ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Lê Việt Hà

NGHIÊN CỨU MỘT SỐ PHƯƠNG PHÁP HỌC SÂU TRONG PHÁT HIỆN ĐOẠN MÃ ĐỘC

Chuyên ngành: Hệ thống thông tin
Mã: 9480104

TÓM TẮT LUẬN ÁN TIẾN SỸ
CHUYÊN NGÀNH HỆ THỐNG THÔNG TIN

Cán bộ hướng dẫn:
PGS.TS Nguyễn Ngọc Hóa
TS. Phùng Văn Ổn

Hà Nội - 2024

Công trình được hoàn thành tại: **Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội**.

Người hướng dẫn khoa học:

- Phó giáo sư, Tiến sĩ Nguyễn Ngọc Hóa

- Tiến sĩ Phùng Văn Ổn

Phản biện: .....

.....

.....

Phản biện: .....

.....

.....

Phản biện: .....

.....

.....

Luận án sẽ được bảo vệ trước Hội đồng cấp Đại học Quốc gia chấm luận án tiến sĩ họp tại .....

vào hồi .....giờ .....ngày .....tháng .....năm .....

Có thể tìm hiểu luận án tại:

- Thư viện quốc gia Việt Nam.
- Trung tâm Thông tin - Thư viện, Đại học Quốc gia Hà Nội.

Giới thiệu

Động lực nghiên cứu

Sự đổi mới của công nghệ phát triển web đã khiến các ứng dụng web ngày càng phổ biến và đang dần thay thế các ứng dụng gốc truyền thống vì chúng không phụ thuộc vào hệ điều hành. Cùng với đó, các vấn đề về bảo mật thông tin cho hệ thống web ngày càng trở nên quan trọng. Trong đó, tấn công tiêm mã độc hại (webshell) là loại tấn công ứng dụng web phổ biến nhất và cũng nguy hiểm nhất.

Theo phân tích từ Cloudflare, hơn hai phần ba webshell thể hiện một số dạng che giấu. Những tiến bộ trong các kỹ thuật phát hiện đã phải vật lộn để theo kịp khi những kẻ tấn công liên tục phát hành các công cụ webshell mới, được che giấu kỹ lưỡng để tránh các biện pháp phòng thủ. Sự tinh vi và đa dạng ngày càng tăng của các webshell, đặc biệt là những webshell được thiết kế để tránh các phương pháp phát hiện truyền thống, nhấn mạnh nhu cầu cấp thiết về các kỹ thuật tiên tiến để cải thiện khả năng phát hiện của chúng. Nhiều nghiên cứu chứng minh hiệu quả của việc sử dụng các thuật toán ML và DL để cải thiện khả năng phát hiện các biến thể mới của webshell. Tuy nhiên, các nghiên cứu này vẫn có một số hạn chế nhất định và vẫn còn nhiều dư địa để nghiên cứu, phát triển.

Thách thức trong nghiên cứu

Từ bối cảnh nghiên cứu cho thấy các thuật toán phát hiện webshell để có thể áp dụng trong thực tế vẫn còn tồn tại một số thách thức như sau:

1. **Sự đa dạng của ngôn ngữ phát triển webshell** thực sự là một thách thức lớn đối với các chuyên gia an ninh mạng trong việc triển khai các biện pháp phòng thủ. Mỗi ngôn ngữ lập trình có những tính năng và đặc trưng riêng biệt, do đó, làm thế nào để biểu diễn được các tệp mã nguồn cho các mô hình AI mã vẫn giữ được đầy đủ các đặc trưng của webshell là một bài toán.

2. **Webshell tân tiến** thường có được trang bị các kỹ thuật lẩn tránh tinh vi, chúng tận dụng các tính năng che giấu, mã hóa và đa hình để ẩn đi sự hiện diện của chúng và tránh bị phát hiện bằng các biện pháp bảo mật truyền thống. Do đó, việc phát hiện đòi hỏi các kỹ thuật tiên tiến hơn, chẳng hạn như giám sát

dựa trên hành vi, phát hiện bất thường và phân tích các tiến trình hoạt động trên bộ nhớ.

3. **Chất lượng của tập dữ liệu** là một trong những yếu tố then chốt trong việc phát triển các kỹ thuật phòng thủ webshell. Tuy nhiên webshell là dữ liệu nhạy cảm nên sẽ không có nhiều nguồn dữ liệu chính thức, đáng tin cậy sẵn sàng chia sẻ rộng rãi, đặc biệt là các loại webshell tân tiến.

4. **Độ chính xác và Tốc độ phát hiện** là hai tiêu chí cần thiết của mọi giải pháp an ninh mạng. Trong bài toán phát hiện webshell, thách thức lớn là xây dựng một giải pháp đồng thời đạt được cả hai tiêu chí về độ chính xác trong việc phát hiện webshell tiên tiến và tốc độ phát hiện đủ nhanh để giảm thiểu thiệt hại cho hệ thống.

Mục tiêu nghiên cứu

Mục tiêu chính của luận án là đề xuất các phương pháp phát hiện webshell sử dụng các mô hình học sâu nhằm cải thiện độ chính xác và hiệu xuất. Để đạt được mục tiêu chính của luận án, bốn mục tiêu cụ thể như sau:

- Mục tiêu 1: Tổng quan về webshell, các kỹ thuật tiên tiến nhất được tin tặc sử dụng để ẩn giấu các hành vi tấn công webshell của chúng. Nghiên cứu các kỹ thuật phát hiện webshell, phân tích ưu nhược điểm của từng phương pháp. Đánh giá kết quả nghiên cứu mới nhất hiện nay.

- Mục tiêu 2: Đề xuất một Khung kiến trúc dựa trên kỹ thuật phân tích mã nguồn được hỗ trợ bởi mô hình học sâu, đặt tên là ASAF, kết hợp các kỹ thuật dựa trên chữ ký với các thuật toán học sâu. Phương pháp kết hợp này cho phép phát hiện nhanh chóng và chính xác các loại webshell đã biết và chưa biết. Khung kiến trúc đề xuất là hướng dẫn phát triển các mô hình cụ thể phù hợp với nhiều ngôn ngữ lập trình khác nhau.

- Mục tiêu 3: Dựa trên khung kiến trúc đề xuất ở trên, phát triển hai hệ thống hoàn chỉnh để phát hiện các cuộc tấn công webshell cho từng ngôn ngữ lập trình PHP và ASP.NET. Các mô hình AI được tích hợp trong hệ thống phải được tối ưu hóa cho việc phát hiện webshell cụ thể để đảm bảo phát hiện hiệu quả với nguồn lực tính toán tối thiểu. Kết quả phát hiện của hệ thống phải được so sánh với các nghiên cứu khác.

- Mục tiêu 4: Đề xuất phương pháp phát hiện tốc độ cao các cuộc tấn công webshell sử dụng mô hình học sâu để thực hiện phân tích chuyên sâu các truy vấn HTTP vào hệ thống ứng dụng web, xác định hiệu quả các truy vấn tấn công sử dụng cả những loại webshell đã biết và chưa biết. Hệ thống phải có khả năng tích hợp hoàn chỉnh với hệ thống IDS/IPS để tự động chặn các địa chỉ nguồn đáng ngờ trong thời gian thực.

Đóng góp của nghiên cứu

Luận án có những đóng góp chính sau:

1. Đề xuất một khung kiến trúc rà quét mã nguồn để phát hiện webshell kết hợp các kỹ thuật so khớp mẫu với các thuật toán học sâu. Khung kiến trúc này này là cơ sở hướng dẫn để phát triển các mô hình cụ thể nhằm phát hiện webshell chính xác và hiệu quả đối với nhiều ngôn ngữ lập trình khác nhau. Với mỗi loại ngôn ngữ lập trình biên dịch và thông dịch, chúng tôi đã chọn PHP và ASP.NET là những ngôn ngữ phổ biến nhất của từng loại để xây dựng mô hình phát hiện webshell dựa trên ASAF. Chúng tôi đã tiến hành các thí nghiệm và so sánh mô hình trên với các nghiên cứu khác để chứng minh tính hiệu quả của ASAF. *Đóng góp này đã được thể hiện trong bốn công bố, bao gồm một bài báo trên tạp chí SCI-E/Scopus [LVH-J1], một bài báo trên tạp chí quốc tế [LVH-J3] (được E-SCI lập chỉ mục cho đến năm 2023), một bài báo trên tạp chí khoa học và công nghệ quốc gia về an ninh thông tin [LVH-J4] và một bài báo tại hội nghị WoS/Scopus [LVH-C1]. Phương pháp phát hiện mã độc trong mã nguồn ứng dụng web sử dụng PHP và ASP.NET cũng đã được đăng ký sáng chế tại Cục Sở hữu trí tuệ, Bộ Khoa học và Công nghệ [LVH-P1, LVH-P2]. Trong đó, phương pháp phát hiện webshell ASP.NET đã được cấp bằng sáng chế vào ngày 19 tháng 5 năm 2023.*

2. Đề xuất một mô hình học sâu để phân tích sâu lưu lượng HTTP đến máy chủ ứng dụng web nhằm phát hiện nhanh các truy vấn webshell. Để giải quyết vấn đề mất cân bằng dữ liệu cho các tập dữ liệu huấn luyện, chúng tôi cũng đề xuất một thuật toán để cải thiện chất lượng của các tập dữ liệu được sử dụng trong mô hình học sâu. Để chứng minh tính hiệu quả của nó, chúng tôi đã thử nghiệm và so sánh mô hình với các nghiên cứu khác trên cùng một tập dữ liệu CSE-CIC-IDS2018. Mô hình học sâu có thể tích hợp với hệ thống phát hiện và ngăn chặn xâm nhập để tự động thêm các IP nguồn tấn công vào danh sách đen và chủ động chặn các truy vấn URI đến webshell trên máy chủ web trước

khi chúng xảy ra. *Đóng góp này đã được thể hiện trong hai công bố: một bài báo trên tạp chí SCI-E/Scopus [LVH-J2], một bài báo tại hội nghị WoS/Scopus [LVH-C2].*

Cấu trúc của luận án

Ngoài phần mở đầu và kết luận, luận văn được tổ chức thành 3 chương chính với các nội dung chính sau:

- Chương 1 trình bày tổng quan về webshell và các phương pháp phát hiện tấn công webshell, ML/DL trong phát hiện webshell, tổng hợp tài liệu các công trình khoa học và tiêu chí đánh giá tính hiệu quả của mô hình ML/DL.
- Chương 2 đề xuất một khung kiến trúc rà quét mã nguồn kết hợp các kỹ thuật so khơp mẫu với các thuật toán học sâu. Phương pháp tiếp cận kết hợp này cho phép phát hiện nhanh chóng và chính xác các loại webshell đã biết và chưa biết. Khung kiến trúc là cơ sở hướng dẫn để phát triển các mô hình cụ thể phù hợp với nhiều ngôn ngữ lập trình khác nhau. Dựa trên khung kiến trúc được đề xuất ở trên, chúng tôi sẽ phát triển hai hệ thống toàn diện phù hợp để phát hiện các cuộc tấn công webshell bằng PHP và ASP.NET.
- Chương 3 đề xuất mô hình học sâu thực hiện phân tích sâu các truy vấn HTTP đến hệ thống ứng dụng web, nhằm phát hiện hiệu quả các truy vấn webshell đã biết và chưa biết. Chúng tôi thử nghiệm mô hình trên hai tập dữ liệu và so sánh kết quả với các nghiên cứu khác để chứng minh tính hiệu quả của nó. Chúng tôi cũng đã tích hợp mô hình vào hệ thống NetIDPS để tự động chặn các địa chỉ nguồn đáng ngờ theo thời gian thực, đảm bảo tính ứng dụng thực tế của nó.

1 Kiến thức chung

1.1 Khái niệm

1.1.1 Ứng dụng web

Máy chủ web sử dụng Giao thức truyền siêu văn bản (HTTP) cùng với các giao thức khác để nhận yêu cầu của người dùng thông qua trình duyệt web. Phía máy chủ sẽ cần một ngôn ngữ lập trình để xử lý các yêu cầu được gửi từ phía máy khách (trình duyệt). Trong quá trình biên dịch, toàn bộ chương trình được trình biên dịch

dịch sang mã máy trước khi thực thi. Trong thông dịch, chương trình được dịch từng dòng hoặc từng câu lệnh bởi một trình thông dịch trong quá trình thực thi.

1.1.2 Tấn công webshell

Webshell thường là một đoạn mã độc hại được kẻ tấn công chèn vào mã nguồn máy chủ web để cho phép truy cập và thực thi lệnh từ xa. Đoạn mã được viết bằng các ngôn ngữ lập trình phát triển web phổ biến (ví dụ: ASP, PHP, JSP).

Về cơ bản, một cuộc tấn công webshell được chia thành bốn giai đoạn: Tìm và khai thác các lỗ hổng bảo mật, cài đặt webshell, leo thang đặc quyền, phát động tấn công.

1.1.3 Phân loại webshell

Webshell có thể có nhiều cách phân loại khác nhau dựa trên đặc điểm, ngôn ngữ kịch bản, tính năng, v.v. Dưới đây là cách phân loại phổ biến nhất dựa trên ngôn ngữ lập trình: webshell PHP, webshell ASP/ASPX, webshell Shell Script và các webshell sử dụng các ngôn ngữ lập trình phía máy chủ khác.

Ngoài ra, còn có một phương pháp phân loại khác dựa trên cách webshell giao tiếp với máy tính điều khiển của hacker.

1.1.4 Kỹ thuật ẩn giấu webshell

Tin tặc sử dụng nhiều kỹ thuật khác nhau để tránh bị phát hiện và cho phép webshell vượt qua các biện pháp bảo vệ an ninh. Các chiến thuật trốn tránh này sẽ làm lẫn các hành vi của mã webshell với các kênh liên lạc và môi trường thực thi thông thường để tránh bị hệ thống bảo mật phát hiện. Một chiến thuật phổ biến là làm xáo trộn mã nguồn webshell thông qua các phương pháp như mã hóa, đa hình.

1.1.5 Đặc trưng của webshell

Webshell tương đối giống với các tệp tin lành tính, điều này dẫn đến khó khăn trong việc phân biệt chúng. Các nghiên cứu trước đây đã sử dụng ba loại siêu dữ liệu và năm bộ tính năng để phân biệt các webshell độc hại. Ba loại siêu dữ liệu riêng biệt thường được liên kết với webshell, mỗi loại cung cấp thông tin chi tiết riêng về đặc điểm và hành vi của chúng: mã nguồn, mã thực thi và truy vấn HTTP.

1.2 Phương pháp phát hiện

1.2.1 Phân tích mã nguồn

Để phát hiện webshell, phân tích mã nguồn và opcode liên quan đến việc kiểm tra mã mà không thực thi mã, tập trung vào việc xác định các mẫu, cấu trúc và bất thường chỉ ra sự hiện diện của webshell. Phân tích mã nguồn đòi hỏi phải xem xét kỹ lưỡng các câu lệnh lập trình dạng văn bản bao gồm webshell và tận dụng các tính năng cú pháp, ngữ nghĩa và thống kê để phân biệt giữa mã lành tính và mã độc hại. Phân tích này có thể liên quan đến việc xác định các từ khóa, chức năng hoặc mẫu cụ thể thường liên quan đến webshell, chẳng hạn như thực thi lệnh hệ thống, thao tác tệp hoặc giao tiếp mạng.

1.2.2 Phân tích luồng dữ liệu HTTP

Phân tích lưu lượng HTTP để phát hiện webshell liên quan đến việc xem xét kỹ lưỡng giao tiếp mạng giữa máy khách và máy chủ web để xác định các mẫu, bất thường hoặc chữ ký chỉ ra hoạt động của webshell. Phân tích này thường liên quan đến việc nắm bắt và kiểm tra các yêu cầu và phản hồi HTTP được trao đổi giữa máy khách và máy chủ, tập trung vào các thuộc tính khác nhau như phương thức yêu cầu, URI, tiêu đề, tham số, tải trọng và mã phản hồi.

1.3 Nghiên cứu liên quan

Thống kê nghiên cứu liên quan đến ***Webshell Detection*** từ các nguồn uy tín [1] cho thấy rằng có 41 nghiên cứu trước đó, 17 trong số đó họ (42%) áp dụng học máy, 12 nghiên cứu (29%) sử dụng công nghệ học sâu và 12 nghiên cứu (29%) đề xuất các loại giải pháp khác.

1.4 Hướng nghiên cứu của luận án

Luận án đề xuất xây dựng hai mô hình theo hai hướng tiếp cận: phân tích dựa trên mạng và phân tích mã nguồn, để tận dụng ưu điểm của từng hướng tiếp cận, cụ thể như sau:

- Đề xuất phương pháp phân tích mã nguồn ứng dụng web để phát hiện webshell. Phương pháp này kết hợp những ưu điểm của hai phương pháp dựa trên so khớp

[1] *IEEE Xplore, *ACM Digital Library, *SpringerLink, *Wiley Online Library, *ScienceDirect

mẫu và dựa trên CNN, có khả năng phát hiện webshell sáng tạo với độ chính xác rất cao. Vì mỗi ngôn ngữ lập trình đều có những đặc điểm riêng, nên trong phạm vi nghiên cứu này, chúng tôi sẽ tập trung vào hai ngôn ngữ lập trình phía máy chủ phổ biến nhất hiện nay: PHP được biên dịch và ASP.NET được biên dịch. Ngoài ra, phương pháp này cũng đề xuất các thuật toán biểu diễn các tệp nguồn dưới dạng chuỗi vectơ, phản ánh đầy đủ các đặc điểm của webshell và được sử dụng làm đầu vào cho các mô hình học sâu.

- Đề xuất một mô hình DNN để phân tích các truy vấn HTTP chuyên sâu vào hệ thống ứng dụng web để phát hiện các truy vấn chỉ ra webshell trên máy chủ ứng dụng web. Mô hình có khả năng tích hợp vào các hệ thống IDS/IPS để tự động thêm các địa chỉ nguồn đáng ngờ vào danh sách đen và chặn URI của webshell trên máy chủ web.

2 Phát hiện webshell bằng phân tích mã nguồn

2.1 Phát biểu bài toán

Các công trình nghiên cứu điển hình trong phân tích mã nguồn ứng dụng để phát hiện webshell đã chỉ ra rằng các phương pháp truyền thống và phương pháp sử dụng ML/DL đều có những ưu điểm và nhược điểm khác nhau, nhưng hiện tại chưa có nhiều nghiên cứu sử dụng phương pháp kết hợp để tận dụng ưu điểm của hai phương pháp này. Luận án xác định hướng nghiên cứu sẽ tập trung vào việc đề xuất một khung kiến trúc kết hợp các kỹ thuật phát hiện dựa trên chữ ký và các kỹ thuật phát hiện dựa trên thuật toán AI. Khung kiến trúc sẽ cho phép phát hiện nhanh các webshell đã biết và khả năng phát hiện chính xác các webshell chưa biết. Và trong chương này, nghiên cứu lựa chọn hai ngôn ngữ lập trình phía máy chủ phổ biến nhất hiện nay là PHP và ASP.NET để chứng minh tính đúng đắn và khả thi của kiến trúc. **Ba mục tiêu cụ thể như sau:**

- Đề xuất một Khung kiến trúc phân tích mã nguồn sử dụng mô hình học sâu, đặt tên là ASAF, kết hợp hai kỹ thuật dựa trên so khớp mẫu và mô hình học sâu, để cho phép phát hiện nhanh và chính xác các loại webshell, bao gồm các loại đã biết và chưa biết. Khung kiến trúc sẽ là định hướng để xây dựng từng mô hình cụ thể áp dụng cho từng loại ngôn ngữ lập trình khác nhau.

- Dựa trên ASAF đề xuất một mô hình phát hiện webshell PHP là ngôn ngữ thông dịch. Mô hình này bao gồm một thuật toán chuyển đổi tệp nguồn PHP thành một vectơ phẳng chứa tất cả các tính năng webshell. Mô hình này cũng bao gồm một mô hình ML/DL với các tham số được điều chỉnh để phù hợp nhất với vấn đề phát hiện webshell PHP, để đảm bảo phát hiện hiệu quả mà không yêu cầu quá nhiều tài nguyên tính toán. Đánh giá hiệu quả của mô hình được đề xuất dựa trên các tiêu chí đo lường và so sánh với các nghiên cứu có liên quan.

- Dựa trên ASAF đề xuất một mô hình phát hiện webshell ASP.NET là ngôn ngữ biên dịch. Mô hình này bao gồm một thuật toán chuyển đổi tệp nguồn ASP.NET thành một vectơ phẳng chứa tất cả các tính năng webshell. Mô hình cũng bao gồm một mô hình ML/DL với các tham số được điều chỉnh để phù hợp nhất với vấn đề phát hiện webshell ASP.NET để đảm bảo phát hiện hiệu quả mà không yêu cầu quá nhiều tài nguyên tính toán. Đánh giá hiệu quả của mô hình đề xuất dựa trên tiêu chí đo lường và so sánh với các nghiên cứu có liên quan.

2.2 Đề xuất khung kiến trúc Phân tích mã nguồn ASAF

Như đã phân tích ở trên, sự tinh vi và phổ biến ngày càng tăng của webshell dẫn đến nhu cầu về một khung kiến trúc phân tích mã nguồn chung có thể áp dụng cho nhiều ngôn ngữ lập trình khác nhau và có khả năng phát hiện nhanh với tỷ lệ dương tính giả thấp đối với các loại webshell đã biết. Đồng thời, đó là khả năng phát hiện với độ chính xác cao các loại webshell mới. Dựa trên các kết quả nghiên cứu trước đây, nghiên cứu này đề xuất một khung kiến trúc phân tích mã nguồn được hỗ trợ bởi mô hình học sâu, có tên là ASAF, kết hợp các quy tắc Yara để phát hiện webshell đã biết với mô hình Mạng nơ-ron tích chập (CNN) để phát hiện các biến thể webshell mới, tinh vi. Bằng cách tận dụng thế mạnh của cả phương pháp dựa trên so khớp mẫu và dựa trên mô hình học sâu, khung kiến trúc này cung cấp khả năng phát hiện webshell toàn diện và hiệu quả. Cấu trúc của khung kiến trúc sẽ bao gồm năm mô-đun/thành phần:

- **Thành phần YARA:** Kiến trúc của mô-đun YARA trong ASAF xoay quanh hệ thống YARA. Chức năng chính của hệ thống này là phát hiện các webshell đã biết dựa trên các mẫu được xác định trước. YARA được tạo thành từ hai thành phần: cơ chế so khớp mẫu và cơ sở dữ liệu tập luật Yara.

- **Thành phần Vectơ hoá:** Mục đích của mô-đun này khung kiến trúc phát hiện webshell là nâng cao độ chính xác và độ sâu của phân tích mã tĩnh bằng cách chuyển đổi mã nguồn web thành các chuỗi opcode tương ứng.

- **Thu thập và làm sạch dữ liệu:** Trong ASAF, tập dữ liệu đóng một vai trò quan trọng trong việc đào tạo, kiểm thử mô hình Mạng CNN. Tập dữ liệu phải bao gồm cả tệp nguồn ứng dụng web lành tính và độc hại để đào tạo CNN một cách hiệu quả. Để thu thập tập dữ liệu đòi hỏi phải thu thập từ nhiều nguồn khác nhau. Kho lưu trữ mã nguồn chia sẻ trực tuyến, hoặc từ các nguồn dữ liệu webshell thông qua các diễn đàn bảo mật và kho lưu trữ nguồn mở, chẳng hạn như: Exploit Database, Hack Forums, GitHub Repositories, ... Cuối cùng, các mạng cá nhân và chuyên nghiệp cung cấp quyền truy cập vào các loại webshell mới chưa được chia sẻ rộng rãi.

- **Kiến trúc mô hình CNN:** Trong ASAF, thiết kế kiến trúc mô hình CNN đóng một vai trò quan trọng. Kiến trúc của mô hình CNN bao gồm các lớp, mối quan hệ giữa các lớp và cả các siêu tham số có giá trị được xác định trước khi quá trình huấn luyện bắt đầu. Thông thường, với mỗi bài toán cụ thể sẽ có những kiến trúc nhất định thể hiện được những ưu điểm vượt trội. Tuy nhiên, nó cần phải trải qua một quá trình gọi là chuyển đổi siêu tham số (hyperparameter tuning) để đạt được hiệu quả, hiệu suất và tốc độ tốt nhất. Việc lựa chọn siêu tham số tiêu tốn khá nhiều thời gian và tài nguyên, vì vậy không phải tất cả các siêu tham số sẽ được tinh chỉnh khi chúng ta đã chắc chắn những siêu tham số đó là tối ưu. Do đó, ở bước này, chúng tôi phác thảo kiến trúc mô hình CNN bằng cách sử dụng cùng cấu trúc và siêu tham số tối ưu như trong Hình. **??**. Các siêu tham số khác sẽ được chọn sau khi chúng ta thực hiện bước tối ưu.

- **Tối ưu hoá Hyperparameter:** Kiến trúc mô hình CNN ở trên chỉ là một khung kiến trúc cơ bản được thiết kế phù hợp nhất với bài toán phát hiện webshell, tuy nhiên ngôn ngữ lập trình có ảnh hưởng rất lớn đến đặc điểm của từng loại webshell. Vì vậy, việc thực hiện điều chỉnh siêu tham số để xây dựng mô hình CNN cho từng loại webshell được viết bằng các ngôn ngữ lập trình khác nhau là rất quan trọng.

- **Luồng thực thi ASAF:** Quá trình này bắt đầu với các tệp nguồn ứng dụng web trải qua quá trình phân tích so khớp mẫu bằng cách sử dụng các thành phần YARA, bao gồm cơ chế so khớp mẫu và cơ sở dữ liệu quy tắc YARA. Nếu tìm thấy kết quả khớp, tệp sẽ ngay lập tức được gắn cờ webshell. Nếu không phát hiện thấy kết quả trùng khớp nào, tệp được coi là lành tính và chuyển sang

giai đoạn tiếp theo, bao gồm phân tích sâu hơn bằng cách sử dụng các mô-đun tạo và vector hóa mã opcode. Các mô-đun này chuyển đổi mã nguồn thành các chuỗi opcode.

Các chuỗi opcode sau đó được vector hóa và đưa vào Mạng CNN để phân tích sâu. Tập dữ liệu đã được làm sạch đóng một vai trò quan trọng trong việc đào tạo, xác thực và thử nghiệm mô hình CNN. Mô hình này cũng đã được tinh chỉnh thông qua việc điều chỉnh siêu tham số, dự đoán xem tệp tin đó là webshell hay lành tính. Nếu CNN phát hiện thấy webshell, dự đoán này sẽ được chuyển tiếp đến các chuyên gia an ninh mạng để xác minh và cập nhật luật YARA, đảm bảo rằng các mẫu mới được tích hợp vào cơ sở dữ liệu luật YARA. Khung kiến trúc này cũng cho phép tự động cập nhật các quy tắc YARA từ cơ sở dữ liệu IoC được chia sẻ. Ngược lại, nếu CNN dự đoán mã là lành tính thì nó được xác nhận là an toàn. Cách tiếp cận hai lớp này, tận dụng cả quy tắc YARA cho các mối đe dọa đã biết và mô hình CNN cho các mối đe dọa chưa xác định, đảm bảo phát hiện mạnh mẽ và linh hoạt các webshell, tăng cường tính bảo mật của ứng dụng web.

2.3 Phát hiện webshell PHP

2.3.1 YARA

Bộ dữ liệu luật YARA được sử dụng trong nghiên cứu này được thu thập từ nhiều nguồn và trong một thời gian dài làm việc trong lĩnh vực bảo mật thông tin. Bộ dữ liệu chứa tổng cộng *699 quy tắc*, cho phép phát hiện nhiều webshell PHP, JSP, ASP, ASP.NET, Python phổ biến hiện nay. Bộ quy tắc này sẽ tiếp tục được cập nhật thường xuyên để nâng cao khả năng phát hiện các mẫu webshell mới.

2.3.2 Vectơ hoá Opcode

VLD, viết tắt của Vulcan Logic Disassembler, là một tiện ích mở rộng của ngôn ngữ PHP được thiết kế để phân tách mã PHP đã biên dịch, cung cấp bản dịch chi tiết về mã hoạt động bên trong của nó.

2.3.3 Thu thập và làm sạch

Sau khi xóa các tệp trùng lặp và không liên quan, tổng số tệp lành tính là **7275** và tổng số tệp webshell là **4087**.

2.3.4 Hyperparameter Tuninng cho mô hình CNN

Mô hình CNN phát hiện webshell PHP được xây dựng trên nền tảng mô hình CNN của ASAF, các tham số chi tiết của mô hình được lựa chọn thông qua quá trình điều chỉnh siêu tham số. Tôi chọn sử dụng kỹ thuật *tìm kiếm lưới* để lựa chọn trên một tập hợp con được chỉ định trong không gian siêu tham số. Có sáu siêu tham số có thể được tối ưu và chúng nhận hai loại giá trị là phạm vi và lựa chọn. Đặc biệt đối với *kích thước bộ lọc*, do mô hình sử dụng ba lớp được hợp nhất với nhau trong lớp tích chập nên mỗi lớp nhận được một kích thước bộ lọc khác nhau, do đó giá trị của nó sẽ là một vectơ 3 chiều có dạng $[x, x+1, x+2]$.

2.3.5 Thực nghiệm

Phần thử nghiệm được thực hiện với 3 kịch bản cùng với tập dữ liệu thử nghiệm được xây dựng trong 2.3.3.

- S1: Đánh giá hiệu quả phát hiện webshell PHP của thành phần YARA trong PHP-ASAF.
- S2: Đánh giá hiệu quả phát hiện webshell PHP của mô hình CNN trong PHP-ASAF..
- S3: Đánh giá hiệu quả phát hiện webshell PHP của PHP-ASAF.

2.3.6 So sánh

Để chứng minh hiệu quả của PHP-ASAF, chúng tôi so sánh kết quả của mình với các phương pháp khác. Do hạn chế trong việc chia sẻ bộ dữ liệu thử nghiệm của các nghiên cứu khác và hiện chưa có bộ dữ liệu công khai nào được sử dụng rộng rãi nên chúng tôi sử dụng cùng bộ dữ liệu tại 2.3.3 để so sánh với các phương pháp khác. Bảng sau đây cho thấy sự so sánh PHP-ASAF của chúng tôi với các phương pháp khác:

Để chứng minh hiệu quả của ASAF, chúng tôi so sánh với các phương pháp khác. Chúng tôi đã chọn hai phương pháp tiếp cận không sử dụng AI [1, 7] và hai phương pháp tiếp cận sử dụng ML/DL [2, 6]. Do những hạn chế trong việc chia sẻ mã nguồn, chúng tôi đã mô phỏng mô hình RF-GBDT và Word2Vec+CNN như tác giả mô tả, hiện là một trong những mô hình có kết quả phát hiện tốt nhất, để đánh giá trên tập dữ liệu của chúng tôi đã đề cập ở Phần 2.3.3. Kết quả thực tế của các mô hình mô phỏng RF-GBDT và Word2Vec+CNN đạt được không cao như đã công bố. Kết quả so sánh trong Bảng 2.1 cho thấy mô hình ASAF đạt được kết quả tốt nhất về cả độ

Bảng 2.1: So sánh với kết quả của các nghiên cứu khác trên tập dữ liệu của chúng tôi (%)

Method	Venue	Accuracy	F1-Score
Simulated Word2Vec+CNN[6]	ICNCC, 2017	98.42	97.80
Simulated RF-GBDT[2]	DSC, 2018	98.59	98.05
GuruWS[7]	TCCI, 2019	85.56	92.00
php-malware-finder[1]	NBS, 2022	94.23	96.46
ASAF (our)	VCRIS, 2024	**98.9**	**98.48**

chính xác là 98,9% và F1-Score là 98,48% khi so sánh với các phương pháp khác trên tập dữ liệu của chúng tôi.

2.4 Mô hình phát hiện webshell ASP.NET

Theo W3Techs, ASP.NET là ngôn ngữ lập trình phía máy chủ được sử dụng nhiều thứ hai trên thế giới sau PHP. Điều này có nghĩa là số lượng các cuộc tấn công webshell vào hệ thống web sử dụng ASP.NET cũng rất lớn. Không giống như PHP là ngôn ngữ thông dịch, ASP.NET là ngôn ngữ được biên dịch nên cơ chế tạo opcode từ mã nguồn của ứng dụng web là hoàn toàn khác. Việc xây dựng giải pháp phát hiện tấn công webshell ASP.NET dựa trên ASAF, cụ thể là ASP.NET-ASAF, sẽ thể hiện rõ hơn khả năng triển khai giải pháp phát hiện tấn công webshell hỗ trợ tất cả các ngôn ngữ lập trình trang web máy chủ.

2.4.1 Yara

Mô-đun yara trong giải pháp phát hiện tấn công webshell ASP.NET được sử dụng kết hợp với bộ quy tắc trong phần 2.3.1. Bộ quy tắc chứa tổng cộng 699 mẫu webshell. Chúng tôi sẽ thường xuyên cập nhật bộ quy tắc này để cải thiện khả năng phát hiện webshell mới.

2.4.2 Thu thập và làm sạch dữ liệu

Tổng số tệp nguồn ASP.NET chúng tôi thu thập được là 2.064 webshell và 3.347 tệp lành tính. Để huấn luyện mô hình và kiểm tra tính hiệu quả của phương pháp đề xuất, tập dữ liệu sẽ được chia thành hai phần với tỷ lệ 8:2 tương ứng với tập dữ liệu huấn luyện và tập dữ liệu kiểm tra. Bảng sau đây hiển thị các tập dữ liệu cuối cùng của chúng tôi để đào tạo và thử nghiệm.

2.4.3 Vectơ hoá Opcode

Mã nguồn của ứng dụng ASP.NET có thể được biểu diễn dưới dạng MSIL, mã này có thể giải quyết các vấn đề liên quan đến việc mã hóa của webshell. Không giống như PHP thông dịch mỗi khi một trang web được yêu cầu, ASP.NET biên dịch các trang web động thành các tệp DLL mà máy chủ có thể thực thi nhanh chóng và hiệu quả. Sau đó, các file DLL sẽ tiếp tục được chuyển đổi sang Opcode. Có hai công cụ thường được sử dụng: ILDasm (IL Disassembler) và Mono.Cecil. Chúng tôi chọn sử dụng Mono.Cecil, đây là một thư viện hỗ trợ khả năng đọc, thao tác và viết các tập hợp .NET.

2.4.4 Hyperparameter tunning CNN Model

Tương tự như quá trình lựa chọn tham số cho mô hình CNN của PHP-ASAF, chúng tôi cũng thực hiện quá trình này để chọn tham số tối ưu cho mô hình ASP.NET-ASAF. Có sáu siêu tham số có thể được điều chỉnh và chúng nhận hai loại giá trị là phạm vi và lựa chọn.

2.4.5 Thực nghiệm

Để đánh giá tính hiệu quả của ASP.NET-ASAF, chúng tôi đã tiến hành thử nghiệm với ba kịch bản sau:

- S1: Đánh giá hiệu quả phát hiện webshell ASP.NET của thành phần YARA trong ASP.NET-ASAF.
- S2: Đánh giá hiệu quả phát hiện webshell ASP.NET của mô hình CNN trong ASP.NET-ASAF.
- S3: Đánh giá hiệu quả phát hiện webshell ASP.NET của ASP.NET-ASAF.

S3: Kết quả thực nghiệm của ASP.NET-ASAF

Kết hợp ưu điểm của hai phương pháp YARA và mô hình học sâu CNN, ASP.NET-ASAF sẽ giải quyết vấn đề phát hiện hiệu quả các cuộc tấn công webshell, bao gồm cả các mẫu chưa xác định. Thử nghiệm trên tập dữ liệu trong **??** cho kết quả về ma trận nhầm lẫn và các độ đo chính trong Bảng 2.2 và Bảng 2.3. Kết quả cho thấy điểm F1 tăng từ 97,95% lên 98,07%, độ chính xác từ 98,43% lên 98,52% khi so sánh với kết quả dự đoán của CNN. Với ưu điểm là có thể phát hiện các webshell đã biết với độ chính xác rất cao, YARA sẽ giảm thiểu số lượng webshell bị CNN phân loại nhầm là

lành tính. Khi kết hợp YARA với mô hình CNN, 1 webshell bị phân loại sai sẽ được sửa. Sau đó, FNR giảm từ 1,69% xuống 1,49%.

Bảng 2.2: Confusion matrix of webshell detection by using ASP.NET-ASAF

	Real Webshell	**Real Benign**
Predicted Webshell	407	10
Predicted Benign	6	659

Bảng 2.3: Key metrics of of ASP.NET webshell detection by using CNN (%)

Measure	**Value (%)**
Accuracy	98.52
Precision	97.60
Recall	98.55
Specificity	98.51
F1-Score	98.07
False Positive Rate	1.49
False Negative Rate	1.45

Hiện nay chưa có nhiều nghiên cứu có khả năng phát hiện các mẫu webshell ASP.NET bằng kỹ thuật phân tích tĩnh và số lượng nghiên cứu sẵn sàng chia sẻ mã nguồn càng hạn chế hơn. Vì vậy, việc so sánh kết quả với nhiều nghiên cứu để đảm bảo tính khách quan là tương đối khó khăn.

2.5 Tổng kết Chương 2

Chương 2 của luận án đề xuất một Khung kiến trúc phân tích mã nguồn được hỗ trợ bởi mô hình học sâu, có tên là ASAF, để phát hiện hiệu quả các cuộc tấn công chèn mã độc vào mã nguồn ứng dụng web bằng cách sử dụng các webshell đã biết và chưa biết.

Luận án hướng đến việc kết hợp các ưu điểm của hai kỹ thuật phát hiện phổ biến hiện nay, kỹ thuật khớp mẫu sử dụng Yara để phát hiện hiệu quả các webshell đã biết và mô hình học sâu CNN để phát hiện các webshell mới. Khung kiến trúc bao gồm tổng cộng năm thành phần được liên kết với nhau thông qua quy trình làm việc ASAF. Khung kiến trúc này cho phép chúng tôi xây dựng từng hệ thống cụ thể để phát hiện hiệu quả các cuộc tấn công webshell được phát triển bằng các ngôn ngữ khác nhau.

Để chứng minh tính khả thi của khung kiến trúc, đối với từng loại ngôn ngữ được biên dịch và thông dịch, nghiên cứu đã thử nghiệm xây dựng các hệ thống để phát hiện hai ngôn ngữ lập trình phía máy chủ phổ biến nhất hiện nay, PHP tại 2.3 và ASP.NET tại 2.4. Kết quả thử nghiệm được so sánh với một số kết quả nghiên cứu khác để chứng minh tính hiệu quả.

Kết quả nghiên cứu trong chương này đã được trình bày trong bốn công bố, bao gồm một bài báo trên tạp chí SCI-E/Scopus [LVH-J1], một bài báo trên tạp chí quốc tế [LVH-J3] (được E-SCI lập chỉ mục đến năm 2023), một bài báo trên tạp chí khoa học công nghệ quốc gia về an ninh thông tin [LVH-J4] và một bài báo tại hội nghị WoS/Scopus [LVH-C1]. Các phương pháp phát hiện mã độc trong mã nguồn ứng dụng web sử dụng PHP và ASP.NET cũng đã được đăng ký sáng chế tại Cục Sở hữu trí tuệ, Bộ Khoa học và Công nghệ [LVH-P1, LVH-P2]. Đặc biệt, phương pháp phát hiện webshell ASP.NET đã được cấp.

3 Phát hiện webshell bằng phân tích luồng dữ liệu HTTP

3.1 Phát biểu bài toán

Hệ thống phát hiện/phòng ngừa xâm nhập mạng (NetIDPS) đóng vai trò then chốt trong việc xác định và giảm thiểu các cuộc tấn công webshell, gây ra mối đe dọa đáng kể đối với bảo mật máy chủ web bằng cách cung cấp quyền truy cập từ xa trái phép cho kẻ tấn công. Cách tiếp cận này có hiệu quả trong việc xác định các webshell đã được ghi chép trước đó nhưng yêu cầu cập nhật thường xuyên cơ sở dữ liệu chữ ký để duy trì sự liên quan trước các mối đe dọa mới nổi. Nhờ sự phát triển của các thuật toán ML/DL cho phép phân tích sâu lưu lượng mạng để phát hiện nhanh các bất thường, nó đã thúc đẩy nhiều nghiên cứu. Tuy nhiên, kết quả nghiên cứu trước đây đã chỉ ra rằng có một số thách thức lớn cần được giải quyết. Luận án xác định hướng nghiên cứu sẽ tập trung vào việc đề xuất một giải pháp toàn diện kết hợp các kỹ thuật phát hiện dựa trên chữ ký với mô hình mạng nơ-ron sâu để phát hiện và ngăn chặn hiệu quả các loại tấn công webshell khác nhau theo thời gian thực.

Ba mục tiêu cụ thể như sau:

- Đề xuất một mô hình mạng nơ-ron sâu (DNN) cho phép phân tích chuyên sâu các lưu lượng mạng được trao đổi với máy chủ web để phát hiện các dấu hiệu tấn công webshell theo thời gian thực. Mô hình DNN sẽ trải qua quá trình điều

chỉnh siêu tham số để tối ưu hóa hiệu suất và độ chính xác.

- Đề xuất một thuật toán hàm mất mát được cải tiến để giải quyết vấn đề mất cân bằng trong tập dữ liệu.
- Tích hợp mô hình DNN với giải pháp NetIDPS để cho phép tự động thêm IP nguồn tấn công vào danh sách đen và chủ động chặn các truy vấn URI tới webshell trên máy chủ web.

3.2 Kiến trúc đề xuất

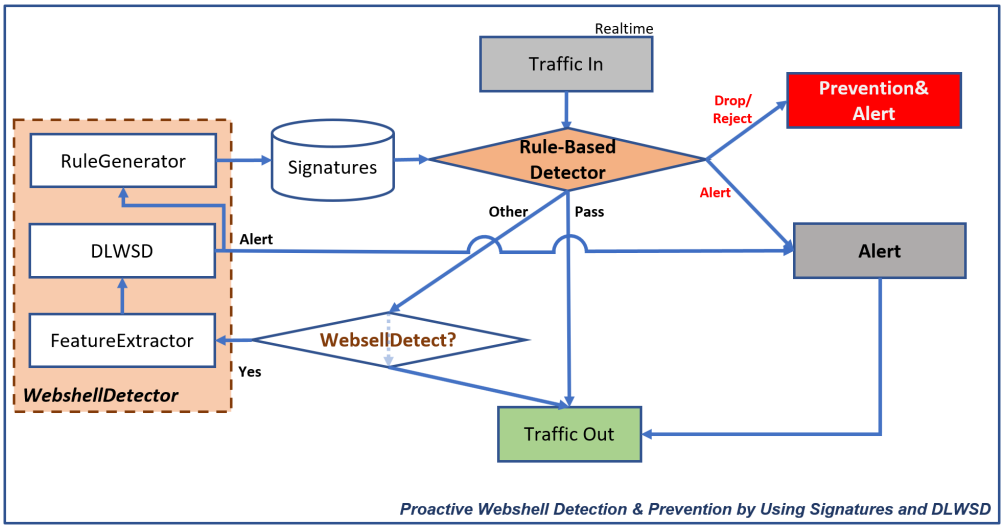
Phương pháp phát hiện xâm nhập kết hợp so khớp quy tắc và mô hình học sâu được minh họa trong Hình 3.1a và Hình. 3.1b. Trong phương pháp này, khi hệ thống NetIDPS được triển khai ở chế độ nội tuyến, lưu lượng mạng, theo cả hướng nhận và truyền, sẽ được hệ thống NetIDPS ghi lại. Sau khi thực hiện bước giải mã sẽ tiến hành quá trình phát hiện xâm nhập dựa trên 2 máy dò. Trình phát hiện đầu tiên sử dụng bộ quy tắc để phát hiện xâm nhập. Bộ quy tắc này được lưu trữ trong các tệp có cùng định dạng quy tắc với quy tắc của Suricata. Mỗi quy tắc sẽ có một mẫu hoặc một chữ ký duy nhất cho phép xác định lưu lượng mạng đang bị tấn công. Dựa trên bộ quy tắc, ở chế độ nội tuyến, mỗi gói mạng sẽ được NetIDPS kiểm tra và cấp phép bằng bốn hành động: (i) thả gói; (ii) từ chối gói (loại bỏ gói và thông báo cho nguồn đã gửi gói); (iii) vượt qua và cảnh báo; (iv) và vượt qua mà không báo trước.

3.3 Mô hình học sâu phát hiện sâm nhập

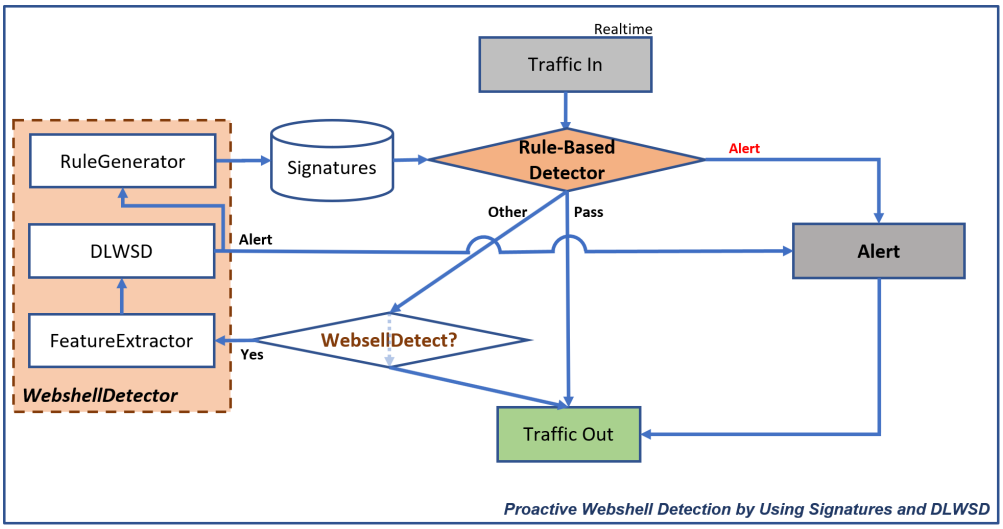
Trong phương pháp kết hợp mà chúng tôi đề xuất, chúng tôi chọn sử dụng mô hình học sâu bằng DNN. Dựa trên kết quả thử nghiệm của chúng tôi, việc phát hiện xâm nhập mạng bằng mô hình học sâu DNN sử dụng kỹ thuật *tabular_learner* của khung phát triển fastAI cho phép đạt độ chính xác tốt nhất, giảm thiểu tỷ lệ phát hiện sai khi so sánh với các mô hình deep learning khác cũng như các framework phát triển khác như Keras, TensorFlow, Theano, v.v. [5].

3.3.1 Webshell Detection and Prevention

Từ mô hình học sâu DLWSD được huấn luyện và lưu trữ, chúng tôi đã xây dựng *DeepInspector*, triển khai một dịch vụ chạy ở chế độ kernel (daemon), để nhận các luồng lưu lượng mạng được lưu dưới dạng tệp PCAP bởi NetIDPS. DeepInspector có vai trò giám sát các luồng lấy mẫu được tạo thường xuyên *inspection_information*



(a) IPS mode



(b) IDS mode

Hình 3.1: Proactive Webshell Detection Method based Signatures and DNN model

trong *inspection_duration* bởi NetIDPS để phân tích và phát hiện các cuộc tấn công webshell. Theo đó, khi hệ thống NetIDPS được cấu hình để kích hoạt chế độ phát hiện xâm nhập kết hợp cả luật và machine learning, các luồng dữ liệu mạng sẽ được nhận và gửi đến phần mềm hệ thống DeepInspector theo cơ chế giao tiếp đa tiến trình sử dụng socket Unix.

3.3.2 Xử lý dữ liệu mất cân bằng

Trên thực tế, khi triển khai hệ thống IDPS, các luồng lưu lượng bị xâm phạm thường rất nhỏ so với các luồng mạng "lành tính". Thống kê cũng cho thấy rằng hầu hết các bộ dữ liệu tiêu chuẩn liên quan đến xâm nhập mạng đều có số lượng luồng bị tấn công thấp hơn nhiều so với các luồng lành tính. Sự mất cân bằng mẫu giữa các

lớp gây ảnh hưởng nghiêm trọng đến cả quá trình huấn luyện và quá trình phân loại trong các mô hình machine learning, đặc biệt là các mô hình deep learning. Đối với bài toán phân loại nhị phân, lớp có mẫu cao hơn được gọi là "lớp đa số"; nhóm còn lại được gọi là "tầng lớp thiểu số".

Từ đó cần có các phương pháp xử lý tình trạng mất cân bằng dữ liệu giữa các lớp. Một trong những phương pháp xử lý sự mất cân bằng dữ liệu giữa các lớp thường được sử dụng trong các mô hình học máy là gán trọng số cho các lớp. Sự mất cân bằng mẫu được sử dụng để tạo trọng số cho từng lớp trong quá trình đào tạo. Các trọng số này được sử dụng bởi hàm mất entropy chéo để đảm bảo rằng lớp đa số được giảm trọng số tương ứng.

3.4 Thực nghiệm và đánh giá

3.4.1 Môi trường thực nghiệm

Để xác thực phương pháp được đề xuất, chúng tôi đã triển khai NetIDPS trong một thiết bị máy chủ có cấu hình 2 x Intel Xeon-Platinum 8160 (2.1GHz/24-core); RAM 384GB DDR4-2666; Bộ tăng tốc tính toán NVIDIA Tesla T4 16GB;SmartNIC Napatech NT40E3-4-PTP (10Gb 4 cổng SFP+, bộ đệm RAM DDR3 4 GB). Suricata v6.0.3 được sử dụng làm lõi của NetIDPS. Tuy nhiên, nhiều thành phần quan trọng cũng đã được thêm vào Suricata để có thể kiểm soát luồng giao thông và thực hiện chiến lược kiểm tra sâu.

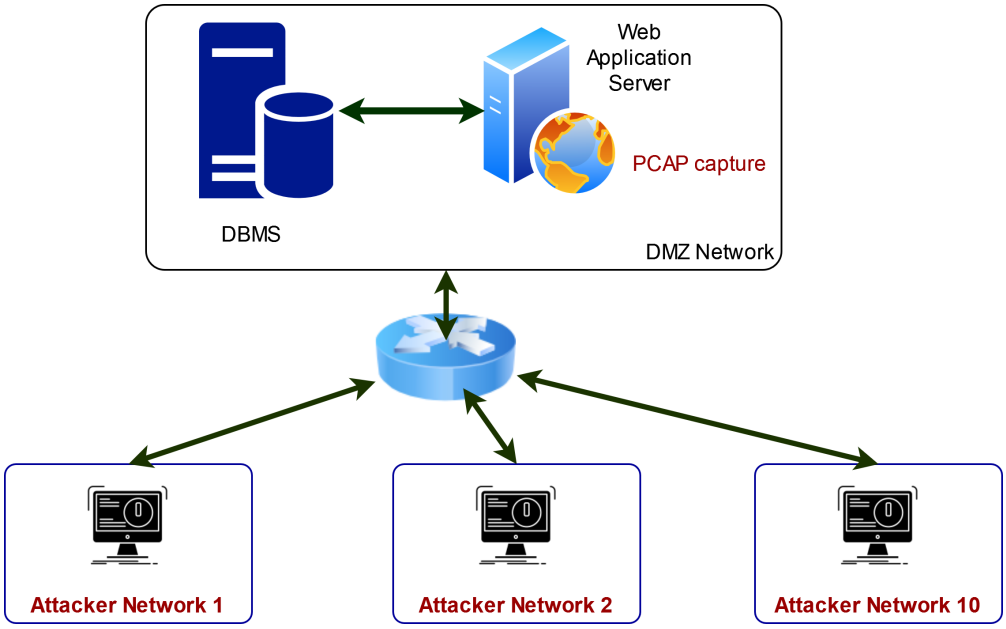
3.4.2 Tiêu chí đánh giá

Để đánh giá phương pháp phát hiện webshell, chúng tôi sử dụng một số số liệu phổ biến được tính toán từ ma trận nhầm lẫn như Accuracy, Precision, F1-score, Recall, False Positive Rate (FPR), ..., được tính theo True Positive (TP).), Dương tính giả (FP), Âm tính giả (FN) và Âm tính thật (TN).

3.4.3 Chuẩn bị tập dữ liệu

Chúng tôi đã xây dựng một hệ thống thử nghiệm như trong Hình. 3.2 được chia thành hai mạng hoàn toàn tách biệt, đó là Mạng DMZ và Mạng tấn công. Chúng tôi triển khai tất cả các thiết bị thông dụng và cần thiết, bao gồm bộ định tuyến, tường lửa, thiết bị chuyển mạch và 03 máy chủ là máy chủ web, máy chủ ứng dụng web và máy chủ cơ sở dữ liệu web. Trên Mạng tấn công, chúng tôi sử dụng hệ điều hành Kali Linux làm máy chủ tấn công sử dụng hơn 400 loại webshell PHP, ASP, ASPX, JS.

Chúng tôi sử dụng một số công cụ thu thập thông tin trang web để tạo lưu lượng truy cập HTTP thông thường với tư cách là truy cập hợp lệ. Ngoài ra, chúng tôi còn mô phỏng các cuộc tấn công Webshell bằng cách sử dụng Kali Linux để tải lên và thực thi webshell nhằm tạo lưu lượng xâm nhập. Suricata được sử dụng như một công cụ thu thập gói, lọc HTTP và lưu lưu lượng mạng vào các tệp PCAP.



Hình 3.2: Architecture of testbed system

Để xác thực và đánh giá tính hiệu quả của phương pháp DLWSD, chúng tôi sử dụng hai bộ dữ liệu:

- Dataset 1 (DS1): Đây là tập dữ liệu mà chúng tôi trực tiếp xây dựng thông qua hệ thống thử nghiệm được mô tả ở trên.
- Dataset 2 (DS2): Chúng tôi sử dụng bộ dữ liệu phổ biến CSE-CIC-IDS2018 được sử dụng trong nhiều các nghiên cứu.

3.5 Tối ưu tham số

Chúng tôi lựa chọn các tham số mô hình dựa trên một kỹ thuật có tên là Tối ưu hóa tham số [4]. Chúng tôi sử dụng Axe cho các thông số tối ưu. Có bốn siêu tham số có thể chuyển đổi và chúng nhận hai loại giá trị là phạm vi và lựa chọn như trong Bảng 3.1. Từ đó ta thu được tỷ lệ học là 0,003; lô 96; số kỷ nguyên 2; và [400, 100] cho các tính năng của lớp.

Bảng 3.1: Hyperparameter optimization value

Hyperparameter	**Value**	**Type**	**Optimal value**
Learning rate	[0.001, 1.0]	Range	0.003
Batch size	[16, 32, 48, 64, 96, 128]	Choice	64
Epochs	[1, 2, ..., 15, 16]	Choice	2
Layers	[[200, 100], [400, 100] [1,000, 500]]	Range	[400, 100]

Bảng 3.2: Comparison of DLWSD with other methods with DS2

Method	**Accuracy**(%)	**Precision**(%)	**F1-score**(%)	**Recall**(%)
DLWSD	**99.99**	**99.99**	**99.98**	**99.96**
DNN fast.ai [5]	99.92	99.85	99.85	99.85
DSSTE+ miniVGGNet [3]	96.97	97.94	97.04	96.97

3.5.1 Thực nghiệm

Từ phương pháp đề xuất, chúng tôi xây dựng công cụ để đánh giá phương pháp của mình với bộ dữ liệu đã được làm sạch nêu trên. Chúng tôi triển khai DLWSD dựa trên khung FastAI. Để xác thực tính hiệu quả của phương pháp DLWSD của chúng tôi, ba kịch bản đã được xây dựng và mô tả như sau:

- S1: Sử dụng liên tiếp DS1 và DS2 để thực hiện huấn luyện và kiểm tra phương pháp DLWSD mà không áp dụng hiệu chỉnh dữ liệu mất cân bằng.
- S2: Sử dụng liên tiếp DS1 và DS2 để thực hiện huấn luyện và kiểm tra phương pháp DLWSD có áp dụng quá trình xử lý dữ liệu không cân bằng.
- S3: Sử dụng DS3 để huấn luyện và kiểm tra DLWSD trong việc cân bằng các lớp.

3.5.2 So sánh kết quả

Để đánh giá khách quan tính hiệu quả của phương pháp DLWSD, chúng tôi đã thử nghiệm và so sánh kết quả với mô hình DNN sử dụng fast.ai trong [5] và mô hình DSSTE+miniVGGNet trong [3] trên cùng một CSE-CIC -IDS2018 tập dữ liệu ngày 02-03-2018 (DS2). Kết quả trong Bảng 3.2 cho thấy tất cả các chỉ số hiệu suất của phương pháp DLWSD đều cao hơn các phương pháp khác.

Bạn có thể truy cập miễn phí mã nguồn và tập dữ liệu được sử dụng trong thử nghiệm của chúng tôi từ liên kết GitHub: https://github.com/levietha0311/DLWSD/.

3.6 Tổng kết Chương 3

Trong Chương này, chúng tôi đề xuất phương pháp DLWSD dựa trên mô hình mạng học sâu DNN kết hợp với mô hình phát hiện dựa trên quy tắc truyền thống. So với kết quả cơ bản từ các nghiên cứu trước đây, DLWSD cũng cho thấy kết quả vượt trội trên cùng một tập dữ liệu thử nghiệm. Từ kết quả đó, chúng tôi đã xây dựng bộ DeepInspector chuyên dụng để phát hiện các cuộc tấn công xâm nhập kiểu khai thác Webshell. Bộ dò tìm này được tích hợp vào hệ thống NetIDSP theo cơ chế truyền thông Unix socket IPS. Phương pháp so khớp Regex đa mẫu bằng Hyperscan cũng được chúng tôi triển khai trong NetIDPS. Kết quả thử nghiệm thực tế cho phép chúng tôi kiểm soát, phát hiện và ngăn chặn các hành vi xâm nhập, đặc biệt là với hàng tấn hoạt động khai thác Webshell với luồng mạng 4x10Gbps.

Tổng kết và hướng nghiên cứu trong tương lai

Đóng góp chính của luận án

Luận án đã có những đóng góp cụ thể sau:

- Luận án đã thực hiện các nghiên cứu chuyên sâu về các kỹ thuật tấn công webshell, mã hóa webshell, trốn tránh và làm rối. Khảo sát nghiên cứu liên quan đến phát hiện tấn công chèn mã độc vào mã nguồn ứng dụng web. Từ những nghiên cứu này, giờ đây tôi có cái nhìn tổng quát hơn về các hướng nghiên cứu trong bài toán phát hiện tấn công webshell.

- Đề xuất Khung kiến trúc phân tích mã nguồn (ASAF) kết hợp các kỹ thuật dựa trên so khớp mẫu với thuật toán học sâu (DL). Cách tiếp cận kết hợp này tạo điều kiện cho việc phát hiện nhanh chóng và chính xác cả loại webshell đã biết và chưa biết. Khung kiến trúc được đề xuất đóng vai trò là kim chỉ nam để phát triển các mô hình cụ thể phù hợp với các ngôn ngữ lập trình khác nhau.

- Đề xuất hai giải pháp phát hiện webshell hoàn chỉnh dựa trên ASAF với các mô hình CNN được thiết kế riêng cho từng ngôn ngữ PHP và ASP.NET. Mỗi mô hình bao gồm một thuật toán biến đổi các tệp nguồn tương ứng thành các vectơ phẳng có chứa tất cả các tính năng của webshell. Hơn nữa, các mô hình này kết hợp các thuật toán ML/DL được tối ưu hóa cho các vấn đề phát hiện webshell

cụ thể để đảm bảo phát hiện hiệu quả với nguồn lực tính toán tối thiểu. Hiệu quả của các mô hình này sẽ được đánh giá dựa trên các tiêu chí đo lường xác định và so sánh với các nghiên cứu liên quan. *Hai giải pháp này đã được ứng dụng thực tế trong Đề tài nghiên cứu quốc gia số KC.01.19/16-20 do Bộ Khoa học và Công nghệ Việt Nam (MOST) cấp.* và *Giải pháp phát hiện đoạn mã độc trong mã nguồn ứng dụng web ASP.NET đã được Cục Sở hữu trí tuệ Việt Nam cấp bằng sáng chế.*

- Đề xuất phương pháp DLWSD kết hợp mô hình mạng học sâu DNN với mô hình phát hiện dựa trên quy tắc truyền thống. Bằng cách sửa hàm mất để giải quyết sự mất cân bằng của tập dữ liệu huấn luyện, phương pháp DLWSD được đề xuất của chúng tôi đã đạt được kết quả rất tốt trên cả tập dữ liệu do chúng tôi tạo ra (DS1) và tập dữ liệu uy tín từ Viện An ninh mạng Canada (DS2). Từ kết quả đó, chúng tôi đã phát triển một bộ DeepInspector chuyên dụng để phát hiện các cuộc tấn công xâm nhập kiểu khai thác webshell. Trình phát hiện này được tích hợp vào hệ thống NetIDPS bằng cơ chế giao tiếp IPS socket Unix. *Hệ thống NetIDPS này là kết quả nghiệm thu của Dự án Nghiên cứu Quốc gia KC.01.28/16-20 do Bộ Khoa học và Công nghệ Việt Nam (MOST) cấp.*

Hạn chế của luận án

Mặc dù luận án đã đạt được kết quả nghiên cứu tốt và có những đóng góp thiết thực như đã nêu ở trên nhưng luận án vẫn còn những hạn chế, cụ thể như sau:

- Hầu hết các nghiên cứu hiện tại liên quan đến việc phát hiện các cuộc tấn công webshell đều sử dụng các bộ dữ liệu được tạo tự động. Điều này cho thấy thực tế chưa có bộ dữ liệu webshell nào được coi là chuẩn và được sử dụng rộng rãi trong cộng đồng nghiên cứu. Không nằm ngoài xu hướng chung đó, luận án này cũng phải sử dụng bộ số liệu tự thu thập nên gây nhiều khó khăn trong việc so sánh khách quan kết quả của các nghiên cứu khác.

- Sự đa dạng của các ngôn ngữ lập trình phía máy chủ dẫn đến sự đa dạng của các loại webshell. Ngoài ra, mỗi loại webshell theo ngôn ngữ lập trình lại có những đặc điểm khác nhau nên cần xây dựng các phương pháp trích chọn đặc trưng khác nhau. Do hạn chế về thời gian và nguồn lực nên luận án chỉ chọn hai ngôn ngữ phổ biến nhất hiện nay là PHP và ASP.NET làm đối tượng nghiên cứu, thực nghiệm.

- Lĩnh vực trí tuệ nhân tạo hiện nay đang bùng nổ và không ngừng có những tiến

bộ mới. Việc giới thiệu các mô hình deep learning/machine learning tiên tiến liên tục được công bố. Do hạn chế về thời gian nên luận án chưa thể nghiên cứu và thử nghiệm các mô hình mới nhất hiện nay để áp dụng cho bài toán phát hiện webshell.

Hướng nghiên cứu trong tương lai

Mặc dù luận án đạt được những kết quả đáng kể nhưng lĩnh vực này vẫn còn nhiều dư địa để mở rộng nghiên cứu. Trong các công trình tiếp theo, chúng tôi mong muốn tìm hiểu các hướng nghiên cứu sau:

- Tiến hành khảo sát tổng thể các bộ dữ liệu webshell được sử dụng trong nghiên cứu hiện tại, từ đó xây dựng bộ dữ liệu làm chuẩn cho các nghiên cứu sau này liên quan đến webshell.
- Tiếp tục nghiên cứu và thử nghiệm các mô hình DL/ML đơn lẻ và các mô hình tổng hợp mới nhất để cải thiện khả năng phát hiện chính xác các webshell nâng cao.
- Nghiên cứu sâu hơn về cơ chế hoạt động và đặc điểm của webshell cho phép xây dựng các bộ công cụ tự động hóa quá trình tạo quy tắc YARA.
- Mở rộng nghiên cứu về webshell viết bằng các ngôn ngữ khác như JSP, Ruby, Python,... hướng tới xây dựng mô hình tổng quát có thể phát hiện hiệu quả tất cả các loại webshell mà không phụ thuộc vào ngôn ngữ lập trình.

Danh mục công bố

1. Nguyễn Ngọc Hoá và **Lê Việt Hà** Phương pháp phát hiện đoạn mã độc trong mã nguồn ứng dụng web sử dụng ngôn ngữ ASP.NET. 1-0036538-000 (27/06/2023).

2. **Le, Viet Ha** and Nguyen, Ngoc Tu and Nguyen, Ngoc Hoa and Le, Linh (2021) An Efficient Hybrid Webshell Detection Method for Webserver of Marine Transportation Systems. IEEE Transactions on Intelligent Transportation Systems. ISSN 1524-9050 (SCI, Scopus-Q1 2021).

3. **Le, Viet Ha** and Du, Phuong Hanh and Nguyen, Ngoc Cuong and Nguyen, Ngoc Hoa and Hoang, Viet Long (2021) A Proactive Method of the Webshell Detection and Prevention based on Deep Traffic Analysis. International Journal of Web and Grid Services. ISSN 1741-1114 (SCI-E, Scopus-Q1 2021);

4. **Le, Viet Ha** and Phung, Van On and Nguyen, Ngoc Hoa (2020) Information Security Risk Management by a Holistic Approach: a Case Study for Vietnamese e-Government. IJCSNS International Journal of Computer Science and Network Security, 20 (6). pp. 72-82. ISSN 1738-7906 (E-SCI 2020);

5. Phung, Van On and **Le, Viet Ha** and Nguyen, Ngoc Hoa, A Solution for Assessing and Managing Information Security Risks in e-Government. Journal of Science and Technology on Information Security, 1(13), pp. 35-48, 2022, ISSN 1859-1256, DOI: 10.54654/isj.v1i13.144.

6. Nguyen, Hoa & **Le, Viet Ha** & Phung, Van-On & Du, Phuong-Hanh. (2019). Toward a Deep Learning Approach for Detecting PHP Webshell. SoICT 2019: Proceedings of the Tenth International Symposium on Information and Communication Technology. 514-521. 10.1145/3368926.3369733;

7. **Ha V. Le**, Hoang V. Vo, Tu N. Nguyen, Hoa N. Nguyen, and Hung T. Du (2022) Towards a Webshell Detection Approach Using Rule-Based and Deep HTTP Traffic Analysis. Computational Collective Intelligence: 14th International Conference, ICCCI 2022. vol 13501 pp. 571–584.