TRƯỜNG ĐẠI HỌC SƯ PHẠM TP HỒ CHÍ MINH
TẠP CHÍ KHOA HỌC
KHOA HỌC TỰ NHIÊN VÀ CÔNG NGHỆ
ISSN: 1859-3100 Tập 15, Số 12 (2018): 82-93

HO CHI MINH CITY UNIVERSITY OF EDUCATION
JOURNAL OF SCIENCE
NATURAL SCIENCES AND TECHNOLOGY
Vol. 15, No. 12 (2018): 82-93

Email: tapchikhoahoc@hcmue.edu.vn; Website: http://tckh.hcmue.edu.vn

PHƯƠNG PHÁP PHÁT HIỆN VIRUS MÁY TÍNH DỰA TRÊN HỆ MIỄN DỊCH NHÂN TẠO KẾT HỢP THÔNG TIN TỪ CẤU TRÚC PE CỦA TẬP TIN TRÊN HỆ ĐIỀU HÀNH WINDOWS

Nguyễn Tấn Toàn[1*], Vũ Thanh Nguyên[1], Trịnh Quốc Sơn[1], Lê Đình Tuấn[2]

[1] Trường Đại học Công nghệ Thông tin – ĐHQG TPHCM

[2] Trường Đại học Kinh tế Công nghiệp Long An

Ngày nhận bài: 28-8-2018; ngày nhận bài sửa: 24-9-2018; ngày duyệt đăng: 21-12-2018

TÓM TẮT

Bài báo này nghiên cứu về một phương pháp phát hiện virus dựa trên giải thuật của hệ miễn dịch nhân tạo (AIS), kết hợp với thông tin được trích xuất từ cấu trúc Portable Executable (PE) của các tập tin trên hệ điều hành Windows, nhằm giúp giảm chi phí trích xuất đặc trưng từ việc dùng đặc trưng của cấu trúc PE và tăng thêm sự đa dạng của các bộ phát hiện thông qua giải thuật hệ miễn dịch nhân tạo. Phương pháp đã được thực nghiệm với các bộ dữ liệu và các bộ phân lớp khác nhau (SVM, Naïve Bayes và Decision Tree). Kết quả thực hiện cho thấy độ chính xác của phương pháp có thể đạt lần lượt 89,25%, 79,93% và 87,38% khi sử dụng SVM, Naïve Bayes và Decision Tree trong giai đoạn phân lớp.

***Từ khóa*:** AIS, cấu trúc PE, phát hiện virus máy tính.

ABSTRACT

Computer virus detection method based on artficial immune system with information from PE structure from files on Windows

This paper presents a computer virus detection based on algorithms of artificial immune system (AIS) with information extracted from the Portable Executable (PE) structure of Windows PE files to reducing the cost of feature extraction via using features from the PE structure and increasing the variety of detector set by AIS. The proposal method is evaluated with multiple data sets and different classification methods (including SVM, Naïve Bayes and Decision Tree). The Accuracy of the proposal methods can reach 89.25%, 79.93% and 87.38% when using SVM, Naïve Bayes and Decision Tree in classification respectively.

Keywords: AIS, PE structure, computer virus detection.

1. Mở đầu

Ngày nay, virus máy tính thật sự là mối nguy hiểm và gây ra nhiều thiệt hại. Không những thế, số lượng của chúng lại tăng cực kì nhanh. Do đó, để giảm thiểu thiệt hại từ virus, nhiều nhà khoa học công nghệ thông tin đã và đang cố gắng nghiên cứu các phương pháp khác nhau để phát hiện virus máy tính.

[*] Email: *toannt@uit.edu.vn*

Trong phát hiện virus máy tính, hai phương pháp phát hiện virus kinh điển nhất là phương pháp dựa trên chữ kí và phương pháp dựa trên hành vi. Nhưng so với thời điểm hiện tại, hai phương pháp này không đủ tốt để giải quyết vấn đề của virus. Phương pháp dựa trên chữ kí cơ bản có nhược điểm là không thể nhận dạng được các virus chưa biết (mới hoặc là biến thể của virus trước đó). Trong khi đó, phương pháp dựa trên hành vi mặc dù có thể phát hiện được các virus chưa biết dựa trên chuỗi hành vi của tập tin nhưng chi phí để phân tích của phương pháp này rất tốn kém.

Do đó, gần đây, để tìm ra các phương pháp tốt hơn, nhiều phương pháp mới dựa trên khai thác dữ liệu, máy học, thống kê, hệ miễn dịch nhận tạo đã được các nhiều khoa học quan tâm. Đi theo xu hướng đó, bài báo này cũng sẽ tiếp cận theo hướng phát hiện virus mới dựa trên các giải thuật của hệ miễn dịch nhân tạo kết hợp với thông tin được trích xuất từ cấu trúc PE của tập tin trên hệ điều hành Windows, hi vọng sẽ đóng góp về nghiên cứu thử nghiệm một cách tiếp cận mới với việc kết hợp giá trị của dữ liệu PE trong phát hiện virus và khả năng xây dựng, đa dạng hóa các bộ phát hiện (detector) của AIS khi lượng dữ liệu huấn luyện còn hạn chế so với lượng dữ liệu thực tế trong phát hiện virus máy tính trên hệ điều hành Windows.

2. Các công trình liên quan

Như đã đề cập, hiện tại có nhiều phương pháp mới dựa trên khai thác dữ liệu, máy học, hệ miễn dịch nhân tạo đã được nghiên cứu [1], [2]. Một số ví dụ như sau:

R.Chao và cộng sự [3] đã xây dựng một hệ thống phát hiện virus mà trong hệ thống đó các chuỗi nhị phân của tập tin virus và tập tin sạch sẽ được trích xuất. Sau đó, các chuỗi nhị phân này trải qua quá trình chọn lọc âm tính (NSA), CLONALG (giải thuật nhân bản), và máy học (sử dụng SVM, KNN, RBF networks).

Bài báo [4], đã sử dụng hai giải thuật của hệ miễn dịch nhân tạo gồm NSA và mạng miễn dịch nhân tạo (artificial immune network – aiNet) trên đặc trưng dạng chuỗi nhị phân 32 bit được trích xuất từ các tập tin để xây dựng nên hệ thống phát hiện virus máy tính và kết quả kết quả bước đầu khá tốt.

WU Bin và cộng sự [5], đã xây đựng mô hình phát hiện malware trên smartphone. Các tập tins trong tập luận luyện được chuyển thành các vector đặc trưng. Mỗi vetor đặc trưng có 6 thuộc tính tĩnh (trích xuất mà không cần thực thi tập tin) và 7 thuộc tính động (trích xuất thông tin khi thực thi tập tin). Sau đó các vector đặc trưng trải qua các giai đoạn gồm chọn lọc âm tính (Negative Selection Algorithm – NSA), nhân bản và đột biến để tạo nên tập các bộ phát hiện bằng chọn lọc nhân bản (CLONALG). Sau đó, các bộ phát hiện nào nhận dạng đủ số lượng kháng nguyên trong quá trình hoạt động sẽ giữ lại. Bước cuối cùng của hệ thống là sử dụng phương pháp phân tích bằng phương pháp trọng số và phương pháp dựa trên k-means. Bài viết của tác giả này công bố đã đạt được tỉ lệ phát hiện lên đến 80%.

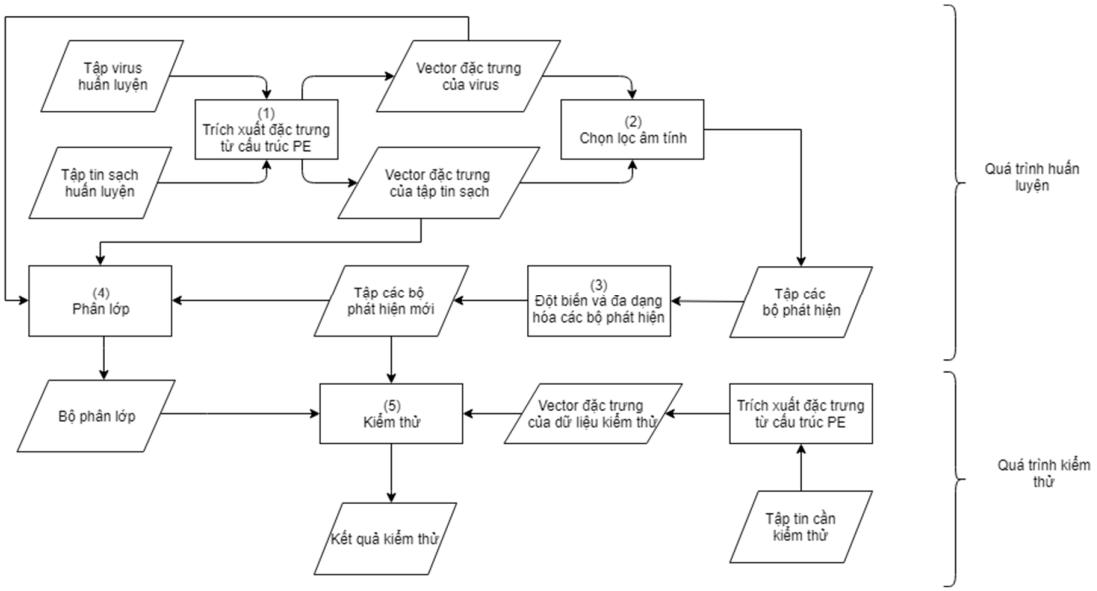
Bên cạnh đó, gần đây, thông tin trích xuất từ cấu trúc PE của các tập tin PE (PE header, DLL...) đã được sử dụng cho việc phát hiện malware [6]. Ví dụ, Baldangombo và cộng sự đã trích xuất và xây dựng các vector đặc trưng từ các cấu trúc PE của các tập tin [6]. Sau đó, các vector đặc trưng này trải qua quá trình phân lớp (SVM, J48, và Naïve Bayes). Kết quả của bài báo này đã công bố một tỉ lệ phát hiện đến 99,6%. Một bài báo khác [7] cũng sử dụng PE header và DLLs để tạo nên các vector đặc trưng. Sau đó, họ đã chạy các vector đặc trưng này trên các giải thuật của khai thác dữ liệu. Tỉ lệ phát hiện được công bố ở bài viết đó là hơn 99%.

Ta thấy, nhiều giải pháp sử dụng hệ miễn dịch nhân tạo cho phát hiện virus máy tính với đặc trưng là các chuỗi nhị phân phân (16 bit, 32 bit hoặc 64 bit) được trích xuất từng bit từ tập tin đầu vào như bài báo [3], [4]. Các hướng tiếp cận này đã cho kết quả khả quan, có đa dạng hóa các bộ phát hiện thông qua các giải thuật của hệ miễn dịch nhân tạo để tăng khả năng nhận dạng vì dữ liệu huấn luyện thường ít hơn so với dữ liệu thực tế rất nhiều nhưng việc sử dụng đặc trưng là chuỗi nhị phân từ các tập tin thường sẽ có số lượng cực kì lớn vì từ mỗi tập tin có thể trích ra rất nhiều chuỗi bit và điều đó có thể dẫn đến bùng nổ dữ liệu làm chi phí thực hiện cao. Bên cạnh đó, trong vài năm gần đây, các đặc trưng trích xuất từ cấu trúc PE ngày càng được thu hút sự chú ý của các nhà khoa học và bước đầu có kết quả khả quan như bài báo [6], [7]. Hầu hết các phương pháp này là áp dụng trực tiếp các giải thuật máy học lên trên các đặc trưng nên chi phí thấp, nhưng việc không sử dụng các phương pháp xử lí để tăng tính đa dạng và linh động trên các đặc trưng thì nếu dữ liệu huấn luyện không đủ lớn so với thực tế có thể sẽ làm hạn chế phần nào khả năng dự đoán các loại malware chưa từng gặp thực tế vì lượng malware trong thực tế là rất lớn và phát triển rất nhanh so với lượng được dùng huấn luyện và kiểm thử trong nghiên cứu.

Do đó, trong bài báo này, chúng tôi sẽ sử dụng thông tin trích xuất từ cấu trúc PE làm đặc trưng đầu vào. Mỗi tập tin sẽ được đại diện bởi một bộ đặc trưng duy nhất nhằm giảm nguy cơ bùng nổ lượng dữ liệu so với phương pháp trích xuất từng chuỗi nhị phân một từ tập tin. Đồng thời, sẽ sử dụng các giải thuật của hệ miễn dịch nhân tạo lên các đặc trưng từ PE để tạo ra các bộ phát hiện và đa dạng hóa chúng đó nhằm mục đích tăng sự đa dạng đó sẽ mở rộng khả năng nhận biết các loại virus mới trong thực tế khi mà dữ liệu huấn luyện, kiểm thử trong nghiên cứu luôn bị giới hạn so với sự phát triển chóng mặt của virus.

3. Phương pháp tiếp cận của bài báo

Phương pháp được đề xuất sẽ có giai đoạn chính: trích xuất đặc trưng, chọn lọc âm tính (NSA), đột biến và đa dạng hóa các bộ phát hiện, phân lớp và kiểm thử. Tổng quan về các bước thực hiện của bài báo được thể hiện ở Hình 1.



Hình 1. *Sơ đồ tổng quan các giai đoạn của phương pháp*

3.1. Trích xuất đặc trưng từ cấu trúc PE

Bước này sẽ xây dựng các vector đặc trưng cho các tập tin của bộ dữ liệu huấn luyện (bao gồm virus và tập tin sạch). Các vector đặc trưng của các tập tin của bộ dữ liệu huấn luyện được xây dựng dựa trên thông tin từ cấu trúc PE của tập tin PE (PE headers, các Dlls...). Trong bài báo, mỗi vector đặc trưng được chia làm hai phần: phần thứ nhất chứa các đặc trưng dạng số thực trích xuất từ các cấu trúc PE, phần thứ hai chứa các đặc trưng dạng nhị phân biểu diễn cho sự hiện diện hay không của DLL của tập tin đang phân tích. Danh sách các DLL được sử dụng trong bài báo là dựa trên kết quả trong bài báo [6]. Danh sách các đặc trưng được mô tả cụ thể ở Bảng 1 và Bảng 2 [6] - [8].

Bảng 1. *Danh sách các đặc trưng dạng số thực được trích xuất từ cấu trúc PE*

STT	Các đặc trưng
1	Tổng kích thước của phần dữ liệu khởi tạo của các vùng
2	Mã xác định đặc điểm DLL
3	Địa chỉ ảo tương đối của danh mục các chứng chỉ
4	Địa chỉ ảo tương đối của danh mục cấu hình tải dữ liệu của tập tin
5	Số lượng kí hiệu trong bảng kí hiệu COFF
6	Thông tin về phiên bản của tập tin
7	Mã kiểm tra lỗi của tập tin
8	Địa chỉ ảo tương đối của danh mục debug
9	Địa chỉ cơ sở để tải lên toàn bộ tập tin
10	Tổng kích thước của vùng tái định vị
11	Chứa giá trị xác định đặc điểm của tập tin
12	Tổng kích thước của vùng chứa thông tin tài nguyên

13	Địa chỉ ảo tương đối của nơi bắt đầu vùng mã nguồn của tập tin
14	Tổng kích thước của vùng dữ liệu
15	Kích thước bộ nhớ ảo dữ trữ cho heap
16	Số lượng các vùng trong tập tin
17	Địa chỉ ảo tương đối của danh mục trích xuất
18	Địa chỉ ảo tương đối của danh mục gọi các ràng buộc
19	Địa chỉ ảo tương đối của danh mục tái định vị địa chỉ cơ sở của tập tin

Bảng 2. *Danh sách các DLL sử dụng*

STT	DLL	STT	DLL
1	MSVFW32.dll	18	urlmon.dll
2	MSACM32.dll	19	version.dll
3	AVIFIL32.dll	20	crtdll.dll
4	MSASN1.dll	21	comdlg32.dll
5	kernel32.dll	22	winnm.dll
6	advapi32.dll	23	rpcrt4.dll
7	gdi32.dll	24	psapi.dll
8	wininet.dll	25	msvcr100.dll
9	comctl32.dll	26	hal.dll
10	shell32.dll	27	mpr.dll
11	wsock32.dll	28	netapi32.dll
12	oleaut32.dll	29	avicap32.dll
13	msvbvm50.dll	30	rasapi32.dll
14	ole32.dll	31	cygwin1.dll
15	shlwapi.dll	32	mscoree.dll
16	ws2_32.dll	33	imagehlp.dll
17	ntdll.dll		

Sau khi được trích xuất, các đặc trưng dạng số thực (ở Bảng 1) sẽ được chuẩn hóa bằng phương pháp min-max tương tự như bài báo [5]. Trong khi đó, mỗi đặc trưng nhị phân (hay là đặc trưng DLL) (xem ở Bảng 2) sẽ có giá trị 0 hoặc 1. Giá trị 1 biểu thị cho việc DLL đó được gọi vào tập tin đang xét. Giá trị 0 biểu thị là DLL đó không được gọi vào tập tin đang trích xuất.

Cuối giai đoạn này, hệ thống sẽ thu được hai tập vector đặc trưng bao gồm tập $V = \{v_1, v_2, \ldots, v_n\}$ và $B = \{b_1, b_2, \ldots, b_m\}$. Trong đó, V là tập vector đặc trưng của các tập tin virus trong tập huấn luyện. B là tập vector đặc trưng của tập tin sạch trong tập huấn luyện.

3.2. Giải thuật chọn lọc âm tính

Mục đích của giai đoạn này là để tạo tập các bộ phát hiện – đây là cơ sở xây dựng bộ phân lớp để dự đoán các tập tin. Chi tiết của giải thuật chọn lọc âm tính (NSA) này được mô tả trong

Giải thuật ***1***. Giải thuật NSA được sử dụng trong bài báo này là dựa trên giải thuật NSA trong bài báo [5], [9] với một vài sự thay đổi ở công thức tính khoảng cách để phù hợp với dữ liệu.

Mỗi vector đặc trưng ngoài có 2 phần như mô tả ở bước trích xuất thông tin, trong bước này mỗi vector đặc trưng sẽ có thêm một thông tin về bán kính biểu diễn cho phạm vi ảnh hưởng của vector đặc trưng đó. Như vậy, lúc này mỗi vector đặc trưng sẽ được cấu trúc dạng $\langle FP, DP, R \rangle$. Trong đó, FP, DP, R lần lượt biểu diễn cho thành phần đặc trưng dạng số, đặc trưng nhị phân cho DLL, và bán kính.

Trong bước này, giải thuật sẽ sử dụng 2 tham số R_{self} và $R_{nonself}$. R_{self} và $R_{nonself}$ lần lượt là 2 giá trị khởi tạo cho phần R của vector đặc trưng cho virus và vector đặc trưng cho tập tin sạch. Thêm vào đó, giải thuật sẽ sử dụng một khoảng cách giữa 2 phần tử e_1 và e_2 (e_1 và e_2 có thể là virus, bộ phát hiện hoặc vector của tập tin sạch). Khoảng cách của e_1 và e_2 viết tắt bằng kí hiệu Dis_{e_1,e_2} và được tính toán bởi công thức (3.1):

$$Dis_{e_1,e_2} = ED_{FP_{e_1},FP_{e_2}} + HD_{DP_{e_1},DP_{e_2}} \tag{3.1}$$

Trong công thức (3.1), $ED_{FP_{e_1},FP_{e_2}}$, một khoảng cách euclidean giữa FP của phần tử e_1 và FP của phần tử e_2, được tính theo công thức (3.2). $HD_{DP_{e_1},DP_{e_2}}$, là khoảng cách hamming giữa DP của e_1 và DP của e_2, được tính theo công thức (3.3), t và tt lần lượt là số đặc trưng có trong FP và DP.

$$ED_{FP_{e_1},FP_{e_2}} = \sqrt{\sum_{i=1}^{t}\left(FP_{e_1,i} - FP_{e_2,i}\right)^2} \tag{3.2}$$

$$\begin{aligned} HD_{DP_{e_1},DP_{e_2}} &= \frac{\sum_{i=1}^{tt}\left(HDU_{DP_{e_1,i},DP_{e_2,i}}\right)}{tt} \\ HDU_{DP_{e_1,i},DP_{e_2,i}} &= \begin{cases} 0, & DP_{e_1,i} \neq DP_{e_2,i} \\ 1, & DP_{e_1,i} = DP_{e_2,i} \end{cases} \end{aligned} \tag{3.3}$$

Giải thuật 1. *Giải thuật chọn lọc âm tính – NSA*

Đầu vào:

Tập vector đặc trưng của virus $V = \{v_1, v_2, \ldots, v_n\}$

Tập vector đặc trưng của tập tin sạch $B = \{b_1, b_2, \ldots, b_m\}$

Đầu ra:

Tập các bộ phát hiện $D = \{d_1, d_2, \ldots, d_k\}$

Begin

For $i \leftarrow 1$ *to* n *do*

$detector \leftarrow v_i$

For $j \leftarrow 1$ *to* m *do*

If $Dis_{detector, a_i} - R_{self} \leq R_{detector}$ *then*

$R_{detector} = Dis_{detector, a_i} - R_{self}$

If $R_{detector} > R_{self}$ *then*

Thêm $detector$ *vào* D

End

3.3. Đột biến và đa dạng hóa các bộ phát hiện

Trong bước này, tập các bộ phát hiện đã thu được ở giải đoạn NSA sẽ trải qua một giải thuật để tăng sự đa dạng và độ phủ của chúng. Giải thuật được sử dụng ở bước này là dựa trên CLONALG trong bài báo [5] với một vài sự thay đổi. Chi tiết của giải thuật được thể hiện ở Giải thuật 2.

Giải thuật 2. *Giải thuật CLONALG cho đột biến và đa dạng hóa tập các bộ phát hiện*

Đầu vào:

Tập vector đặc trưng của virus $V = \{v_1, v_2, \ldots, v_n\}$

Tập vector đặc trưng của tập tin sạch $B = \{b_1, b_2, \ldots, b_m\}$

Tập các bộ phát hiện $D = \{d_1, d_2, \ldots, d_k\}$

Đầu ra:

Tập các bộ phát hiện đã trải qua đột biến và đa dạng hóa $D = \{d_1, d_2, \ldots, d_q\}$

Begin

Tính ái lực của các bộ phát hiện

Sắp xếp tập D *theo chiều giảm dần ái lực*

Chọn N *bộ phát hiện có ái lực cao nhất*

Mỗi bộ phát hiện được chọn tạo M *bảo sao và đột biến bản sao*

Thêm các bản sao được tạo ra vào tập D

End

Trong Giải thuật 2, ái lực của d_1 sẽ được kí hiệu là Aff_{d_i}. Ái lực này được tính bằng công thức (3.4) dựa trên công thức sử dụng trong bài báo [5] với một số thay đổi để phù hợp với đặc trưng bài toán.

$$Aff_{d_i} = Vol_{d_i} - \delta \cdot Olp_{d_i, D} + HDA_{d_i, D} \quad (3.4)$$

Trong công thức (3.4), Vol_{d_i} được tính bằng công thức (3.5):

$$Vol_{d_i} = \frac{\lceil (t/2+1)}{\pi^{t/2}} R_{d_i}^t$$

$$\lceil (t/2+1) = \begin{cases} \frac{t}{2}!, \ t \text{ la so chan} \\ \frac{1 \times 2 \times \ldots \times \left(2 \times \frac{t+1}{2} - 1\right)}{2 \times \frac{t+1}{2}} \sqrt{\pi}, \ t \text{ la so le} \end{cases} \quad (3.5)$$

$Olp_{d_i,D}$ là độ phủ trùng lắp giữa các bộ phát hiện và δ là hệ số trừng phạt cho độ trùng lắp. $Olp_{d_i,D}$ được tính theo công thức (3.6)

$$Olp_{d_i,D} = \sum_{i=1}^{k} Olp_{d_i,d_j}$$

$$Olp_{d_i,d_j} = \begin{cases} 0, \ Dis_{d_i,d_j} \geq R_{d_i} + R_{d_j} \\ \left(\exp\left(\frac{R_{d_i} + R_{d_j} - Dis_{d_i,d_j}}{R_{d_i} + R_{d_j}} \right) - 1 \right)^n, \ Dis_{d_i,d_j} < R_{d_i} + R_{d_j} \end{cases} \quad (3.6)$$

$HDA_{d_i,D}$ là khoảng cách hamming trung bình giữa bộ phát hiện d với các bộ phát hiện trong tập D và được tính theo công thức (3.7). Trong đó, q là số lượng bộ phát hiện trong tập D

$$HDA_{d,D} = \frac{\sum_{i=1}^{q} \left(HD_{FP_d, FP_{d_i}} \right)}{q} \quad (3.7)$$

Trong bước tạo bản sao và đột biến của Giải thuật 2, N là số lượng bộ phát hiện được chọn để trải qua quá trình nhân bản. Mỗi bộ phát hiện được chọn sẽ tạo ra M bản sao. Mỗi bản sao của bộ phát hiện được chọn sẽ được tạo ra bằng sự đột biến FP và DF của bộ phát hiện được chọn với 2 cơ chế khác nhau.

FP của một bản sao được đột biến bằng toán tử đột biến cauchy theo các công thức (3.8) [5], [10].

$$\begin{aligned} FP'_{d_i,j} &= FP_{d_i,j} + \eta_{i,j} \times \delta_j, \text{ vôi } j = 1,2,3,...,t \\ \eta'_{i,j} &= \eta_{i,j} \times \exp\left(\tau_a \times N(0,1) + \tau_b \times N_j(0,1) \right), \text{vôi } j = 1,2,3,...,t \\ \tau_a &= \left(\sqrt{2 \times \sqrt{t}} \right)^{-1} \\ \tau_b &= \left(\sqrt{2 \times t} \right)^{-1} \end{aligned} \quad (3.8)$$

Trong các công thức (3.8), FP'_{d_i} là FP của một bản sao của detecotr d_i. η_i, η'_i lần lượt là các tham số của bản sao và bộ phát hiện. δ_j là một biến ngẫu nhiên theo phân phối

cauchy chuẩn. $FP'_{d_i,j}$, $FP_{d_i,j}$, $\eta_{i,j}$, $\eta'_{i,j}$ lần lượt là thành phần thứ j^{th} của FP'_{d_i}, FP_{d_i}, η_i, η'_i. $N(0,1)$ là một số ngẫu nhiên theo phân phối chuẩn với mean 0 và độ lệch chuẩn là 1. $N_j(0,1)$ là một số ngẫu nhiên cho phần tử thứ j^{th} của FP'_{d_i} [5], [10].

Trong khi đó, DP của bản sao sẽ đột biến bằng đột biến điểm ngẫu nhiên. Z vị trí của DP được chọn ngẫu nhiên và thay đổi giá trị từ 0 thành 1 và ngược lại. Trong đó, Z là một tham số được sử dụng để xác định số lượng đặc trưng của DP sẽ được đột biến.

3.4. Giai đoạn phân lớp

Các vector đặc trưng của tập dữ liệu huấn luyện sẽ được tính toán độ nguy hiểm của nó với các bộ phát hiện D đã thu được.

Mỗi vector nguy hiểm sẽ có dạng $\langle EDA, HDA \rangle$. Chúng được tính toán theo công thức (3.9) và công thức (3.10) mà trong đó q là số lượng các bộ phát hiện có trong D.

$$EDA_{d,D} = \frac{\sum_{i=1}^{q}\left(ED_{FP_d,FP_{d_i}}\right)}{q} \tag{3.9}$$

$$HDA_{d,D} = \frac{\sum_{i=1}^{q}\left(HD_{FP_d,FP_{d_i}}\right)}{q} \tag{3.10}$$

Sau khi thu được các vector độ nguy hiểm cần thiết, các vector đó được chuẩn hóa bằng phương pháp min-max tương tự phương pháp được sử dụng ở bài báo [5] và tiếp đến là trải qua quá trình phân lớp (SVM, Naive Bayes và Decision Tree). Kết quả của quá trình này là một model được sử dụng cho phát hiện virus về sau.

3.5. Kiểm thử

Trong bước kiểm thử, các tập tin kiểm thử được chuyển thành các vector đặc trưng bằng kĩ thuật trích xuất đặc trưng từ cấu trúc PE như đã nếu trong mục 0. Tiếp theo, các vector độ nguy hiểm của từng vector đặc trưng của tập tin kiểm thử được tính toán và chuẩn hóa theo như giai đoạn phân lớp 0. Cuối cùng, các vector độ nguy hiểm được kiểm tra bằng bộ phân lớp thu được ở bước phân lớp 0 và đánh giá kết quả.

4. Thực nghiệm

Trong thí nghiệm của bài báo, để trích xuất thông tin từ cấu trúc PE, phần mềm Microsoft dumpbin được sử dụng. Có tổng số 5 tập dữ liệu được sử dụng. Tỉ lệ số lượng tập tin trong bộ dữ liệu huấn luyện và kiểm thử là 7:3. Chi tiết của bộ dữ liệu được thể hiện trong Bảng 3.

Bảng 3. *Các tập dữ liệu trong bộ dữ liệu thử nghiệm*

		Số lượng tập tin huấn luyện		**Số lượng tập tin kiểm thử**	
Stt	**Tập dữ liệu**	**Virus**	**Tập tin sạch**	**Virus**	**Tập tin sạch**
1	Tập 1	200	100	86	43
2	Tập 2	400	200	171	86
3	Tập 3	600	300	257	129

4	Tập 4	800	400	343	171
5	Tập 5	1000	500	429	214

Sau khi chạy tất cả các bộ dữ liêu, chúng tôi đã thu được các kết quả được thể hiện trong Bảng 4 và Biểu đồ 1.

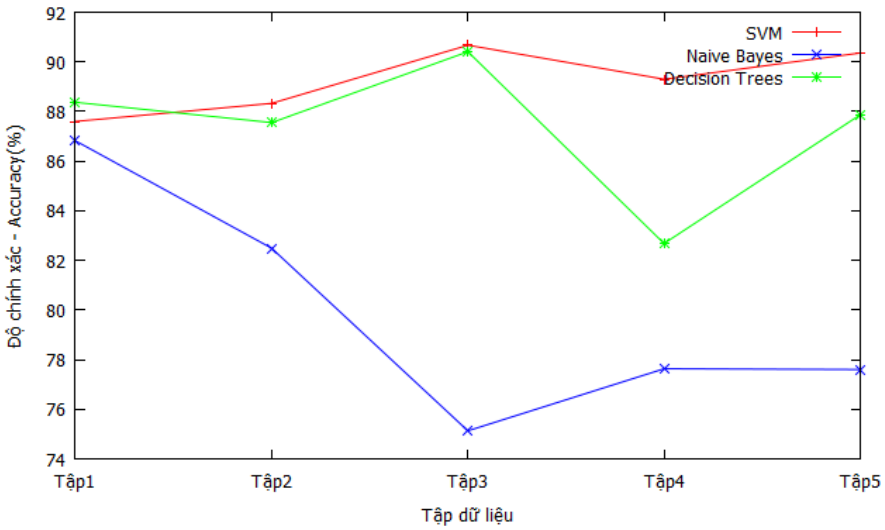
Biểu đồ 1Trong đó độ chính xác được tính bằng công thức (4.1):

$$\text{Ño chính xac } (Accuracy) = \frac{\text{So löông tap tin dö ñoan ñung}}{\text{So löông tap tin kiem thö}} \quad (4.1)$$

Bảng 4. *Kết quả độ phát hiện chính xác*

		Độ chính xác (Accuracy - %)		
STT	**Tập dữ liệu**	**SVM**	**Naïve Bayes**	**Decision Tree**
1	Tập 1	87,6	86,82	88,37
2	Tập 2	88,33	82,49	87,55
3	Tập 3	90,67	75,13	90,41
4	Tập 4	89,3	77,63	82,68
5	Tập 5	90,36	77,6	87,87
Trung bình		89,25	79,93	87,38

Biểu đồ 1. *Biểu đồ kết quả độ phát hiện chính xác khi thực nghiệm*



Kết quả trong bảng Bảng 4 và Biểu đồ 1 cho thấy, độ chính xác của hệ thống có thể đạt được lần lượt 89,25%, 79,93%, và 87,38% khi sử dụng SVM, Naive Bayes, và Decision Tree. Tỉ lệ phát hiện cao nhất đạt được của SVM trong thử nghiệm là 90,67% và thấp nhất của SVM là 87,6%. Trong khi đó cao nhất, thấp nhất của phương pháp sử dụng Decision Tree và Naive Bayes lần lượt là 86,82%, 75,13% và 90,41%, 82,68%. Dễ dàng thấy rằng phương pháp được đề xuất có thể đạt được độ chính xác cao và điều đó phản ánh tiềm năng của phướng pháp mà bài báo tiếp cận. Ta thấy, bộ phân lớp có tỉ lệ phát hiện cao

nhất là SVM nên có thể là phương pháp phân lớp tiềm năng nhất cho phương pháp tiếp cận của bài báo.

5. Kết luận và hướng phát triển

Bài báo đã tiếp cận việc phát hiện virus máy tính bằng phương pháp sử dụng AIS kết hợp với thông tin được trích xuất từ cấu trúc PE. Phương pháp tiếp cận của bài báo có thể dự đoán các tập tin chưa biết trước đó với một hiệu suất khá tốt. Các thử nghiệm đã cho thấy rằng cách tiếp cận của bài báo có thể đạt lần lượt độ chính xác trung bình là 89,25%, 79,93%, và 87,38% khi sử dụng SVM, Naive Bayes, và Decision Tree. Về lí thuyết, ta thấy thông tin số thực từ cấu trúc PE có sự khác nhau trong miền giá trị giữa tập tin sạch và tập tin virus. Bên cạnh đó, các DLL quan trọng chứa các hàm liên quan đế đọc, viết, copying dữ liệu... của hệ thống và đây là các loại hành vi mà virus thường sử dụng nên các DLL chứa các hành vi này thường xuyên được gọi trong các virus máy tính. Do đó, mà việc kết hợp thông tin dạng số từ thông tin của các PE header và thông tin của DLL có thể sẽ là đặc trưng tốt trong phân biệt virus và tập tin sạch. Trong giai đoạn sử dụng AIS, NSA sẽ loại bỏ các vector đặc trưng không tốt khi nó quá gần giống với tập tin sạch nhằm để giảm thiểu sai sót trong bước dự đoán và giữ lại các vector đặc trưng tốt cho quá trình tạo các bộ phát hiện. Để làm cho hệ thống có thể dự đoán được virus chưa biết tốt hơn, một biến thể của CLONALG trong bài báo [5] được sử dụng để nhân bản các bộ phát hiện với một vài đột biến nhằm làm cho các bộ phát hiện trở nên đa dạng hơn. Tuy nhiên, việc tìm một bộ phân biệt virus và tập tin sạch một cách thủ công là tương đối khó nên phương pháp phân lớp được sử dụng. Để phân lớp, các vector nguy hiểm được tính toán, chuẩn hóa và trải qua quá trình phân lớp nhằm tạo bộ phân lớp cho việc dự đoán virus. Đó là các nguyên nhân lí giải cho việc phương pháp có thể đạt được kết quả khá tốt như mong đợi.

Trong tương lai, để làm cho phương pháp này trở nên tốt hơn, chúng tôi sẽ đánh giá phương pháp bằng các đặc trưng khác, các biến thể của giải thuật AIS khác nhau cùng với tập các dữ liệu đa dạng, phong phú hơn. Bên cạnh đó, ở thời điểm hiện tại, hệ thống chỉ có thể dự đoán được một tập tin là virus hay tập tin sạch nên các nỗ lực để dự đoán được kiểu virus sẽ được đầu tư nhằm làm cho hệ thống tốt hơn.

❖ ***Tuyên bố về quyền lợi:*** *Các tác giả xác nhận hoàn toàn không có xung đột về quyền lợi.*

❖ ***Lời cảm ơn:*** *Nghiên cứu này được tài trợ bởi Đại học Quốc gia Thành phố Hồ Chí Minh (VNUHCM) cho đề tài mã số C2018-26-06.*

TÀI LIỆU THAM KHẢO

[1] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," *Hum-Centric Comput. Inf. Sci.*, *8*(1), p. 3, Jan. 2018.

[2] V. Mehare and R. S. Thakur, "Data Mining Models for Anomaly Detection Using Artificial Immune System," in *Proceedings of International Conference on Recent Advancement on Computer and Communication*, Singapore, 2018, pp. 425-432.

[3] R. Chao and Y. Tan, "A Virus Detection System Based on Artificial Immune System," in *2009 International Conference on Computational Intelligence and Security*, 2009, *1*, pp. 6-10.

[4] V. T. Nguyen, T. T. Nguyen, K. T. Mai and T. D. Le, "A Combination of Negative Selection Algorithm and Artificial Immune Network for Virus Detection," in *Future Data and Security Engineering*, Cham, 2014, pp. 97-106.

[5] B. Wu, T. Lu, K. Zheng, D. Zhang and X. Lin, "Smartphone malware detection model based on artificial immune system," *China Commun.*, *11*(13), pp. 86-92, Supplement 2014.

[6] U. Baldangombo, N. Jambaljav and S.-J. Horng, "A Static Malware Detection System Using Data Mining Methods," *CoRR*, abs/1308.2831, 2013.

[7] M. Z. Shafiq, S. M. Tabish, F. Mirza, and M. Farooq, "PE-Miner: Mining Structural Information to Detect Malicious Executables in Realtime," in *Recent Advances in Intrusion Detection*, Berlin, Heidelberg, 2009, pp. 121-141.

[8] Y. Liao, "Pe-header-based malware study and detection," *Retrieved Univ. Ga. Httpwww Cs Uga Edu~ LiaoPEFinalReport Pdf*, 2012.

[9] Z. Ji and D. Dasgupta, "Real-Valued Negative Selection Algorithm with Variable-Sized Detectors," in *Genetic and Evolutionary Computation – GECCO 2004*, Berlin, Heidelberg, 2004, pp. 287-298.

[10] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, 2015, pp. 11-20.