(12) BẢN MÔ TẢ SÁNG CHẾ THUỘC BẰNG ĐỘC QUYỀN SÁNG CHẾ

(19) **Cộng hòa xã hội chủ nghĩa Việt Nam (VN)**
CỤC SỞ HỮU TRÍ TUỆ

(11) **1-0036538**

(51)[2020.01] **G06F 21/00**

(13) **B**

(21) 1-2021-00205

(22) 15/01/2021

(45) 25/08/2023 425

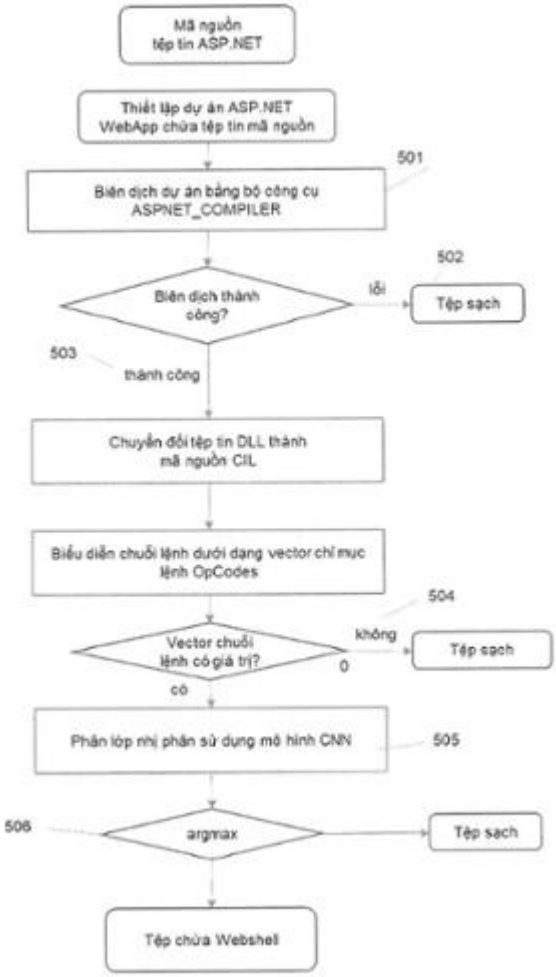
(43) 25/10/2021 403

(73) 1. Trường Đại học Công Nghệ - Đại học Quốc Gia Hà Nội (VN)
E3, 144 Xuân Thủy, phường Dịch Vọng Hậu, quận Cầu Giấy, thành phố Hà Nội
2. Nguyễn Ngọc Hoá (VN)
P311-E3, 144 Xuân Thuỷ, phường Dịch Vọng Hậu, quận Cầu Giấy, thành phố Hà Nội

(72) Nguyễn Ngọc Hoá (VN); Lê Việt Hà (VN).

(54) PHƯƠNG PHÁP PHÁT HIỆN ĐOẠN MÃ ĐỘC TRONG MÃ NGUỒN ỨNG DỤNG WEB SỬ DỤNG NGÔN NGỮ ASP.NET

(57) Sáng chế đề cập đến phương pháp phát hiện đoạn mã độc Webshell trong mã nguồn ứng dụng Web được viết bằng ngôn ngữ ASP.NET. Mô hình này bao gồm các bước: (i) sử dụng phương pháp đối sánh mẫu để xây dựng một bộ dữ liệu chứa đoạn mã độc Webshell, (ii) chuyển đổi tệp mã nguồn ứng dụng ASP.NET thành một chuỗi các mã lệnh thực thi CIL Opcodes và (iii) áp dụng phương pháp học sâu với mô hình mạng nơron tích hợp (CNN) dựa trên các bộ dữ liệu đã chuyển sang vectơ chuỗi mã lệnh để xác định tệp mã nguồn có bị nhúng đoạn mã độc hay không.


Mã nguồn
tệp tin ASP.NET
Thiết lập dự án ASP.NET
WebApp chứa tệp tin mã nguồn
501
Biên dịch dự án bằng bộ công cụ
ASPNET_COMPILER
502
Biên dịch thành
công?
lỗi
Tệp sạch
503
thành công
Chuyển đổi tệp tin DLL thành
mã nguồn CIL
Biểu diễn chuỗi lệnh dưới dạng vector chỉ mục
lệnh OpCodes
504
Vector chuỗi
lệnh có giá trị?
không
0
Tệp sạch
có
Phân lớp nhị phân sử dụng mô hình CNN
505
506
argmax
Tệp sạch
Tệp chứa Webshell

Lĩnh vực kỹ thuật được đề cập

Sáng chế đề cập đến phương pháp phát hiện đoạn mã độc trong mã nguồn. Cụ thể, sáng chế đề cập đến phương pháp để phát hiện đoạn mã độc Webshell trong mã nguồn ứng dụng ASP.NET. Phương pháp bao gồm các bước: (i) biểu diễn mã nguồn tệp tin ASP.NET dưới dạng vector mã lệnh CIL OpCode, (ii) làm sạch bộ dữ liệu huấn luyện mô hình dựa trên kỹ thuật đối sánh mẫu so khớp sử dụng bộ công cụ yara, và (iii) áp dụng phương pháp học sâu sử dụng mô hình mạng nơ ron tích chập CNN để huấn luyện mô hình và xác định xem mã nguồn ứng dụng Web có chứa Webshell hay không.

Tình trạng kỹ thuật của sáng chế

Ngày nay, với sự phổ biến của .NET Framework, các ứng dụng Web sử dụng ASP.NET, một nền tảng ứng dụng web (web application framework) được phát triển và cung cấp bởi Microsoft, cho phép những lập trình viên dễ dàng tạo ra những trang web động, ứng dụng web và dịch vụ web, cũng xuất hiện nhiều hơn. Lần đầu tiên được đưa ra thị trường vào tháng 2 năm 2002 cùng với phiên bản 1.0 của .NET Framework, là công nghệ nối tiếp của Microsoft's Active Server Pages(ASP). ASP.NET được biên dịch dưới dạng Common Language Runtime (CLR) cho phép những người lập trình viết mã ASP.NET với bất kỳ ngôn ngữ nào được hỗ trợ bởi .NET Framework. Giống như các loại ứng dụng Web khác, ASP.NET cũng phải đối mặt với một trong những hình thức tấn công phổ biến nhất hiện nay đó là chèn đoạn mã độc vào mã nguồn ứng dụng hay còn gọi là tấn công Webshell cho phép tin tặc tạo giao diện để có thể truy cập, thi hành các lệnh/thao tác điều khiển máy chủ Web từ xa, giống như sử dụng công cụ Shell tại máy chủ đó. Khi bị nhúng vào các máy chủ Web, các đoạn mã webshell trở thành mã độc và

là một trong những công cụ hữu hiệu để tin tặc sử dụng, tấn công hệ thống thông tin liên quan đến máy chủ Web. Các đoạn mã độc Webshell có thể bị nhúng một cách chủ động thông qua những hành vi bất hợp pháp của nhà phát triển ứng dụng Web, cũng có thể bị nhúng thụ động bởi tin tặc dựa vào việc khai thác lỗ hổng bảo mật máy chủ Web. Để đối phó với những hình thức mất an toàn thông tin đó, cần thiết phải có công cụ cho phép phân tích tĩnh toàn bộ mã nguồn ứng dụng Web để phát hiện và loại bỏ những đoạn mã độc Webshell.

Hiện nay, các công trình nghiên cứu và các sáng chế phục vụ phát hiện đoạn mã độc (Webshell) trong mã nguồn ứng dụng Web chủ yếu tập trung theo 2 hướng chính: phát hiện đoạn mã độc bằng các phương pháp so khớp mẫuvà phát hiện đoạn mã độc bằng ứng dụng những mô hình học máy, học sâu.

Tài liệu sáng chế CN109600382A, công bố năm 2019, đề xuất phương pháp phát hiện WebShell sử dụng mô hình Mô hình Markov ẩn (HMM) một mô hình học máy tiêu biểu thường được sử dụng để nhận dạng ngôn ngữ, xử lý ngôn ngữ tự nhiên, nhận dạng mẫu, v.v.. Phương pháp này được mô tả gồm các bước: lấy nhật kí truy cập ứng dụng web; huấn luyện mô hình HMM với tập dữ liệu các log truy cập bao gồm cả webshell; phát hiện webshell bằng mô hình HMM đã được huấn luyện. Cuối cùng, sáng chế sử dụng phương pháp học sâu Recursive_LSTM để phát hiện các WebShell.

Tài liệu sáng chế KR101068931B1, công bố năm 2015, cung cấp phương pháp phát hiện WebShell dựa trên kỹ thuật so khớp mẫu. Bằng cách phân tích mã nguồn của ứng dụng web hiện có trong máy chủ web để kiểm tra các đoạn mẫu mã độc thường được sử dụng trong webshell và phân tích nhật ký của máy chủ ứng dụng web. Bằng cách kiểm tra xem có bất kỳ dấu vết hoạt động nào của webshell, nó có thể phát hiện sự tồn tại của webshell tương đối chính xác và cung cấp một phương pháp mới cho quản trị viên để kiểm tra và cách ly từ xa chương trình web bị nghi ngờ là webshell theo kết quả phát hiện.

Tài liệu sáng chế CN104967616A năm 2015, nhóm tác giả đã đưa ra phương pháp phát hiện các tệp tin webshell trên máy chủ ứng dụng bằng cách đề xuất và thực hiện cách thức tính toán các giá trị tham chiếu, giá trị thời gian, giá trị thuộc tính của một tệp tin đồng thời cũng đánh trọng số cho 03 loại giá trị này, từ đó tính toán ra giá trị tổng hợp của một tệp tin để có thể xác định tệp tin này là sạch, nghi ngờ là webshell hoặc khẳng định là webshell. Phương pháp này có thể phát hiện tệp webshell trong máy chủ một cách nhanh chóng và chính xác.

Trong số các nghiên cứu, sáng chế hiện nay hầu hết chưa có nghiên cứu nào tập trung vào phương pháp phát hiện đoạn mã độc webshell trong mã nguồn ứng dụng Web sử dụng ngôn ngữ ASP.NET. Ngoài ra, việc làm sạch các bộ dữ liệu phục vụ quá trình huấn luyện mô hình cũng chưa được chú trọng nghiên cứu trong các phương pháp đó. Chính vì thế, trong sáng chế này chúng tôi sẽ đề xuất phương pháp phát hiện đoạn mã độc Webshell dựa trên phương pháp học sâu nhưng áp dụng cả phương pháp đối sánh mẫu trong quá trình xây dựng bộ dữ liệu huấn luyện để nâng cao hiệu suất phát hiện đối với các ứng dụng Web xây dựng bằng ngôn ngữ ASP.NET.

Bản chất kỹ thuật của sáng chế

Ở khía cạnh quản trị hệ thống ứng dụng web, một trong những nhiệm vụ thường xuyên phải thực hiện để đảm bảo an toàn thông tin là rò quét, phát hiện và loại bỏ các nguy cơ rủi ro từ mã độc (chẳng hạn Webshell) trong mã nguồn ứng dụng. Tại sáng chế này, chúng tôi đề xuất phương pháp phát hiện đoạn mã độc Webshell trong mã nguồn ứng dụng Web sử dụng công nghệ ASP.NET bằng cách sử dụng kỹ thuật biểu diễn mã nguồn ứng dụng ASP.NET dưới dạng vector chuỗi mã thực thi và mô hình mạng nơron tích chập (CNN) để phát hiện các đoạn mã độc (WebShell). Phương pháp dựa trên những ý tưởng chính sau:

(a) thu thập tập dữ liệu các tệp tin mã nguồn ứng dụng ASP.NET bao gồm các tệp tin lành tính từ các hệ thống quản lý nội dung CMS nổi tiếng như Dotnetnuke, Umbraco, Kentico, … để tạo thành bộ dữ liệu lành tính(bộ dữ

liệu Benign) và thu thập các tệp mã nguồn chứa đoạn mã độc 101 để hình thành bộ dữ liệu các đoạn mã độc (bộ dữ liệu Webshell), các tệp tin này đã được thu thập bởi cộng đồng mạng (chẳng hạn như, cộng đồng chia sẻ mã nguồn Github (tennc/webshell, /ysrc/webshell-sample, ...));

(b) làm sạch tập dữ liệu tệp mã nguồn ASP.NET chứa đoạn mã độc bằng cách sử dụng phương pháp đối sánh102 (chẳng hạn như, theo bộ quy tắc Yara) để loại bỏ các tệp tin lành tính 103 bị lẫn trong bộ dữ liệu các đoạn mã độc với mục tiêu chỉ giữ lại các tệp chứa đoạn mã độc nhằm mục đích giảm nhiễu trong quá trình huấn luyện mô hình mạng nơron tích chập, hoặc để bổ sung vào bộ dữ liệu các đoạn mã độc hại 104 nếu các tệp tin được xác định là độc hại;

(c) chuyển hóa mã nguồn các tệp tin ASP.NET thành dạng không gian vectơ bằng cách chuyển hóa sang dạng mã lệnh thực thi (opcodes); Quá trình này được thực hiện theo 3 bước chính (*hình 2*). Đầu tiên, các tệp tin mã nguồn sẽ được thiết lập trong một dự án ASP.NET trống để thực hiện tiền biên dịch 201 thành các tệp tin thư viện liên kết động DLL (Dynamic Library Link) thông qua công cụ biên dịch ASPNET_COMPILER của Microsoft. Bước tiếp theo, các tệp tin DLL này sẽ được chuyển đổi thành chuỗi mã lệnh opcodes bằng công cụ MSIL Disassembler. Cuối cùng, ánh xạ chuỗi lệnh đó thành vector chuỗi chỉ mục tương ứng với mã lệnh opcodes;

(d) loại bỏ các không gian vectơ không phù hợp với độ dài quá ngắn hoặc các không gian vectơ đã tồn tại 206; trong đó việc loại bỏ các không gian vectơ không phù hợp với độ dài quá ngắn do các không gian vectơ này là quá nhỏ đối với một đoạn mã độc; trong đó việc loại bỏ các không gian vectơ đã tồn tại trong bộ dữ liệu để đảm bảo mỗi không gian vectơ trong bộ dữ liệu là duy nhất để giảm thiểu kích thước của tập dữ liệu và tăng tốc độ huấn luyện mô hình mạng nơron tích chập;

(e) huấn luyện mô hình mạng nơron tích chập (CNN) với bộ dữ liệu sạch và bộ dữ liệu các đoạn mã độc đã làm sạch; mô hình này được thiết lập với ba lớp

tích chập để học đặc trưng, sử dụng phương pháp hồi quy tuyến tính để phân lớp với thuật toán tối ưu Adam; trong đó:

- tầng đầu vào 401là tập dữ liệu các tệp tin sạch thuộc bộ dữ liệu sạch (Benign) và tệp tin mã độc thuộc bộ dữ liệu các đoạn mã độc (bộ dữ liệu Webshell) sau khi đã loại bỏ các không gian vectơ không phù hợp với độ dài quá ngắn hoặc các không gian vectơ đã tồn tại;

- tầng thứ thứ hai 402 là tầng tích chập được cấu thành từ ba lớp tích chập;tiếp đó, các lớp tích chập này sẽ được ghép nối lại thành một lớp tích chập hợp nhất theo cơ chế nối chuỗi (concatenate), mục đích cuối cùng của tầng thứ hai để thực hiện phát hiện ra những đặc trưng hiệu quả nhất được thể hiện bởi ma trận đặc trưng (feature map).

- tầng thứ ba 403 là lớp gộp hay còn gọi là lớp tổng hợp (pooling layer) nó làm giảm số tham số mà ta cần phải tính toán, từ đó giảm thời gian tính toán, tránh quá mức (overfitting);

- tầng thứ tư 404 là tầng phân lớp, tầng này có chức năng chuyển ma trận đặc trưng ở tầng trước thành vectơ chứa xác suất của các đối tượng cần được dự đoán;

(f) phân tích tệp mã nguồn ASP.NET với mô hình mạng nơron tích chập đã được huấn luyện để xác định có chứa đoạn mã độc (WebShell) hay không.

Theo một khía cạnh khác của sáng chế, trong bước chuyển hóa mã nguồn mỗi tệp tin ASP.NET thành dạng không gian vectơ, bộ công cụ biên dịch ASP.NET của Microsoft (ASP.NET compilation tool (aspnet_compiler.exe)) cho phép tiền biên dịch mã nguồn của ứng dụng ASP.NET thành các tệp tin dạngPE (Portable Executable) cụ thể là các tệp tin thư viện liên kết động DLL, trong trường hợp biên dịch không thành công 202 có nghĩa là mã nguồn tệp tin ASP.NET có lỗi, khi đó tệp tin này sẽ bị loại bỏ khỏi tập dữ liệu.Trong trường hợp biên dịch thành công 203 các tệp tin PE này sẽ được chuyển hóa thành mã opcode thông qua bộ công cụ MSIL Disassembler; mã lệnh thực thi khi đó sẽ được chuyển đổi thành dạng không gian vectơ 204 tương ứng.

Theo một khía cạnh khác của sáng chế, trong bước tiến hành huấn luyện mô hình mạng nơron tích chập, tầng thứ thứ hai 402 bao gồm 128 bộ lọc (filters) có kích thước lần lượt là 3, 4, 5 sử dụng hàm kích hoạt phi tuyến ReLU $f(x) = \max(0, x)$ trong đó x là giá trị đầu vào để tạo ra thông tin trừu tượng hơn (Abstract/higher-level) cho các lớp tiếp theo.

Theo một khía cạnh khác của sáng chế, trong bước tiến hành huấn luyện mô hình mạng nơron tích chập, tầng thứ ba 403 sử dụng lớp tổng hợp tối đa (global_max_pooling) với kích thước bằng với kích thước của dữ liệu đầu vào và hệ số sụt giảm (dropout) 0,6 nghĩa là trong quá trình huấn luyện mô hình, với mỗi lần thực hiện vòng lặp, ta ngẫu nhiên loại bỏ 60% số lượng nút trong lớp; mục đích chính của tầng này là giảm chiều của tầng trước đó, loại bỏ những đặc trưng không còn cần thiết, thay vào đó giữ lại các đặc trưng đã đủ để phân loại đối tượng.

Theo một khía cạnh khác của sáng chế, trong bước tiến hành huấn luyện mô hình mạng nơron tích chập, sử dụng hàm kích hoạt softmax, là một cách ràng buộc đầu ra của các mạng nơron phải có tổng bằng 1; qua đó, các giá trị đầu ra của hàm softmax có thể được coi như là một phân phối xác suất của các biến đầu ra hay nói cách khác hàm softmax sẽ chuyển đổi giá trị đầu ra của mạng nơron bằng cách chia cho tổng giá trị; hàm softmax được thể hiện trong công thức sau:

$$\text{softmax}(X)_{ij} = \frac{\exp(X_{ij})}{\sum_k \exp(X_{ik})}$$

hàm mất mát cross entropy, sử dụng để so sánh khoảng cách giữa các giá trị đầu ra của softmax và quá trình biến đổi từng giá trị thành các đặc trưng nhị phân (one-hot encoding), trong đó quá trình biến đổi từng giá trị thành các đặc trưng nhị phân (One-hot encoding) là quá trình biến đổi từng giá trị thành các đặc trưng nhị phân chỉ chứa giá trị 1 hoặc 0, mỗi mẫu trong đặc trưng phân loại sẽ được biến đổi thành một vectơ có kích thước m chỉ với một trong các giá trị là 1 (biểu thị nó là active); cross-entropy là một hàm mất mát và giá trị của nó có thể được cực tiểu hoá; điều này giúp cho một mạng nơron đánh giá được xác suất (độ chắc chắn) của phép dự

đoán một mẫu dữ liệu tương ứng với một lớp (class); cross entropy là tổng của các xác suất logarit âm; hàm cross entropy được định nghĩa theo công thức sau:

$$-(y\log(p) + (1-y)\log(1-p))$$

đối với mục tiêu làm mịn các bước của gradien xuống để nó có thể hội tụ nhanh hơn, sử dụng thuật toán tối ưu adam (Adaptive Moment Estimation) thông dụng trong các kiến trúc mạng nơron tích chập; các tham số được thiết lập cho bộ tối ưu này gồm learning_rate=0,05; $\beta 1$=0,9; $\beta 2$=0,999 và ε =1e-08.

Mô tả tóm tắt các hình vẽ kèm theo

Hình 1 là hình vẽ dạng sơ đồ minh hoạ bước làm sạch tập dữ liệu tệp mã nguồn ASP.NET chứa đoạn mã độc bằng phương pháp đối sánh;

Hình 2 là hình vẽ dạng sơ đồ minh hoạ bước chuyển hóa mã nguồn mỗi tập tin ASP.NET thành vectơ chuỗi mã lệnh;

Hình 2A là hình vẽ minh hoạ kết quả sau khi biên dịch mã nguồn ứng dụng ASP.NET;

Hình 2B là hình vẽ thể hiện danh mục các tệp tin thư viện liên kết động DLL trong thư mục "bin" của mã nguồn ứng dụng ASP.NET sau khi biên dịch.

Hình 3 là hình vẽ dạng sơ đồ minh hoạ cách thức biểu diễn chuỗi mã lệnh từ tệp tin mã nguồn CIL OpCodes;

Hình 4 là hình vẽ dạng sơ đồ minh hoạ bước huấn luyện mô hình mạng nơron tích chập bằng tập dữ liệu đã được biểu diễn dưới dạng không gian vectơ;

Hình 5 là hình vẽ dạng sơ đồ minh hoạ bước phân tích tệp ASP.NET bằng mô hình mạng nơron tích chập để phát hiện mã độc;

Hình 6 là hình vẽ dạng sơ đồ minh hoạ phương pháp phát hiện đoạn mã độc ứng dụng Web sử dụng ngôn ngữ ASP.NET theo sáng chế.

Mô tả chi tiết sáng chế

Theo một khía cạnh của sáng chế, sáng chế đề xuất phương pháp phát hiện đoạn mã độc trong mã nguồn ứng dụng Web ASP.NET, trong đó phương pháp này sử dụng mô hình mạng nơron tích chập (CNN) để phát hiện các đoạn mã độc (WebShell), phương pháp này bao gồm các bước:

(a) thu thập tập dữ liệu các tệp tin mã nguồn ứng dụng ASP.NET bao gồm các tệp tin lành tính từ các hệ thống quản lý nội dung CMS nổi tiếng như Dotnetnuke, Umbraco, Kentico, … để tạo thành bộ dữ liệu lành tính (bộ dữ liệu Benign) và thu thập các tệp mã nguồn chứa đoạn mã độc 101 để hình thành bộ dữ liệu các đoạn mã độc (bộ dữ liệu Webshell), các tệp tin này đã được thu thập bởi cộng đồng mạng (chẳng hạn như, cộng đồng chia sẻ mã nguồn Github (tennc/webshell, /ysrc/webshell-sample, …));

(b) làm sạch tập dữ liệu tệp mã nguồn ASP.NET chứa đoạn mã độc bằng cách sử dụng phương pháp đối sánh102 (chẳng hạn như, theo bộ quy tắc Yara) để loại bỏ các tệp tin lành tính 103 bị lẫn trong bộ dữ liệu các đoạn mã độc với mục tiêu chỉ giữ lại các tệp chứa đoạn mã độc nhằm mục đích giảm nhiễu trong quá trình huấn luyện mô hình mạng nơron tích chập, hoặc để bổ sung vào bộ dữ liệu các đoạn mã độc hại 104 nếu các tệp tin được xác định là độc hại;

(c) chuyển hóa mã nguồn các tệp tin ASP.NET thành dạng không gian vectơ bằng cách chuyển hóa sang dạng mã lệnh thực thi (opcodes); quá trình này được thực hiện theo 3 bước chính (*hình 2*). Đầu tiên, các tệp tin mã nguồn sẽ được thiết lập trong một dự án ASP.NET trống để thực hiện tiền biên dịch 201 thành các tệp tin thư viện liên kết động DLL (Dynamic Library Link) thông qua công cụ biên dịch ASPNET_COMPILER của Microsoft. Bước tiếp theo, các tệp tin DLL này sẽ được chuyển đổi thành chuỗi mã lệnh opcodes bằng công cụ MSIL Disassembler. Cuối cùng, ánh xạ chuỗi lệnh đó thành vector chuỗi chỉ mục tương ứng với mã lệnh opcodes;

(d) loại bỏ các không gian vectơ không phù hợp với độ dài quá ngắn hoặc các không gian vectơ đã tồn tại 206; trong đó việc loại bỏ các không gian vectơ không phù hợp với độ dài quá ngắn do các không gian vectơ này là quá nhỏ đối với một đoạn mã độc; trong đó việc loại bỏ các không gian vectơ đã tồn tại trong bộ dữ liệu để đảm bảo mỗi không gian vectơ trong bộ dữ liệu là duy nhất để giảm thiểu kích thước của tập dữ liệu và tăng tốc độ huấn luyện mô hình mạng nơron tích chập;

(e) huấn luyện mô hình mạng nơron tích chập (CNN) với bộ dữ liệu sạch và bộ dữ liệu các đoạn mã độc đã làm sạch; mô hình này được thiết lập với ba lớp tích chập để học đặc trưng, sử dụng phương pháp hồi quy tuyến tính để phân lớp với thuật toán tối ưu Adam; trong đó:

- tầng đầu vào 401là tập dữ liệu các tệp tin sạch thuộc bộ dữ liệu sạch (Benign) và tệp tin mã độc thuộc bộ dữ liệu các đoạn mã độc (bộ dữ liệu Webshell) sau khi đã loại bỏ các không gian vectơ không phù hợp với độ dài quá ngắn hoặc các không gian vectơ đã tồn tại;

- tầng thứ thứ hai 402 là tầng tích chập được cấu thành từ ba lớp tích chập; tiếp đó, các lớp tích chập này sẽ được ghép nối lại thành một lớp tích chập hợp nhất theo cơ chế nối chuỗi (concatenate), mục đích cuối cùng của tầng thứ hai để thực hiện phát hiện ra những đặc trưng hiệu quả nhất được thể hiện bởi ma trận đặc trưng (feature map).

- tầng thứ ba 403 là lớp gộp hay còn gọi là lớp tổng hợp (pooling layer) nó làm giảm số tham số mà ta cần phải tính toán, từ đó giảm thời gian tính toán, tránh quá mức (overfitting);

- tầng thứ tư 404 là tầng phân lớp, tầng này có chức năng chuyển ma trận đặc trưng ở tầng trước thành vectơ chứa xác suất của các đối tượng cần được dự đoán;

(f) phân tích tệp mã nguồn ASP.NET với mô hình mạng nơron tích chập đã được huấn luyện để xác định có chứa đoạn mã độc (WebShell) hay không.

Trong bản mô tả sáng chế này, "phương pháp lai" được hiểu là phương pháp sử dụng kết hợp phương pháp so khớp mẫu thông qua sử dụng bộ quy tắc yara và phương pháp học sâu với mạng nơron tích chập; "mã độc Webshell" được hiểu là một dạng mã độc có nhiều chức năng được sử dụng bởi tin tặc nhằm mục đích điều khiển, truy cập trái phép từ xa đối với máy chủ ứng dụng Web.Webshell thường được viết bằng nhiều loại ngôn ngữ và thường thì chính là ngôn ngữ mà trang Web đó đang sử dụng; "bộ quy tắc Yara" được hiểu là tập hợp các quy tắc, thường được thể hiện dưới dạng các biểu thức chính quy, có chứa kèm các dấu hiệu để mô tả cách thức nhận diện các đoạn mã độc.

Để xây dựng phương pháp này, tác giả sử dụng hai bộ dữ liệu để huấn luyện mô hình học sâu. Bộ dữ liệu chứa các tệp mã nguồn ASP.NET sạch được gọi tắt là "bộ dữ liệu sạch Benign"; trong khi đó, bộ dữ liệu chứa các tệp mã nguồn ASP.NET có chứa đoạn mã độc Webshell được gọi tắt là "bộ dữ liệu Webshell".

Sau đây giải pháp sẽ được mô tả chi tiết có dựa vào các hình vẽ kèm theo.

Các ứng dụng Web sử dụng .NET Framework hiện nay được sử dụng rất rộng rãi, đặc biệt đối với rất nhiều các hệ thống phục vụ chính phủ điện tử Việt Nam. Chính vì thế, việc đảm bảo an toàn thông tin cho các ứng dụng Web ASP.NET là cấp thiết, đặc biệt cần phải đảm bảo an toàn ngay từ khâu phát triển ứng dụng đến triển khai thực tế (theo mô hình DevSecOps). Trong sáng chế này, tác giả tập trung xây dựng phương pháp lai để phát hiện đoạn mã độc Webshell trong các tệp mã nguồn ứng dụng ASP.NET.

Phương pháp lai đề xuất được xây dựng với ý tưởng áp dụng phương pháp đối sánh mẫu bằng cách áp dụng bộ quy tắc Yara để làm sạch và xây dựng được bộ dữ liệu Webshell và bộ dữ liệu sạch Benign có chất lượng. Từ đó, tác giả xây dựng mô hình biểu diễn tệp mã nguồn ứng dụng ASP.NET dưới dạng một vectơ chuỗi mã lệnh. Sau đó, tất cả các bộ dữ liệu sạch Benign và Webshell sẽ được chuyển đổi thành tập các vectơ chuỗi mã lệnh phục vụ huấn luyện mô hình học máy. Tác giả sử dụng phương pháp học sâu với mô hình mạng nơron tích chập gồm ba lớp tích chập

phục vụ quá trình học đặc trưng và bộ phân lớp nhị phân dựa vào hồi quy. Từ mô hình đã huấn luyện, tác giả xây dựng quy trình để phát hiện đoạn mã độc Webshell trong tệp tin mã nguồn ASP.NET.

Để xây dựng bộ dữ liệu sạch Benign, tác giả đã tiến hành thu thập các tệp mã nguồn sạch từ các hệ quản trị nội dung CMS, framework nguồn mở phổ biến trên thế giới như DotNetNuke, Umbraco, Kentico, Sitefinity, Orchard Project, v.v.. Tổng cộng tập dữ liệu đã thu thập được khoảng hơn 2000 tệp tin mã nguồn.

Đới với việc xây dựng bộ dữ liệu Webshell tốt, tác giả đã tiến hành thu thập các tệp .aspx chứa Webshell từ một số nguồn tổng hợp tin cậy trên Github, mặc dù vậy, qua kiểm tra cho thấy vẫn có những tệp tin sạch nhất định bị lẫn trong bộ dữ liệu Webshell. Do đó, các tệp tin này cũng cần được làm sạch để loại đi những tệp không thực sự chứa đoạn mã độc Webshell. Lý do cần loại bỏ xuất phát từ việc quá trình huấn luyện mô hình học sâu sẽ bị nhiễu, giảm độ chính xác nếu có cả tệp sạch trong bộ dữ liệu Webshell. Đây cũng chính là luận điểm tác giả sẽ yêu cầu bảo hộ với việc làm sạch bộ dữ liệu Webshell bằng phương pháp so khớp mẫu theo bộ quy tắc Yara.

Hình 1 minh họa quá trình sử dụng phương pháp đối sánh theo bộ quy tắc Yara để làm sạch bộ dữ liệu Webshell. Tại bước thu thập tập dữ liệu các tệp chứa các đoạn mã độc 101, tác giả tiến hành thu thập từ các nguồn có số sao được đánh giá cao trên nền tảng chia sẻ mã nguồn Github (*https://github.com/*). Ở bước này, tác giả đã thu thập 517 tệp nghi có chứa Webshell. Đối với các tệp tin aspx độc hại, mặc dù có thể dễ dàng thu thập được từ các nguồn có độ tin cậy cao trên Github, tuy nhiên, các nguồn tập hợp này không thể tránh khỏi bị lẫn các tệp tin lành tính, làm ảnh hưởng rất nhiều chất lượng của tập dữ liệu. Vì vậy việc làm sạch các tệp tin lành tính trong tập dữ liệu các tệp tin độc hại là vô cùng cần thiết.

Tại bước làm sạch tập dữ liệu tệp mã nguồn ASP.NET chứa đoạn mã độc bằng cách sử dụng phương pháp đối sánh102, từ bộ dữ liệu thô này, tác giả sẽ tiến hành sử dụng công cụ yara với tập luật đã được tác giả xây dựng từ công trình

"GuruWS: A Hybrid Platform for Detecting Malicious Web Shells and Web Application Vulnerabilities"(*Van-Giap Le, Huu-Tung Nguyen, Duy-Phuc Pham, Van-On Phung, Ngoc-Hoa Nguyen, "GuruWS: A Hybrid Platform for Detecting Malicious Web Shells and Web Application Vulnerabilities", Transactions on Computational Collective Intelligence XXXII. Lecture Notes in Computer Science, vol 11370. Springer, Cham 978-3-319-90286-9, 2019 (ISI WoS, Scopus Journal)*). Bộ quy tắc Yara với đặc điểm là phát hiện các đoạn mã độc bằng phương pháp đối sánh mẫu, nên sẽ rất hiệu quả đối với các mẫu đoạn mã độc đã biết. Quá trình làm sạch được thực hiện với tất cả các tệp aspx thô đã thu thập. Với mỗi tệp aspx, nếu bộ công cụ yara so khớp với tập luật mẫu cho kết quả tệp tin lành tính tại 103, tệp tin đó sẽ không được đưa vào bộ dữ liệu Webshell. Nếu kết quả trả về là tệp tin độc hại tại 104, tác giả sẽ đưa tệp tin aspx vào bộ dữ liệu Webshell.

Từ hai bộ dữ liệu sạch Benign và Webshell để có được dữ liệu đầu vào cho quá trình huấn luyện mô hình mạng nơron tích chập, tác giả đã sử dụng phương pháp biểu diễn tệp tin mã nguồn ASP.NET sang vectơ chuỗi mã lệnh. Hình 2 mô tả phương pháp đề xuất biểu diễn mã nguồn ASP.NETdưới dạng không gian vectơ bằng cách chuyển hóa sang dạng mã lệnh OpCode (Operation Code). Hiện tập mã lệnh này bao gồm 229 lệnh đối với phiên bản .NET Framework 5.0 (*https://docs.microsoft.com/en-us/dotnet/api/system.reflection.emit.opcodes?view=net-5.0*).

Sử dụng ngôn ngữ biên dịch, nên các ứng dụng web ASP.NET phải được biên dịch trước. Khác với ngôn ngữ PHP sẽ đọc và thông dịch mỗi khi trang web được yêu cầu, ASP.Net biên dịch những trang web động thành những tập tin DLL mà Server có thể thi hành nhanh chóng và hiệu quả. Do vậy, để biên dịch được các tệp tin mã nguồn cần phải thiết lập một dự án ứng dụng web ASP.NET ở bước 201 chứa các tệp tin mã nguồn cần biên dịch, và sử dụng bộ công cụ biên dịch ASP.NET (aspnet_compiler.exe) được cung cấp kèm theo bộ .NET Framework. Công cụ biên dịch ASP.NET cho phép sử dụng dưới dạng giao diện dòng lệnh, với cú pháp:

aspnet_compiler -p "đường dẫn thư mục dự án ứng dụng ASP.NET" -v / -f -u

"đường dẫn thư mục đích"

Sau khi thực thi câu lệnh biên dịch trên, ứng dụng ASP.NET sẽ được biên dịch và kết quả được lưu trữ tại đường dẫn thư mục đích như trong Hình 2A. Thư mục "bin" nằm trong thư mục đích Hình 2B, chứa các tệp tin thư viện liên kết động DLL, tất cả các tệp tin aspx, ascx đều sẽ được biên dịch thành các tệp tin DLL này với tiền tố App_Web. Các tệp tin có phần mở rộng .compiled lưu trữ dưới dạng XML thông tin ánh xạ tệp tin mã nguồn với tệp được biên dịch.

Ví dụ dưới đây là nội dung tệp tin "*contact.aspx.cdcab7d2.compiled*" chứa thông tin biên dịch của tệp tin "*contact.aspx*"sẽ là tệp tin "*App_Web_on0t4wgc.dll*" được lưu tại trường "*assembly*".

```
<?xml version="1.0" encoding="utf-8"?>
<preserve resultType="3" virtualPath="/Contact.aspx" hash="ffffffffa01bf7d47"
filehash="61ebcf8b0a636be7" flags="110000" assembly="App_Web_on0t4wgc" type="ASP.contact_aspx">
<filedeps>
<filedep name="/Contact.aspx" />
<filedep name="/Contact.aspx.cs" />
<filedep name="/navigation.ascx" />
<filedep name="/navigation.ascx.cs" />
</filedeps>
</preserve>
```

Trong trường hợp biên dịch không thành công 202, có nghĩa là mã nguồn tệp tin có lỗi, khi đó tệp tin này sẽ bị loại bỏ khỏi tập dữ liệu. Trong trường hợp 203, biên dịch thành công, các tệp tin DLL sẽ tiếp tục được chuyển đổi thành mã lệnh OpCode bằng bộ công cụ IL Disassembler (*Ilasm.exe*) 204. Cú pháp để thực hiện lệnh chuyển đổi là:

ildasm "đường dẫn tệp tin DLL"/output "đường dẫn tệp tin mã lệnh OpCode"

Ví dụ với một đoạn mã lệnh đơn giản viết bằng C#:

```
static void Main(string[] args)
{
 Console.WriteLine("Hello World");
}
```

Khi được chuyển đổi thành mã lệnh OpCode sẽ có dạng:

```
.assembly extern mscorlib
{
  .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
  .ver 4:0:0:0
}
.assembly HelloWorld
{
 ----lines omitted
}
.module HelloWorld.exe
.class private auto ansi beforefieldinit HelloWorld.Program
       extends [mscorlib]System.Object
{
 .method private hidebysig static void  Main(string[] args) cil managed
  {
    .entrypoint
    // Code size       13 (0xd)
    .maxstack  8
    IL_0000:  nop
    IL_0001:  ldstr      "Hello World"
    IL_0006:  call       void [mscorlib]System.Console::WriteLine(string)
    IL_000b:  nop
    IL_000c:  ret
  } // end of method Program::Main

  .method public hidebysig specialname rtspecialname
         instance void  .ctor() cil managed
  {
    .maxstack  8
    IL_0000:  ldarg.0
    IL_0001:  call       instance void [mscorlib]System.Object::.ctor()
    IL_0006:  ret
  } // end of method Program::.ctor
```

} // end of class HelloWorld.Program

CIL Opcodes thực chất là mã nhị phân (0x01) nhưng có cách biểu diễn thân thiện tương ứng (như đối với 0x01, tương ứng với tên hàm "Break") để hỗ trợ các lập trình viên có thể đọc hiểu, gỡ lỗi và viết mã trực tiếp bằng ngôn ngữ trung gian.

Đoạn mã lệnh OpCode sẽ được chuyển đổi thành dạng không gian vectơ 204 tương ứng, như trong mô tả Hình 3, khi đó ta thu được vectơ chuỗi lệnh là:

[0, 176, 162, 40, 176, 185, 0, 97, 40, 185]

Số 0 trong vectơ mã lệnh này thể hiện cho việc bắt đầu khai báo một hàm, các giá trị tiếp đó thể hiện chỉ mục các hàm trong danh sách tập mã lệnh OpCode. Ví dụ giá trị 176 thể hiện cho hàm "*nop*", 162 thể hiện cho hàm "*ldstr*". Với cách biểu diễn này, chuỗi vectơ mã lệnh sẽ có thể biểu diễn hầu hết tính chất của một đoạn mã nguồn ứng dụng Web ASP.NET trong đó bao gồm cả những dữ liệu có thể được coi như là đặc trưng để phát hiện đoạn mã độc, điều này rất quan trọng vì thông qua chuỗi mã lệnh này sẽ là dữ liệu đầu vào để huấn luyện các mô hình mạng nơ ron tích chập. Cách thức chuyển hóa và biểu diễn mã nguồn ứng dụng Web ASP.NET này cũng chính là một trong những nội dung bảo hộ của sáng chế này.

Ở bước loại bỏ các không gian vectơ không phù hợp với độ dài quá ngắn hoặc các không gian vectơ đã tồn tại 205,loại bỏ các tệp tin mà vectơ chuỗi lệnh của nó có giá trị vô nghĩa hoặc bị trùng lắp trong tập dữ liệu. Những vectơ vô nghĩa là những vectơ khi thực hiện lệnh biên dịch từ mã nguồn thành opcode vì một lý do nào đó gặp lỗi, hoặc có độ dài quá bé đối với một đoạn mã độc (hiện được xác định bằng độ vectơ chuỗi lệnh Webshell ngắn nhất trừ đi một), do đó không có giá trị sử dụng để huấn luyện mô hình học sâu. Bên cạnh đó, cũng sẽ có những trường hợp các nhiều vectơ chuỗi lệnh có giá trị trùng nhau khi đó chỉ cần lưu lại một không gian vectơ duy nhất để giảm thiểu kích thước của tập dữ liệu và tăng tốc độ huấn luyện mô hình.

Sau quá trình chuyển đổi sang không gian vectơ chuỗi mã lệnh và loại bỏ đi những tệp không có giá trị, bộ dữ liệu sạch Benign của tác giả thu được 1876 vectơ; bộ dữ liệu Webshell có 489 vectơ. Đây là các bộ dữ liệu được tác giả sử dụng để huấn luyện mô hình học sâu.

Hình 4 mô tả kiến trúc của mô hình mạng nơron tích chập CNN để dự đoán Webshell trong mã nguồn ứng dụng, trong đó mô hình được hình thành nên bởi bốn tầng.

Đầu tiên là tầng đầu vào 401, đây chính là tập dữ liệu các tệp tin sạch Benignvà tệp tin WebShell dưới dạng các vectơ chuỗi lệnh có kích thước tối đa là 20.000. Giá trị này được lựa chọn do chuỗi lệnh dài nhất trong các tệp tin mã nguồn ứng dụng ASP.NETở cả 2 tập dữ liệu có độ dài không quá 20.000 lệnh. Việc lựa chọn độ dài của vectơ đầu vào có ý nghĩa quan trọng trong việc gia tăng tốc độ huấn luyện nhưng không làm ảnh hưởng đến độ chính xác khi dự đoán của mô hình.

Tầng thứ thứ hai 402 gọi là tầng tích chập được cấu thành từ ba lớp tích chập gồm 128bộ lọc (filters) có kích thước lần lượt là 3, 4, 5 sử dụng hàm kích hoạt phi tuyến ReLU $f(x) = \max(0, x)$ trong đó x là giá trị đầu vào để tạo ra thông tin trừu tượng hơn (Abstract/higher-level) cho các lớp tiếp theo. Tiếp đó, các lớp tích chập này sẽ được ghép nối lại thành một lớp tích chập hợp nhất theo cơ chế nối chuỗi (*concatenate*), mục đích cuối cùng của tầng thứ hai để thực hiện phát hiện ra những đặc trưng hiệu quả nhất được thể hiện bởi ma trận đặc trưng (feature map).

Tầng thứ ba 403 là lớp gộp hay còn gọi là lớp tổng hợp (pooling layer)nó làm giảm số tham số mà ta cần phải tính toán, từ đó giảm thời gian tính toán, tránh quá mức (overfitting). Mô hình đề xuất sử dụng lớp tổng hợp tối đa (global_max_pooling) với kích thước bằng với kích thước của dữ liệu đầu vào và hệ số sụt giảm (dropout) 0,6 nghĩa là trong quá trình huấn luyện mô hình, với mỗi lần thực hiện vòng lặp, ta ngẫu nhiên loại bỏ 60% số lượng nút trong lớp. Mục đích chính của tầng này là giảm chiều của tầng trước đó, loại bỏ những đặc trưng không còn cần thiết, thay vào đó giữ lại các đặc trưng đã đủ để phân loại đối tượng. Giá trị

0,6 được tác giả lựa chọn là giá trị tối ưu dựa vào các kết quả thực nghiệm với nhiều giá trị khác nhau.

Tầng thứ tư 404 là tầng phân lớp, tầng này có chức năng chuyển ma trận đặc trưng ở tầng trước thành vectơ chứa xác suất của các đối tượng cần được dự đoán. Tại tầng này, mô hình đề xuất sử dụng các tham số:

Hàm kích hoạt softmax, là một cách ràng buộc đầu ra của các mạng nơron phải có tổng bằng 1. Qua đó, các giá trị đầu ra của hàm softmax có thể được coi như là một phân phối xác suất của các biến đầu ra hay nói cách khác hàm softmax sẽ chuyển đổi giá trị đầu ra của mạng nơron bằng cách chia cho tổng giá trị. Hàm softmax được thể hiện trong công thức sau

$$softmax(X)_{ij} = \frac{\exp(X_{ij})}{\sum_k \exp(X_{ik})}$$

Hàm mất mát cross entropy, sử dụng để so sánh khoảng cách giữa các giá trị đầu ra của softmax và quá trình biến đổi từng giá trị thành các đặc trưng nhị phân (One-hot encoding), trong đó quá trình biến đổi từng giá trị thành các đặc trưng nhị phân (One-hot encoding) là quá trình biến đổi từng giá trị thành các đặc trưng nhị phân chỉ chứa giá trị 1 hoặc 0, mỗi mẫu trong đặc trưng phân loại sẽ được biến đổi thành một vectơ có kích thước m chỉ với một trong các giá trị là 1 (biểu thị nó là active). Cross-entropy là một hàm mất mát và giá trị của nó có thể được cực tiểu hoá. Điều này giúp cho một mạng nơron đánh giá được xác suất (độ chắc chắn) của phép dự đoán một mẫu dữ liệu tương ứng với một lớp (class). Cross entropy là tổng của các xác suất logarit âm. Hàm cross entropy được định nghĩa theo công thức sau:

$$-(ylog(p) + (1-y)\log(1-p))$$

Đối với mục tiêu làm mịn các bước của gradien xuống để nó có thể hội tụ nhanh hơn, tác giả sử dụng thuật toán tối ưu adam (viết tắt của Adaptive Moment Estimation) thông dụng trong các kiến trúc mạng nơron tích chập. Các tham số

được thiết lập cho bộ tối ưu này gồm learning_rate=0,05;β_1=0,9;β_2=0,999 vàε =1e-08.

Trong khuôn khổ sáng chế này, bài toán phân loại tệp tin mã nguồn có hai lớp tương ứng với hai số nhận giá trị trong khoảng từ 0 đến 1, lớp phân loại này sẽ chuyển ma trận đặc trưng của lớp trước thành vectơ có hai chiều thể hiện xác suất tương ứng với tệp tin là sạch hoặc là WebShell.

Hình 5 mô tả quy trình dự đoán một tệp tin mã nguồn aspx có phải là tệp tin WebShell hay khôngbằng cách sử dụng mô hình mạng nơron tích chập đã được sáng kiến đề xuất xây dựng. Tương tự như quy trình xây dựng tập dữ liệu để huấn luyện mô hình, tệp tin mã nguồn sẽ được biên dịch thành chuỗi lệnh từ một dự án ứng dụng Web ASP.NET bằng các bộ công cụ biên dịch aspnet_compiler.exenếu quá trình biên dịch không thành công 502, có nghĩa là tệp tin này không có khả năng thực thi được nó sẽ được phân loại là tệp tin lành tính. Trong trường hợp biên dịch thành công, bộ công cụ chuyển đổi sang mã CIL OpCode **ildsam.exe** sẽ được sử dụng để biễn các tệp tin mã nguồn đã được biên dịch thành chuỗi lệnh OpCode 503, mô hình đề xuất sẽ biểu diễn chuỗi lệnh này dưới dạng vectơ chỉ mục, nếu vectơ chỉ mục này không có giá trị hoặc giá trị vô nghĩa, tệp tin sẽ là lành tính 504. Ngược lại, vectơ chỉ mục này sẽ được sử dụng làm đầu vào để mô hình CNN đưa ra dự đoán, nếu giá trị dự đoán agrmax của mô hình bằng 0 thì tệp tin là lành tính, bằng 1 thì tệp tin có chứa đoạn mã độc Webshell.

Hình 6 là hình vẽ dạng sơ đồ minh hoạphương pháp phát hiện đoạn mã độctrong mã nguồn ứng dụng Web ASP.NET theo sáng chế.

Với bộ dữ liệu sạch Benign và Webshell đã làm sạch nêu trên, sau khi huấn luyện mô hình học sâu tích chập CNN với các tham số nêu trên, phương pháp phát hiện đoạn mã độc trong mã nguồn ứng dụng Web ASP.NET được đề xuất trong sáng chế đã đạt tỷ lệ chính xác là 98,62%, tỉ lệ nhận dạng nhầm là 0,92%.

YÊU CẦU BẢO HỘ

1. Phương pháp phát hiện đoạn mã độc trong mã nguồn ứng dụng Web sử dụng ngôn ngữ ASP.NET bao gồm các bước:

(a) thu thập tập dữ liệu các tệp tin mã nguồn ứng dụng Web ASP.NET bao gồm các tệp tin lành tính để tạo thành bộ dữ liệu sạch (Benign), các tập tin lành tính này được thu thập từ các nhà cung cấp ứng dụng, và thu thập tập dữ liệu các tệp chứa các đoạn mã độc (101) để tạo thành bộ dữ liệu các đoạn mã độc (bộ dữ liệu Webshell), các tệp chứa các đoạn mã độc này đã được tập hợp bởi cộng đồng mạng;

(b) làm sạch tập dữ liệu tệp mã nguồn ứng dụng Web ASP.NET chứa đoạn mã độc bằng cách sử dụng phương pháp đối sánh (102) để loại bỏ các tệp tin lành tính (103) bị lẫn trong bộ dữ liệu các đoạn mã độc với mục tiêu chỉ giữ lại các tệp chứa đoạn mã độc nhằm mục đích giảm nhiễu trong quá trình huấn luyện mô hình mạng noron tích chập, hoặc để bổ sung vào bộ dữ liệu các đoạn mã độc hại (104) nếu các tệp tin được xác định là độc hại;

(c) biểu diễn các tệp tin mã nguồn ứng dụng ASP.NET thành dưới dạng không gian vectơ bằng cách chuyển hóa sang dạng mã lệnh thực thi (CIL OpCodes); quá trình này được thực hiện theo 3 bước chính, đầu tiên, các tệp tin mã nguồn sẽ được thiết lập trong một dự án ASP.NET trống để thực hiện tiền biên dịch 201 thành các tệp tin thư viện liên kết động DLL (Dynamic Library Link) thông qua công cụ biên dịch ASPNET_COMPILER của Microsoft nếu quá trình biên dịch có lỗi, các tệp tin mã nguồn sẽ bị loại bỏ khỏi tập dữ liệu, các tệp tin DLL này sẽ được chuyển đổi thành chuỗi mã lệnh opcodes bằng công cụ MSIL Disassembler, cuối cùng, ánh xạ chuỗi lệnh đó thành vector chuỗi chỉ mục tương ứng với mã lệnh opcodes;

(d) loại bỏ các không gian vectơ không phù hợp với độ dài quá ngắn hoặc các không gian vectơ đã tồn tại (205); trong đó việc loại bỏ các không gian vectơ không phù hợp với độ dài quá ngắn do các không gian vectơ này là quá nhỏ đối với một đoạn mã độc; trong đó việc loại bỏ các không gian vectơ đã tồn

tại trong bộ dữ liệu để đảm bảo mỗi không gian vectơ trong bộ dữ liệu là duy nhất để giảm thiểu kích thước của tập dữ liệu và tăng tốc độ huấn luyện mô hình mạng nơron tích chập;

(e) huấn luyện mô hình mạng nơron tích chập (CNN) với bộ dữ liệu sạch và bộ dữ liệu các đoạn mã độc đã làm sạch; mô hình này được thiết lập với ba lớp tích chập để học đặc trưng, sử dụng phương pháp hồi quy tuyến tính để phân lớp với thuật toán tối ưu Adam; trong đó:

- tầng đầu vào (401) là tập dữ liệu các tệp tin sạch thuộc bộ dữ liệu sạch (Benign) và tệp tin mã độc thuộc bộ dữ liệu các đoạn mã độc (bộ dữ liệu Webshell) sau khi đã loại bỏ các không gian vectơ không phù hợp với độ dài quá ngắn hoặc các không gian vectơ đã tồn tại;

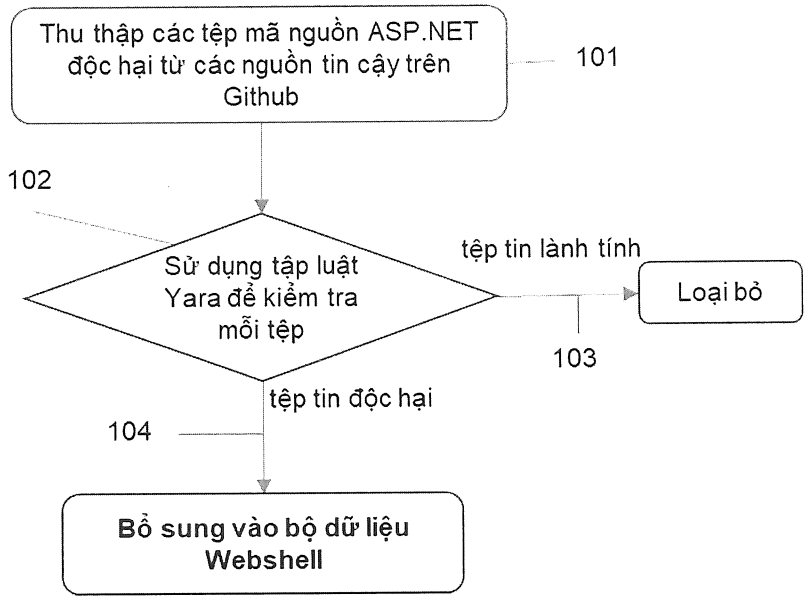
- tầng thứ thứ hai (402) là tầng tích chập được cấu thành từ ba lớp tích chập; tiếp đó, các lớp tích chập này sẽ được ghép nối lại thành một lớp tích chập hợp nhất theo cơ chế nối chuỗi (concatenate), mục đích cuối cùng của tầng thứ hai để thực hiện phát hiện ra những đặc trưng hiệu quả nhất được thể hiện bởi ma trận đặc trưng (feature map).

- tầng thứ ba (403) là lớp gộp hay còn gọi là lớp tổng hợp (pooling layer) nó làm giảm số tham số mà ta cần phải tính toán, từ đó giảm thời gian tính toán, tránh quá mức (overfitting);

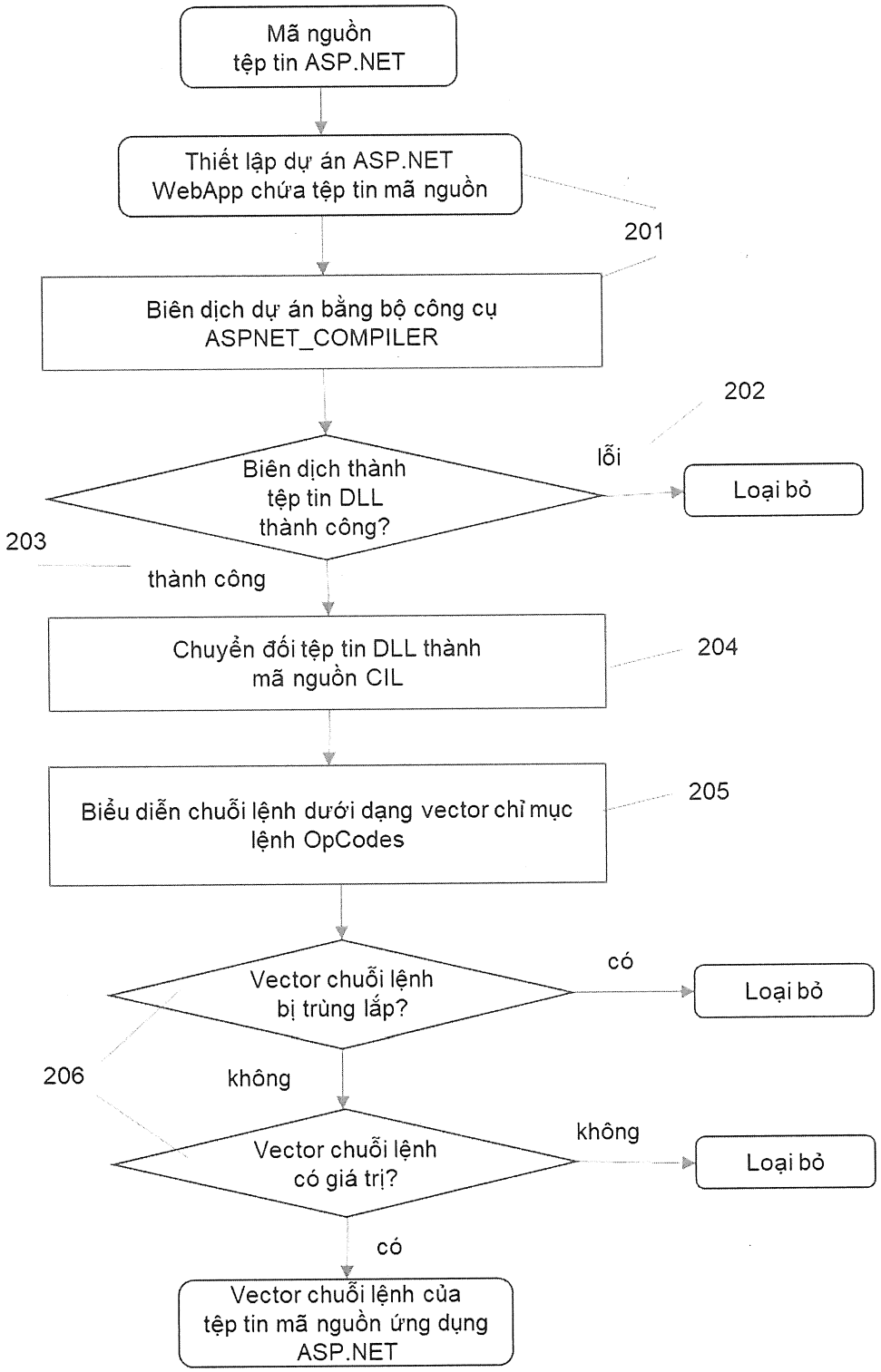
- tầng thứ thứ tư (404) là tầng phân lớp, tầng này có chức năng chuyển ma trận đặc trưng ở tầng trước thành vectơ chứa xác suất của các đối tượng cần được dự đoán;

(f) phân tích tệp tin mã nguồn ứng dụng Web ASP.NET để phát hiện có chứa đoạn mã độc hay không được thực hiện theo 3 bước chính: tệp tin mã nguồn sẽ được biên dịch thành tệp tin thư viện liên kết động DLL thông qua công cụ ASPNET_COMPILER, nếu biên dịch gặp lỗi, tức là phần mã nguồn tệp tin không có khả năng hoạt động, khi đó tệp tin sẽ được phân loại là tệp tin sạch 502; tệp tin DLL sẽ được chuyển đổi thành vectơ mã lệnh CIL OpCode, nếu vectơ mã lệnh này vô giá trị hoặc có độ dài quá ngắn, tệp tin cũng sẽ được

phân loại là tệp tin sạch 504, vectơ mã lệnh được đưa vào mô hình mạng nơron tích chập đã được huấn luyện để xác định có chứa đoạn mã độc (WebShell) hay không.


Thu thập các tệp mã nguồn ASP.NET độc hại từ các nguồn tin cậy trên Github
101
102
Sử dụng tập luật Yara để kiểm tra mỗi tệp
tệp tin lành tính
Loại bỏ
103
tệp tin độc hại
104
Bổ sung vào bộ dữ liệu Webshell

HÌNH 1


Mã nguồn
tệp tin ASP.NET
Thiết lập dự án ASP.NET
WebApp chứa tệp tin mã nguồn
201
Biên dịch dự án bằng bộ công cụ
ASPNET_COMPILER
202
Biên dịch thành
tệp tin DLL
thành công?
lỗi
Loại bỏ
203
thành công
Chuyển đổi tệp tin DLL thành
mã nguồn CIL
204
Biểu diễn chuỗi lệnh dưới dạng vector chỉ mục
lệnh OpCodes
205
Vector chuỗi lệnh
bị trùng lắp?
có
Loại bỏ
206
không
Vector chuỗi lệnh
có giá trị?
không
Loại bỏ
có
Vector chuỗi lệnh của
tệp tin mã nguồn ứng dụng
ASP.NET

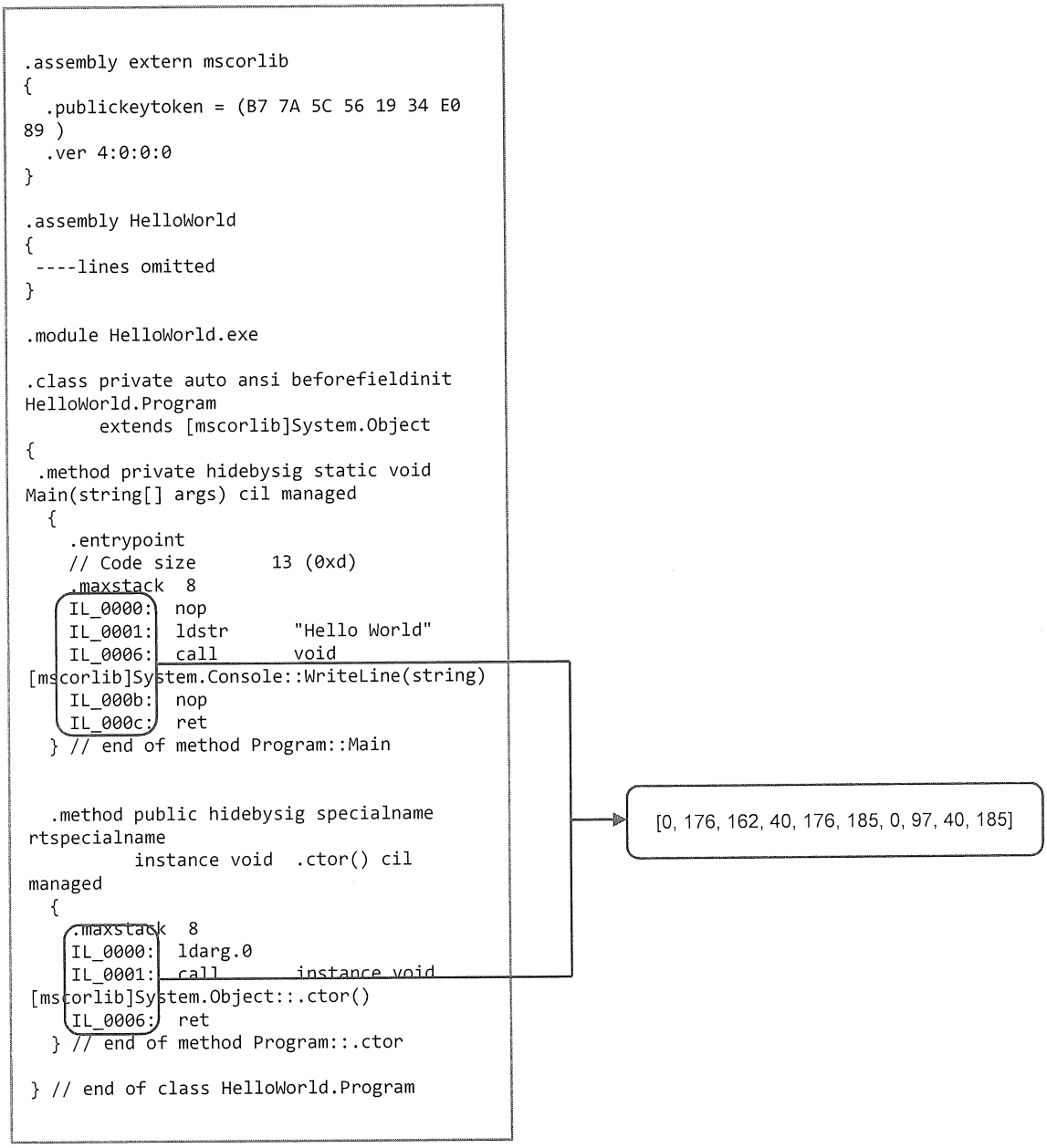
HÌNH 2

benNotStallmen	12/5/2020 9:08 AM	File folder	
bin	12/5/2020 9:08 AM	File folder	
images	12/5/2020 9:08 AM	File folder	
~Stadase_diagram	9/16/2020 10:48 AM	Microsoft Word D...	1 KB
AboutUs.aspx	12/5/2020 9:08 AM	ASP.NET Server Pa...	1 KB
betadase_diagram	9/16/2020 10:48 AM	Microsoft Word D...	46 KB
Contact.aspx	12/5/2020 9:08 AM	ASP.NET Server Pa...	1 KB
Default.aspx	12/5/2020 9:08 AM	ASP.NET Server Pa...	1 KB
default	9/16/2020 10:48 AM	Cascading Style S...	7 KB
Description.aspx	12/5/2020 9:08 AM	ASP.NET Server Pa...	1 KB
Game.aspx	12/5/2020 9:08 AM	ASP.NET Server Pa...	1 KB
PrecompiledApp.config	12/5/2020 9:08 AM	XML Configuratio...	1 KB
Shop.aspx	12/5/2020 9:08 AM	ASP.NET Server Pa...	1 KB
Shopping_Cart.aspx	12/5/2020 9:08 AM	ASP.NET Server Pa...	1 KB
web.config	9/16/2020 10:48 AM	XML Configuratio...	9 KB

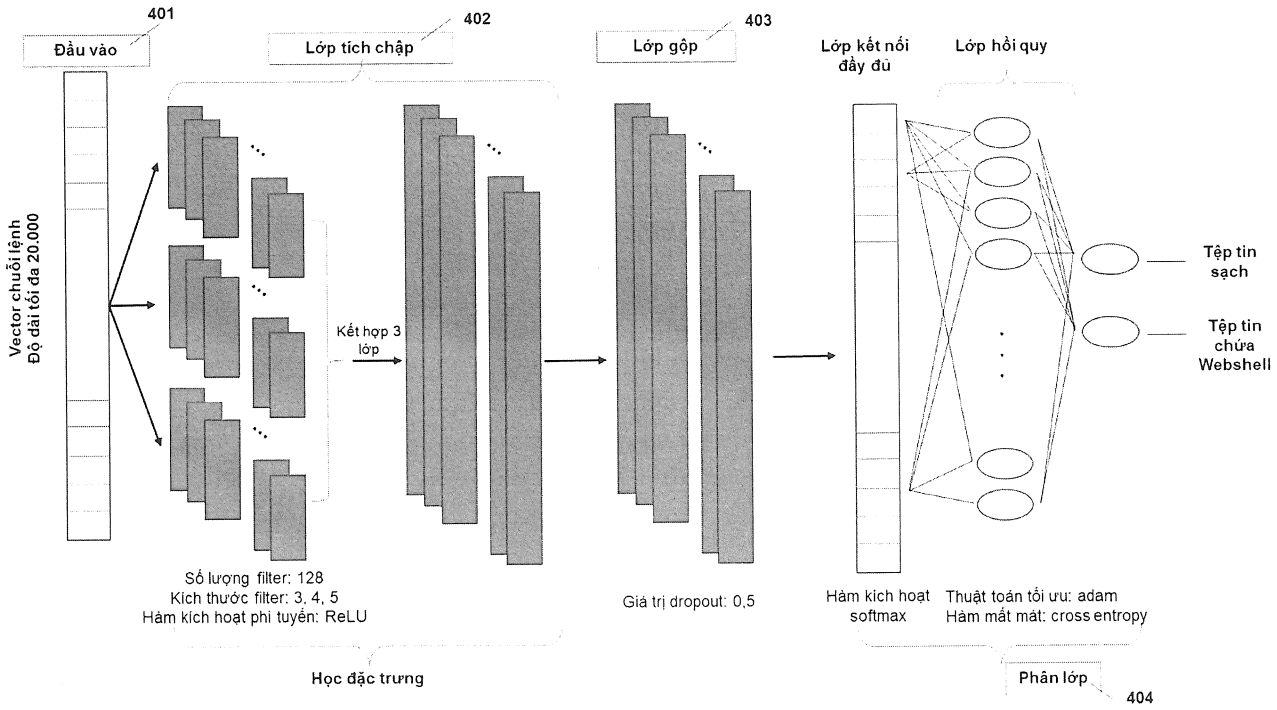
HÌNH 2A

aboutus.aspx.cdcab7d2.compiled	12/5/2020 9:09 AM	COMPILED File	1 KB
App_Code.compiled	12/5/2020 9:09 AM	COMPILED File	1 KB
App_Code.dll	12/5/2020 9:09 AM	Application exten...	7 KB
App_Web_iz5rbxtk.dll	12/5/2020 9:09 AM	Application exten...	41 KB
App_Web_on0t4wgc.dll	12/5/2020 9:09 AM	Application exten...	14 KB
App_Web_tl1qosny.dll	12/5/2020 9:09 AM	Application exten...	7 KB
contact.aspx.cdcab7d2.compiled	12/5/2020 9:09 AM	COMPILED File	1 KB
default.aspx.cdcab7d2.compiled	12/5/2020 9:09 AM	COMPILED File	1 KB
description.aspx.cdcab7d2.compiled	12/5/2020 9:09 AM	COMPILED File	1 KB
game.aspx.cdcab7d2.compiled	12/5/2020 9:09 AM	COMPILED File	1 KB
navigation.ascx.cdcab7d2.compiled	12/5/2020 9:09 AM	COMPILED File	1 KB
shop.aspx.cdcab7d2.compiled	12/5/2020 9:09 AM	COMPILED File	1 KB
shopping_cart.aspx.cdcab7d2.compiled	12/5/2020 9:09 AM	COMPILED File	1 KB

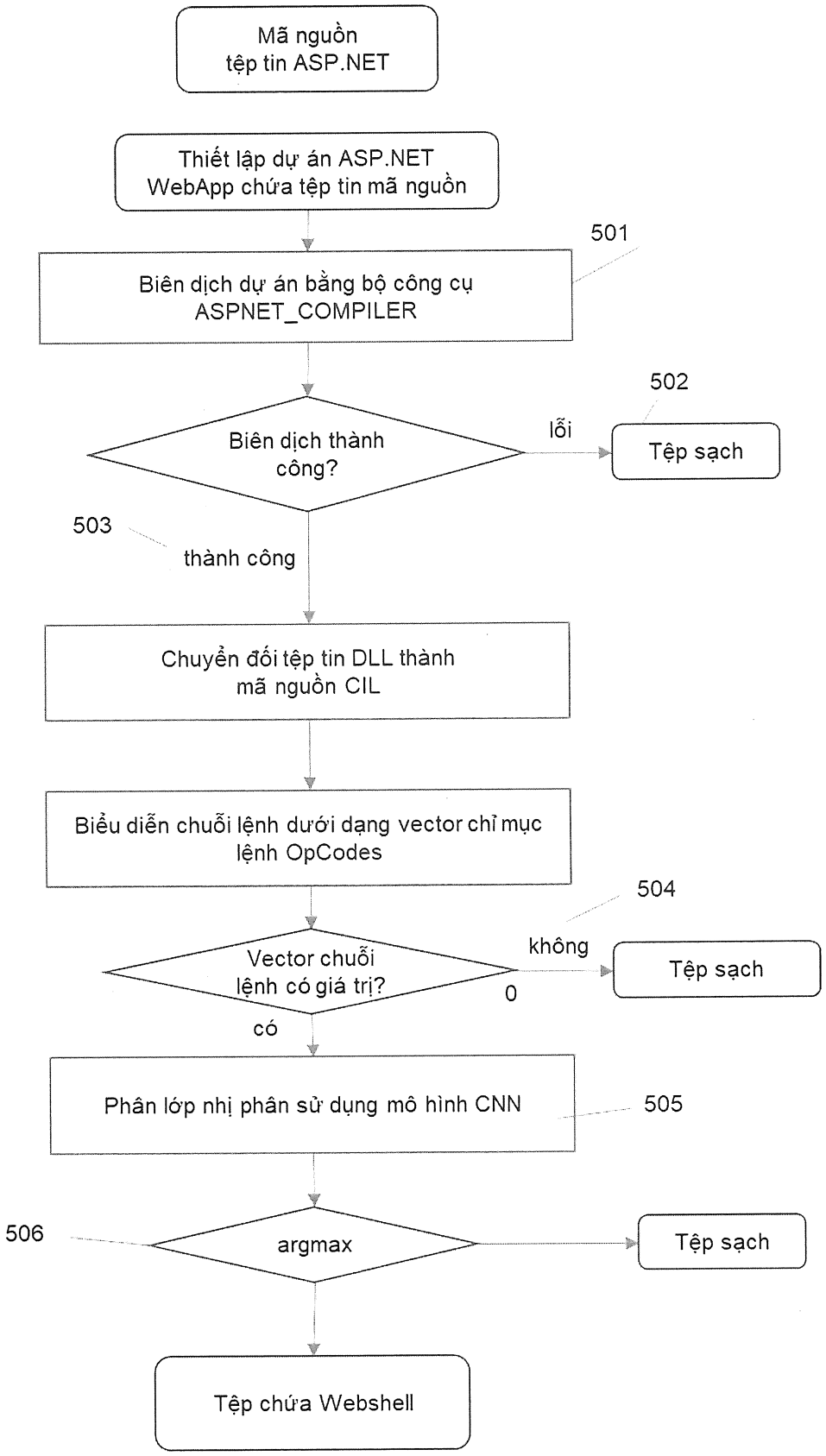
HÌNH 2B


.assembly extern mscorlib
{
.publickeytoken = (B7 7A 5C 56 19 34 E0
89)
.ver 4:0:0:0
}
.assembly HelloWorld
{
----lines omitted
}
.module HelloWorld.exe
.class private auto ansi beforefieldinit
HelloWorld.Program
extends [mscorlib]System.Object
{
.method private hidebysig static void
Main(string[] args) cil managed
{
.entrypoint
// Code size 13 (0xd)
.maxstack 8
IL_0000: nop
IL_0001: ldstr "Hello World"
IL_0006: call void
[mscorlib]System.Console::WriteLine(string)
IL_000b: nop
IL_000c: ret
} // end of method Program::Main
.method public hidebysig specialname
rtspecialname
instance void .ctor() cil
managed
{
.maxstack 8
IL_0000: ldarg.0
IL_0001: call instance void
[mscorlib]System.Object::.ctor()
IL_0006: ret
} // end of method Program::.ctor
} // end of class HelloWorld.Program
[0, 176, 162, 40, 176, 185, 0, 97, 40, 185]

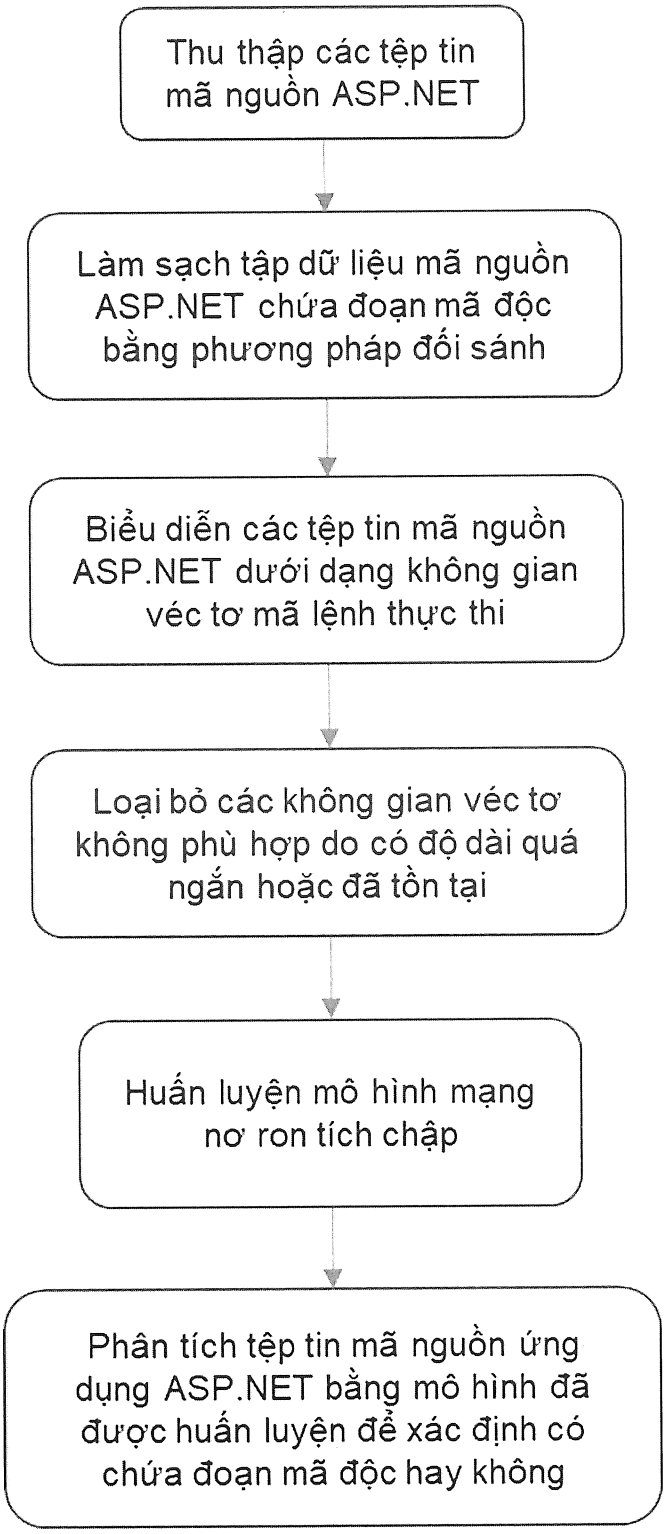
HÌNH 3


401
Đầu vào
402
Lớp tích chập
403
Lớp gộp
Lớp kết nối đầy đủ
Lớp hồi quy
Vector chuỗi lệnh
Độ dài tối đa 20.000
Kết hợp 3 lớp
Tệp tin sạch
Tệp tin chứa Webshell
Số lượng filter: 128
Kích thước filter: 3, 4, 5
Hàm kích hoạt phi tuyến: ReLU
Giá trị dropout: 0,5
Hàm kích hoạt softmax
Thuật toán tối ưu: adam
Hàm mất mát: cross entropy
Học đặc trưng
Phân lớp
404

HÌNH 4


Mã nguồn
tệp tin ASP.NET
Thiết lập dự án ASP.NET
WebApp chứa tệp tin mã nguồn
501
Biên dịch dự án bằng bộ công cụ
ASPNET_COMPILER
502
Biên dịch thành
công?
lỗi
Tệp sạch
503
thành công
Chuyển đổi tệp tin DLL thành
mã nguồn CIL
Biểu diễn chuỗi lệnh dưới dạng vector chỉ mục
lệnh OpCodes
504
Vector chuỗi
lệnh có giá trị?
không
0
Tệp sạch
có
Phân lớp nhị phân sử dụng mô hình CNN
505
506
argmax
Tệp sạch
Tệp chứa Webshell

HÌNH 5


Thu thập các tệp tin mã nguồn ASP.NET
Làm sạch tập dữ liệu mã nguồn ASP.NET chứa đoạn mã độc bằng phương pháp đối sánh
Biểu diễn các tệp tin mã nguồn ASP.NET dưới dạng không gian véc tơ mã lệnh thực thi
Loại bỏ các không gian véc tơ không phù hợp do có độ dài quá ngắn hoặc đã tồn tại
Huấn luyện mô hình mạng nơ ron tích chập
Phân tích tệp tin mã nguồn ứng dụng ASP.NET bằng mô hình đã được huấn luyện để xác định có chứa đoạn mã độc hay không

HÌNH 6