i}$: Các điểm dữ liệu không thuộc nhãn i nhưng lại được gán nhãn i
- $\boldsymbol{FN_i}$: Các điểm dữ liệu thuộc nhãn i nhưng lại được gán không phải nhãn i

Độ chính xác trung bình mịn trên toàn bộ các lớp:

$$\boldsymbol{P_{micro}}=\frac{\sum \boldsymbol{TP_i}}{\sum(\boldsymbol{TP_i}+\boldsymbol{FP_i})}$$

Độ chính hồi tưởng bình mịn trên toàn bộ các lớp:

$$\boldsymbol{R_{micro}}=\frac{\sum \boldsymbol{TP_i}}{\sum(\boldsymbol{TP_i}+\boldsymbol{FN_i})}$$

Độ đo **F** trên toàn bộ các lớp:

$$\boldsymbol{F1_{micro}}=2\cdot\frac{\boldsymbol{P_{micro}}\cdot\boldsymbol{R_{micro}}}{\boldsymbol{P_{micro}}+\boldsymbol{R_{micro}}}$$

b. Đánh giá phân lớp

Tập dữ liệu sẽ được chia làm tập huấn luyện và đánh giá. Chúng tôi tiến hành chọn ra các điểm nguyên mẫu để đại diện cho cho tập huấn luyện theo thuật toán trích xuất điểm nguyên mẫu được đề xuất bên trên. Để đánh giá khả năng phát hiện các mã độc chưa biết, mỗi lần thử chúng tôi coi một lớp mã độc là chưa biết và không cung cấp ví dụ cho tập huấn luyện. Hai độ đo được sử dụng để đánh giá là:

- $\mathbf{F_k}$ là độ đo $\boldsymbol{F1_{micro}}$ được tính trên phần mã độc đã biết của tập đánh giá để đánh giá khả năng phân lớp đúng mã độc đã biết (Thuộc các lớp đã được cung cấp ví dụ cho tập huấn luyện).
- $\mathbf{F_k}$ là độ đo $\boldsymbol{F1_{micro}}$ được tính trên phần mã độc chưa biết của tập đánh giá (Thuộc các lớp không được cung cấp ví dụ cho tập huấn luyện), nhằm đánh giá khả năng phát hiện mã độc chưa biết, chỉ có 2 quyết định được xem xét

đó là coi như chưa biết và coi như đã biết (tức là phân vào 1 trong các lớp đã biết).

Ngưỡng d_p dùng cho thuật toán trích xuất nguyên mẫu được chọn là 0,8 theo cách chọn được giới thiệu bởi [2]. Sau đó các dữ liệu trong tập đánh giá sẽ được gán nhãn chỉ dựa vào các điểm nguyên mẫu trích xuất từ tập huấn luyện theo thuật toán phân lớp 1-NN và FRNN. Ngưỡng d_r được thay đổi ở nhiều mức khác nhau. Kết quả được cho bởi **bảng 1**.

Cả hai thuật toán đều cho kết quả tốt nhất tại ngưỡng d_r=0.75 và kết quả hai thuật toán có sự giống nhau, nguyên nhân là do khi áp dụng thuật toán trích xuất các điểm mẫu, mật độ các điểm nguyên mẫu là thưa (khoảng cách giữa các nguyên mẫu luôn lớn hơn d_p) nếu d_r nhỏ thì trong khu vực bán kính d_r xung quanh một điểm thường là chỉ tồn tại một nguyên mẫu do đó thuật toán FRNN và 1-NN cho kết quả giống nhau. Khi d_r lớn ta có thể tìm được nhiều điểm mẫu xung quanh điểm đang xét, khi đó kết quả hai thuật toán có sự khác biệt. Tuy nhiên khi d_r lớn nhiều mã độc chưa biết sẽ được coi như thuộc một loại mã độc đã biết và ảnh hưởng tới F_u.

	1-NN		FRNN	
d_r	F_k	F_u	F_k	F_u
0.7	0.8820	0.9720	0.8820	0.9720
0.75	0.9572	0.9698	0.9572	0.9698
0.8	0.9730	0.8734	0.9702	0.8734
0.85	0.9799	0.7854	0.9739	0.7854
0.9	0.9843	0.7359	0.9769	0.7359

Bảng 1: So sánh thuật toán 1-NN và thuật toán FRNN trong trường hợp có sử dụng các điểm mẫu

Việc lựa chọn các điểm nguyên mẫu để đại diện cho tập huấn luyện có thể gây mất mát về thông tin và ảnh hưởng tới hiệu suất phân lớp. Chúng tôi sẽ thử nghiệm ảnh hưởng của việc dùng các điểm mẫu để đại diện cho tập huấn luyện. Tại thử nghiệm này chúng tôi sẽ dự đoán nhãn lớp cho tập đánh giá bằng toàn bộ các điểm

	1-NN		FRNN	
d_r	F_k	F_u	F_k	F_u
0.65	0.9812	0.9739	0.9844	0.9739
0.7	0.9863	0.9728	0.9895	0.9728
0.75	0.9887	0.9689	0.9919	0.9689
0.8	0.9907	0.9625	0.9939	0.9625
0.85	0.9907	0.8467	0.9939	0.8467

Bảng 2: Kết quả của 1-NN và FRNN khi không sử dụng điểm mẫu

trong tập huấn luyện chứ không chỉ riêng các điểm mẫu. Kết quả được cho bởi **bảng 2**.

Với d_r =0.8 chúng tôi thấy kết quả $\boldsymbol{F_u}$ kém nên không tiếp tục thử d_r lớn hơn. Kết quả cho thấy hiệu suất của 2 thuật toán đều tốt và cao tương đương nhau tại ngưỡng tốt nhất d_r=0.7, độ đo $\boldsymbol{F_u}$ không có sự khác biệt, $\boldsymbol{F_k}$ có khác nhưng không đáng kể, sự so sánh về thống kê giữa 2 kết quả cho giá trị p-value > 0.05 theo kiểm định Wilcoxon tức là sự khác biệt không có ý nghĩa về thống kê.

Cuối cùng chúng tôi thử nghiệm sức mạnh của hai thuật toán đối với dữ liệu nhiễu. Các dữ liệu trong tập huấn luyện được ngẫu nhiên làm sai nhãn lớp một số phần trăm nhất định, nhiễu này gọi là nhiễu nhãn lớp đồng dạng (uniform class noise), còn một loại nhiễu phổ biến nữa đó là nhiễu cặp (pairwise class noise) vẫn đang trong quá trình thử nghiệm và đánh giá. Chúng tôi cung cấp thêm kiểm định thống kê Wilcoxon để so sánh kết quả được khách quan. Tham số d_r được cố định ở giá trị 0.7 tốt nhất đã tìm ra bên trên. Do việc làm sai nhãn lớp ngẫu nhiên không ảnh hưởng tới khả năng phát hiện mã độc mới nên chúng tôi chỉ xem xét $\boldsymbol{F_k}$. Kết quả được cho bởi **bảng 3**.

Kết quả cho thấy thuật toán FRNN mạnh mẽ hơn khi dữ liệu được cung cấp có sự sai lệch về nhãn lớp. Tức là khi các antivirus cung cấp các nhãn lớp sai một phần nhỏ thì thuật toán FRNN vẫn có thể mang lại kết quả dự báo nhãn lớp tốt đối với các mã độc đã biết.

Mức nhiễu	F_k (1-NN)	F_k (FRNN)	p-value
5%	89.28%	98.42%	3.51e-07
10%	85.41%	98.16%	5.677e-07
20%	78.71%	97.66%	1.563e-06

Bảng 3: Kết quả so sánh 1-NN và FRNN với các dữ liệu nhiễu

c. Một số hạn chế của thuật toán và cách đánh giá

Về mặt tốc độ: Việc sử dụng dữ liệu điểm mẫu giúp giảm số lượng dữ liệu huấn luyện cần xét đi 20 lần do đó tốc độ phân lớp sẽ giảm đi 20 lần nếu không dùng cấu trúc dữ liệu cây đối với cả 2 thuật toán. Khi biểu diễn dữ liệu huấn luyện dưới dạng cây được hỗ trợ trong thư viện sklearn của Python, tốc độ trung bình 30 lần chạy của 1-NN và FRNN sử dụng điểm mẫu trên bộ dữ liệu của chúng tôi là 0.09 giây, 1-NN không dùng điểm mẫu là 0.35 giây và FRNN không sử dụng điểm mẫu là 0.85 giây. Như vậy, khi số lượng dữ liệu lớn, chúng ta cần cân nhắc nên sử dụng điểm mẫu để giảm tốc độ xử lý và chấp nhận sự giảm về hiệu suất hay sẽ chọn thời gian phân lớp lớn.

Chênh lệch nhãn lớp: Một vấn đề khác đối với thuật toán FRNN đó là vấn đề chênh lệch nhãn lớp, nếu hai lớp khá gần nhau mà có số lượng chênh lệch nhau quá nhiều thì có thể gây phân lớp sai đối với việc phân lớp một dữ liệu thuộc lớp ít hơn.

Việc xem xét nhiễu: Chúng tôi vẫn chưa mô hình hóa được sự nhiễu nhãn lớp của các antivirus khi gán nhãn mã độc, chúng tôi chỉ thử nghiệm được một trong số hai loại quy luật nhiễu phổ biến.

KẾT LUẬN

Trong báo cáo chúng tôi đã giới thiệu và thực nghiệm hai phương pháp phân loại có khả năng phát hiện ra các mã độc chưa biết với độ hiệu quả cao. Thuật toán 1-NN mang lại hiệu quả tốt tuy nhiên lại nhạy cảm với các dữ liệu nhiễu. Thuật toán FRNN mạnh mẽ với nhiễu và mang lại hiệu quả tốt hơn. Tuy nhiên nhược điểm của cả hai phương pháp đều là về thời gian phân lớp đối với dữ liệu mới so với các phương pháp sử dụng mô hình. Trong tương lai, chúng tôi sẽ tiếp tục kiếm và phát triển ra các phương pháp phân lớp mới cải thiện về tốc độ và đảm bảo về mặt hiệu suất cũng như có khả năng phát hiện các mã độc chưa biết, việc phân cụm các mã độc chưa biết sẽ được tiếp tục phát triển để hoàn thiện quy trình.

TÀI LIỆU THAM KHẢO

[1] *https://securelist.com/new-trends-in-the-world-of-iot-threats/87991/*

[2] Konrad Rieck, Philipp Trinius, Carsten Willems, and Thorsten Holz "*Automatic Analysis of Malware Behavior using Machine Learning", Journal of Computer Security* (JCS), 19 (4) 639-668, 2011.

[3] Michele De Donno, Nicola Dragoni, Alberto Giaretta, Angelo Spognardi: "*DDoS-Capable IoT Malwares: Comparative Analysis and Mirai Investigation*". Security and Communication Networks 2018: 7178164:1-7178164:30 (2018)

[4] Pa YMP, Suzuki, S Yoshioka, K Matsumoto, TKasama, T Rossow, C 2016, *"IoTPOT: A novel honeypot for revealing current IoT threats*" Journal of Information Processing, vol 24, no. 3, pp. 522-533. DOI: 10.2197/ipsjjip.24.522

[5] Detux sandbox: *https://github.com/detuxsandbox/detux*

[6] https://www.itprotoday.com/iot/survey-showslinux-top-operating-system-internet-thingsdevices

[7] Naoki Hashimoto, Seiichi Ozawa, Tao Ban, Junji Nakazato, Jumpei Shimamura, "*A Darknet Traffic Analysis for IoT Malwares Using Association Rule Learn*ing", Procedia Computer Science, Volume 144, 2018, Pages 118-123

[8] Ensiyeh Modiri, Amin Azmoodeh, Ali Dehghantanha, David Ellis Newton, Reza M. Parizi, Hadis Karimipour, "*A deep Recurrent*

Neural Recurrent Neural Network based approach for Internet of Things malware threat hunting", In Journal of Systems Architecture.

[9] "*A Large-Scale Analysis of the Security of Embedded Firmwares*", Andrei Costin, Jonas Zaddach, Aurélien Francillon, and Davide Balzaro, *Eurecom* https://www.usenix.org/conference /usenixsecurity14/technical-sessions/presentation/ costin

[10] "Multidimensional binary search trees used for associative searching", Bentley, J.L., Communications of the ACM (1975)

[11] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario. Automated classification and analysis of internet malware. In *Proceedings of Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 178–197, Queensland, Australia, 2007. Springer

[12] José A. Sáez, Mikel Galar, Julián Luengo, Francisco Herrera, *Tackling the Problem of Classification with Noisy Data using Multiple Classifier Systems: Analysis of the Performance and Robustness.* Information Sciences, 247 (2013) 1-20.

[13] T. Gonzalez. "*Clustering to minimize the maximum intercluster distance*". Theoretical Computer Science 38, pages 293–306, 1985.

[14] Nghi Phu Tran and Quoc Dung Ngo and Dang Kien Hoang and Ngoc Binh Nguyen and Dai Tho Nguyen (2018) "*Phát hiện mã độc trên các thiết bị IoT dựa trên lời gọi Syscall và phân loại một lớp SVM."* In: Hội thảo lần thứ III Một số vấn đề chọn lọc về an toàn an ninh thông tin, December 6-7, 2018, Da Nang.

[15] Bentley, Jon Louis (1975), *A survey of techniques for fixed-radius near neighbor searching (PDF),* Technical Report SLAC-186 and STAN-CS-75 513, Stanford Linear Accelerator Center.