ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Báo cáo cuối kỳ

TẠO BỘ DỮ LIỆU THỬ NGHIỆM CHO BÀI TOÁN PHÁT HIỆN TẤN CÔNG XSS BẰNG HỌC SÂU

Lớp: IE105.P11.CNVN

Giáo viên hướng dẫn: Nguyễn Tấn Cầm

Sinh viên thực hiện:

Nguyễn Trần Bảo Anh	MSSV: 22520066
Nguyễn Thị Trâm Đan	MSSV: 22520185
Dương Anh Vũ	MSSV: 22521688

TP.Hồ Chí Minh, ngày 6 tháng 12 năm 2024

MỤC LỤC

Mục lục hình ảnh:

Bảng phân công

Người thực hiện	Nội dung công việc	Đánh giá
Nguyễn Trần Bảo Anh MSSV: 22520066	1. Thu thập dataset 2. Train mô hình 3. Slide báo cáo 4. Viết báo cáo: Chương 4: Tạo bộ dữ liệu cho bài toán phát hiện tấn công XSS. 5. Viết báo cáo: Chương 5: Huấn luyện mô hình và đánh giá.	Hoàn thành Tỷ lệ đóng góp: 35%
Nguyễn Thị Trâm Đan MSSV: 22520185	1. Viết báo cáo: Chương 1: Giới thiệu 2. Viết báo cáo: Chương 4: Tạo bộ dữ liệu cho bài toán phát hiện tấn công XSS. 3. Viết báo cáo: Chương 5: Huấn luyện mô hình và đánh giá. 4. Tổng hợp viết báo cáo 5. Thu thập dataset	Hoàn thành Tỷ lệ đóng góp: 35%
Dương Anh Vũ MSSV: 22521688	1. Viết chương 2: Cơ sở lý thuyết và các khái niệm liên quan. 2. Viết chương 3: Mô hình CNN 3. Thu thập dataset	Hoàn thành Tỷ lệ đóng góp: 30%

CHƯƠNG 1: GIỚI THIỆU

1.1. Giới thiệu tổng quan

1.1.1. Tổng quan về An ninh mạng:

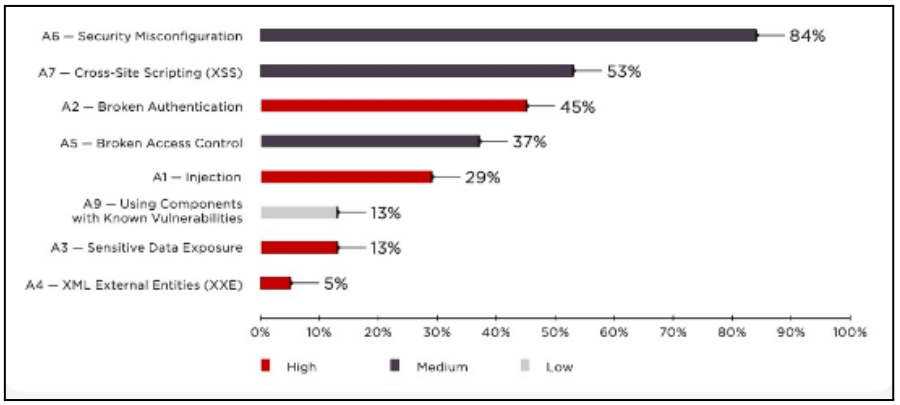
An ninh mạng là một lĩnh vực thiết yếu nhằm bảo vệ hệ thống máy tính, dữ liệu và mạng khỏi các cuộc tấn công, truy cập trái phép, hoặc hành vi gian lận. Khi các doanh nghiệp, tổ chức và cá nhân phụ thuộc ngày càng nhiều vào các dịch vụ và giao dịch trực tuyến, việc bảo vệ các tài nguyên số trở nên cực kỳ quan trọng. An ninh mạng không chỉ giúp đảm bảo tính bảo mật của thông tin mà còn tăng cường độ tin cậy cho các hệ thống, góp phần phát triển môi trường mạng an toàn và lành mạnh.

1.1.2. Các cuộc tấn công mạng:

Hiện nay, các cuộc tấn công mạng phổ biến bao gồm tấn công từ chối dịch vụ phân tán (DDoS), tấn công lừa đảo (phishing), ransomware, và đặc biệt là Cross-Site Scripting (XSS).

Cross-Site Scripting (XSS) là một lỗ hổng bảo mật trong các ứng dụng web, cho phép kẻ tấn công can thiệp vào các hoạt động của người dùng đối với ứng dụng đó. Lỗ hổng này cho phép kẻ tấn công vượt qua Same Origin Policy - chính sách bảo mật nhằm ngăn chặn việc truy cập tài nguyên giữa các trang web khác nhau. Khi khai thác XSS, kẻ tấn công có thể giả dạng người dùng nạn nhân, thực hiện các hành động mà nạn nhân có thể làm, và truy cập vào dữ liệu của nạn nhân. Nếu nạn nhân có quyền truy cập đặc quyền, kẻ tấn công thậm chí có thể nắm toàn quyền kiểm soát các chức năng và dữ liệu của ứng dụng.

XSS là một loại tấn công dạng injection, nơi mà mã độc được chèn vào các trang web tưởng chừng như an toàn và đáng tin cậy. Tấn công XSS xảy ra khi kẻ tấn công sử dụng ứng dụng web để gửi mã độc (thường dưới dạng script) tới một người dùng khác. Lỗ hổng này tồn tại khi ứng dụng web sử dụng đầu vào của người dùng trên client mà không có cơ chế mã hóa hoặc lọc dữ liệu. Các đoạn mã độc này sau đó được thực thi phía client trên trình duyệt của nạn nhân, gây ra nguy cơ nghiêm trọng về bảo mật và bảo vệ thông tin người dùng.



Hình 1: Tỷ lệ các lỗ hổng bảo mật ứng dụng web

1.1.3. Tầm quan trọng của việc phát hiện và phòng ngừa XSS:

Cross-Site Scripting (XSS) là một trong những lỗ hổng bảo mật phổ biến và nguy hiểm nhất đối với các ứng dụng web, ảnh hưởng đến cả doanh nghiệp và người dùng. Việc phát hiện và phòng ngừa XSS là rất quan trọng vì những lý do sau:

Bảo vệ dữ liệu nhạy cảm của người dùng: XSS cho phép kẻ tấn công đánh cắp thông tin nhạy cảm của người dùng như cookie, thông tin đăng nhập và dữ liệu cá nhân. Với các thông tin này, kẻ tấn công có thể giả dạng người dùng, truy cập và thao túng tài khoản của họ hoặc thậm chí lừa đảo người dùng khác. Phòng ngừa XSS giúp đảm bảo rằng thông tin cá nhân của người dùng luôn được bảo mật và không bị kẻ xấu lợi dụng.

Giảm thiểu nguy cơ chiếm quyền kiểm soát tài khoản: Kẻ tấn công có thể khai thác XSS để thực hiện các hành động trái phép trong tài khoản của người dùng mà họ không hề hay biết, ví dụ như thay đổi cài đặt tài khoản hoặc chuyển tiền trong các dịch vụ tài chính. Điều này đặc biệt nguy hiểm với người dùng có quyền truy cập đặc biệt hoặc quyền quản trị, vì kẻ tấn công có thể có toàn quyền truy cập vào dữ liệu và chức năng của hệ thống.

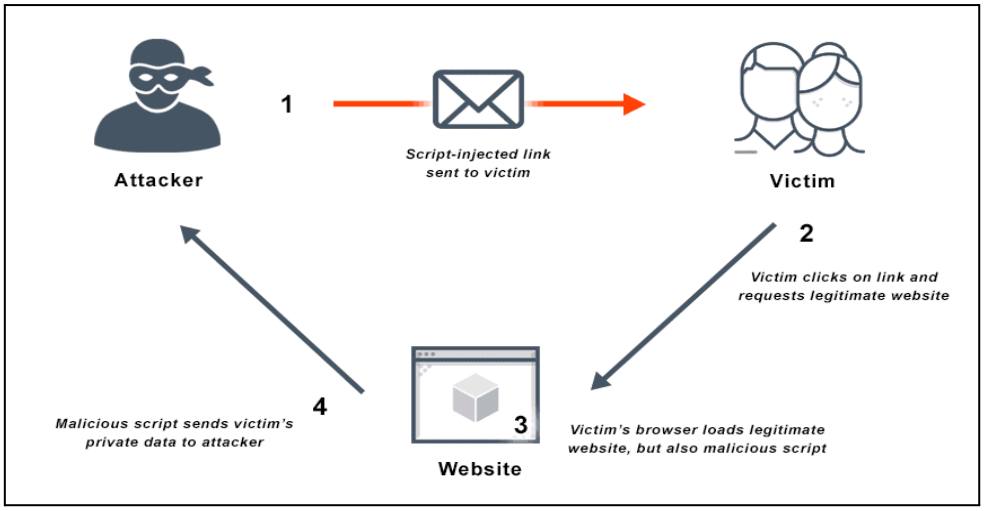
Duy trì sự tin cậy của người dùng: Khi các cuộc tấn công XSS xảy ra, người dùng có thể mất lòng tin vào hệ thống hoặc dịch vụ mà họ đang sử dụng, ảnh hưởng đến uy tín của doanh nghiệp. Các sự cố bảo mật nghiêm trọng có thể gây thiệt hại đáng kể cho thương hiệu và khiến người dùng chuyển sang sử dụng dịch vụ của đối thủ. Việc phát hiện và ngăn chặn XSS giúp duy trì lòng tin của người dùng vào hệ thống.

Tránh tổn thất tài chính và thiệt hại pháp lý: Những cuộc tấn công XSS có thể dẫn đến thiệt hại tài chính cho các doanh nghiệp, bao gồm chi phí xử lý sự cố, bồi

thường cho khách hàng và tổn thất doanh thu do mất lòng tin của người dùng. Ngoài ra, nhiều quốc gia có các quy định nghiêm ngặt về bảo vệ dữ liệu người dùng, và các tổ chức có thể phải đối mặt với các hình phạt pháp lý nếu không tuân thủ các tiêu chuẩn bảo mật.

Đảm bảo tuân thủ các tiêu chuẩn bảo mật: Các tiêu chuẩn bảo mật như OWASP Top 10 liệt kê XSS là một trong những lỗ hổng bảo mật hàng đầu mà các ứng dụng web cần phòng ngừa. Tuân thủ các tiêu chuẩn này giúp doanh nghiệp duy trì an toàn cho hệ thống, tránh những rủi ro không đáng có và xây dựng nền tảng bảo mật vững chắc cho các dịch vụ trực tuyến.

Tóm lại, việc phát hiện và phòng ngừa XSS không chỉ bảo vệ dữ liệu người dùng mà còn giúp doanh nghiệp duy trì uy tín, tránh các tổn thất tài chính và đảm bảo tuân thủ quy định pháp lý. Trong bối cảnh các cuộc tấn công mạng ngày càng tinh vi, đầu tư vào phát hiện và ngăn chặn XSS là một phần quan trọng trong chiến lược an ninh của mọi tổ chức.



Hình 2: Minh họa về một cuộc tấn công Cross-Site Scripting (XSS)

1.2. Mục tiêu và phạm vi nghiên cứu

1.2.1. Mục tiêu nghiên cứu:

Xây dựng kiến thức vững chắc về an ninh mạng và các kỹ thuật tấn công mạng.

Nghiên cứu các kỹ thuật phát hiện và phòng ngừa XSS hiệu quả.

Đề xuất các phương pháp, công cụ và chiến lược để phát hiện và ngăn chặn XSS.

1.2.2. Phạm vi nghiên cứu:

Tập trung vào loại tấn công XSS (Cross-Site Scripting), bao gồm các loại XSS phổ biến như Stored XSS, Reflected XSS và DOM-based XSS.

Phân tích các tình huống thực tế, ví dụ cụ thể, và các công cụ bảo mật sử dụng cho việc phát hiện và phòng ngừa XSS.

Không đi sâu vào các loại tấn công khác như SQL Injection, DDoS, hoặc Malware.

1.3. Kết quả dự kiến

Hoàn thiện tài liệu nghiên cứu chi tiết về XSS, bao gồm khái niệm, phân loại và mức độ nguy hiểm.

Phân tích được các phương pháp phòng ngừa, công cụ hỗ trợ phát hiện XSS hiệu quả nhất.

Đề xuất các giải pháp và quy trình phòng chống XSS cho hệ thống và người dùng.

CHƯƠNG 2: CƠ SỞ LÍ THUYẾT VÀ CÁC KHÁI NIỆM LIÊN QUAN

2.1. Giới thiệu về tấn công XSS (Cross-Site Scripting)

2.1.1. Tấn công XSS là gì ?

Các cuộc tấn công Cross-Site Scripting (XSS) là một dạng tấn công chèn mã độc, trong đó các tập lệnh độc hại được chèn vào các trang web vốn dĩ an toàn và đáng tin cậy. Tấn công XSS xảy ra khi kẻ tấn công sử dụng một ứng dụng web để gửi mã độc, thường là các tập lệnh phía trình duyệt, đến một người dùng khác. Lỗi bảo mật cho phép các cuộc tấn công này thành công rất phổ biến và có thể xảy ra bất cứ nơi nào một ứng dụng web sử dụng dữ liệu đầu vào từ người dùng mà không kiểm tra hoặc mã hóa nó trước khi đưa vào đầu ra.

Kẻ tấn công có thể sử dụng XSS để gửi tập lệnh độc hại đến một người dùng không ngờ tới. Trình duyệt của người dùng không có cách nào để biết rằng tập lệnh này không đáng tin cậy và sẽ thực thi tập lệnh đó. Do trình duyệt cho rằng tập lệnh đến từ một nguồn đáng tin cậy, nên tập lệnh độc hại có thể truy cập bất kỳ cookie, mã phiên, hoặc thông tin nhạy cảm nào khác được trình duyệt lưu trữ và sử dụng cho trang web đó. Những tập lệnh này thậm chí có thể viết lại nội dung của trang HTML.

Hình 3: Ví dụ minh họa về truy xuất thông tin cookie của người dùng

2.1.2. Hậu quả tấn công XSS

Tấn công XSS có thể làm tổn hại uy tín của trang web và gây ra mất mát dữ liệu, tổn thất tài chính, và vi phạm quyền riêng tư. Một số hậu quả như:

Kẻ tấn công có thể đánh cắp cookie phiên, thông tin đăng nhập, hoặc dữ liệu nhạy cảm khác của người dùng. Khi chiếm được cookie, kẻ tấn công có thể mạo danh người dùng để thực hiện các hành động trái phép trên trang web.

Nếu tấn công thành công, kẻ tấn công có thể chiếm quyền kiểm soát tài khoản của người dùng, bao gồm tài khoản cá nhân, tài khoản ngân hàng, hoặc các hệ thống quản lý nội dung. Điều này có thể dẫn đến tổn thất tài chính hoặc mất dữ liệu cá nhân.

Các tập lệnh độc hại có thể được sử dụng để thay đổi hoặc làm sai lệch nội dung của trang web, dẫn đến việc người dùng tiếp nhận thông tin không chính xác hoặc bị chuyển hướng đến các trang web lừa đảo.

Kẻ tấn công có thể sử dụng XSS để tạo các biểu mẫu giả mạo hoặc thông báo giả, lừa người dùng cung cấp thông tin nhạy cảm, chẳng hạn như mật khẩu hoặc số thẻ tín dụng.

2.1.3. Các loại tấn công XSS phổ biến

2.1.3.1. Stored XSS (XSS lưu trữ)

Stored XSS là một kỹ thuật tấn công mà, mã độc được chèn vào một ứng dụng web và được lưu trữ trên máy chủ, ví dụ như trong cơ sở dữ liệu, hệ thống lưu trữ tin nhắn, hoặc các khu vực lưu trữ dữ liệu do người dùng cung cấp (như bài viết, nhận xét, hoặc thông tin cá nhân).

Khi người dùng khác truy cập vào nội dung có chứa mã độc này, mã sẽ được thực thi tự động trong trình duyệt của họ mà không cần bất kỳ hành động nào từ phía người dùng. Điều này cho phép kẻ tấn công tiếp cận và đánh cắp thông tin người dùng hoặc thực hiện các hành vi độc hại khác.

2.1.3.2. Reflected XSS (XSS phản hồi)

Reflected XSS (hay còn gọi là Non-Persistent XSS) là một dạng tấn công Cross-Site Scripting (XSS) mà trong đó mã độc được "phản chiếu" từ trang web ngay lập tức vào trình duyệt của người dùng mà không được lưu trữ trên máy chủ. Kẻ tấn công có thể lợi dụng lỗ hổng này để gửi mã độc thông qua các tham số trong URL hoặc các trường tìm kiếm mà trang web phản hồi lại mà không kiểm tra hoặc mã hóa đúng cách.

Điểm khác biệt chính giữa Stored XSS và Reflected XSS là Reflected XSS không lưu trữ mã độc trên máy chủ mà chỉ phản chiếu mã độc ngay trong phản hồi trả về của trang web.

Reflected XSS và Stored XSS có hai sự khác biệt lớn trong quá trình tấn công:

Đầu tiên, để khai thác Reflected XSS, kẻ tấn công cần phải lừa nạn nhân truy cập vào URL chứa mã độc. Trong khi đó, với Stored XSS, kẻ tấn công không cần phải thực hiện bước này; sau khi chèn mã độc vào cơ sở dữ liệu của ứng dụng, kẻ tấn công chỉ cần chờ đợi nạn nhân truy cập trang mà không hay biết về sự nguy hiểm. Đối với nạn nhân, việc này hoàn toàn bình thường vì họ không nhận ra rằng dữ liệu họ đang truy cập đã bị nhiễm độc.

Thứ hai, mục tiêu của kẻ tấn công dễ dàng đạt được hơn nếu nạn nhân vẫn đang trong phiên làm việc (session) của ứng dụng web vào thời điểm tấn công. Đối với Reflected XSS, kẻ tấn công có thể thuyết phục hoặc lừa nạn nhân đăng nhập và truy cập vào URL độc hại để thực thi mã độc. Tuy nhiên, với Stored XSS, vì mã độc đã được lưu trong cơ sở dữ liệu web, mỗi lần người dùng truy cập vào các chức năng liên quan, mã độc sẽ được thực thi. Hơn nữa, nhiều chức năng này yêu cầu người dùng phải đăng nhập, do đó nạn nhân chắc chắn vẫn đang trong phiên làm việc của ứng dụng.

Từ những điểm này, có thể thấy Stored XSS nguy hiểm hơn nhiều so với Reflected XSS, vì đối tượng bị ảnh hưởng có thể là tất cả người dùng của ứng dụng web. Nếu nạn nhân có quyền quản trị, nguy cơ bị chiếm quyền điều khiển web càng lớn hơn.

2.1.3.3. DOM-based XSS

DOM-based XSS (Document Object Model-based XSS) là một dạng tấn công XSS đặc biệt, trong đó mã độc không được gửi trực tiếp từ máy chủ, mà thay vào đó, mã độc được thực thi khi trình duyệt xử lý dữ liệu trong DOM (Document Object Model) của trang web. Tấn công DOM-based XSS xảy ra khi mã JavaScript trong trang web sử dụng dữ liệu từ người dùng (thông qua URL hoặc các trường nhập liệu) và không xử lý dữ liệu đúng cách trước khi đưa vào DOM. Dữ liệu này sau đó có thể được sử dụng trong các phương thức JavaScript để thao tác DOM và thực thi mã độc.

Khác với Reflected XSS hay Stored XSS, nơi mã độc chủ yếu được gửi từ máy chủ, trong DOM-based XSS, mã độc được tiêm vào thông qua các thao tác JavaScript phía client, và quá trình tấn công chủ yếu diễn ra trong trình duyệt của nạn nhân.

Hình 4: Ví dụ minh họa về tấn công DOM-based XSS

2.1.4. Cách giảm thiểu các lỗi liên quan đến XSS

Chúng ta có thể sử dụng một số phương thức để giảm thiểu các lỗi liên quan đến lỗ hổng bảo mật XSS:

Bật Chính sách bảo mật nội dung (CSP) là một biện pháp bảo mật có thể giúp ngăn chặn mã JavaScript không mong muốn thực thi trên trang web của bạn. CSP cho phép bạn định nghĩa các nguồn hợp lệ cho các tài nguyên như JavaScript, CSS, hình ảnh và nhiều thứ khác.Ví dụ, chỉ cho phép JavaScript chỉ được tải từ một số nguồn tin cậy, giúp ngăn ngừa việc tải và thực thi mã độc từ các nguồn không rõ ràng.

Loại bỏ dữ liệu yêu cầu HTTP không đáng tin cậy dựa trên ngữ cảnh trong đầu ra HTML sẽ ngăn ngừa các lỗ hổng XSS (Cross-Site Scripting). Điều này đảm bảo rằng bất kỳ dữ liệu nào được lấy từ người dùng và đưa vào HTML hoặc các thuộc tính của HTML (ví dụ như src, href, alt, v.v.) đều được xử lý một cách an toàn, không cho phép thực thi mã độc.

Sử dụng các framework hiện đại như Ruby on Rails, Angular,... Chúng cung cấp các công cụ và tính năng bảo mật tích hợp sẵn để tự động xử lý và bảo vệ ứng dụng web khỏi các lỗ hổng XSS (Cross-Site Scripting). Điều này giúp giảm thiểu lỗi

bảo mật do lập trình viên gây ra, đồng thời cung cấp các biện pháp an toàn giúp ngăn ngừa các cuộc tấn công XSS.

2.1.5. Công cụ kiểm tra và phát hiện XSS

Để kiểm tra và phát hiện các lỗ hổng XSS (Cross-Site Scripting), có nhiều công cụ và phần mềm giúp xác định và bảo vệ ứng dụng web khỏi các cuộc tấn công. Một số công cụ phổ biến được sử dụng như:

OWASP ZAP là một công cụ mã nguồn mở mạnh mẽ được phát triển bởi OWASP, giúp phát hiện các lỗ hổng bảo mật trong các ứng dụng web, bao gồm XSS. ZAP có tính năng quét tự động và các công cụ tìm kiếm lỗ hổng như XSS.

Burp Suite là một trong những công cụ bảo mật ứng dụng web phổ biến nhất, cung cấp một loạt các công cụ để kiểm tra và phát hiện XSS và các lỗ hổng bảo mật khác. Phiên bản Burp Suite Professional có nhiều tính năng mạnh mẽ như quét tự động, phát hiện lỗ hổng, và các công cụ phân tích động.

Acunetix là một công cụ quét bảo mật tự động giúp phát hiện và kiểm tra các lỗ hổng bảo mật, bao gồm XSS, SQL Injection, và các lỗ hổng khác. Acunetix có khả năng quét cả ứng dụng web và các API.

2.2. Tổng quan về Deep Learning

2.2.1. Deep Learning là gì?

Học sâu (Deep learning) là một nhánh của học máy, sử dụng các mạng nơ-ron để thực hiện các nhiệm vụ như phân loại, hồi quy và học biểu diễn. Lĩnh vực này lấy cảm hứng từ thần kinh học sinh học, với các mạng nơ-ron được xếp thành nhiều lớp để xử lý dữ liệu. Các phương pháp học sâu có thể là có giám sát, bán giám sát hoặc không giám sát. Các kiến trúc phổ biến như mạng kết nối đầy đủ, mạng tích chập (CNN), mạng nơ-ron hồi tiếp (RNN), và mạng đối kháng tạo sinh (GAN) đã được ứng dụng thành công trong nhiều lĩnh vực như thị giác máy tính, nhận dạng giọng nói, xử lý ngôn ngữ tự nhiên và y tế. Mặc dù lấy cảm hứng từ bộ não con người, các mạng nơ-ron hiện tại không mô phỏng hoàn toàn chức năng não bộ.

2.2.2. Cấu trúc và nguyên lý hoạt động của Deep Learning

Deep learning sử dụng các mạng nơ-ron nhân tạo để mô phỏng cách thức mà bộ não con người xử lý và học từ thông tin. Mạng nơ-ron bao gồm các "nơ-ron" (các đơn vị xử lý) kết nối với nhau theo một cấu trúc tầng lớp.

Một mạng nơ-ron nhân tạo bao gồm ba loại lớp chính:

Lớp đầu vào (Input Layer): Đây là nơi mạng tiếp nhận dữ liệu ban đầu. Dữ liệu có thể là hình ảnh, âm thanh, văn bản, hoặc bất kỳ loại dữ liệu nào khác mà mạng cần

xử lý.Lớp này có số lượng nơ-ron bằng với số lượng đặc trưng hoặc yếu tố đầu vào của dữ liệu. Ví dụ, đối với một hình ảnh có kích thước 28x28 pixel, lớp đầu vào sẽ có 784 nơ-ron.

Lớp ẩn (Hidden Layers):Đây là các lớp nằm giữa lớp đầu vào và lớp đầu ra. Mỗi lớp ẩn chứa nhiều nơ-ron, và mỗi nơ-ron trong lớp này kết nối với các nơ-ron ở các lớp trước và sau.Số lượng lớp ẩn và số nơ-ron trong mỗi lớp là yếu tố quan trọng quyết định sức mạnh của mô hình deep learning. Mạng có nhiều lớp ẩn thường được gọi là "mạng nơ-ron sâu" (deep neural network).Mỗi nơ-ron trong lớp ẩn thực hiện một phép toán nhất định để chuyển đổi tín hiệu đầu vào thành một giá trị xuất ra, và giá trị này được truyền đến các lớp tiếp theo.

Lớp đầu ra (Output Layer):Đây là lớp cuối cùng trong mạng nơ-ron, nơi mạng tạo ra kết quả dự đoán hoặc phân loại. Số lượng nơ-ron trong lớp đầu ra thường phụ thuộc vào bài toán. Ví dụ, đối với bài toán phân loại hai lớp, lớp đầu ra chỉ có một nơ-ron. Còn với bài toán phân loại nhiều lớp, lớp đầu ra sẽ có nhiều nơ-ron tương ứng với số lớp cần phân loại.

Mỗi nơ-ron trong các lớp này có các trọng số và độ lệch, được điều chỉnh trong quá trình huấn luyện bằng cách sử dụng thuật toán lan truyền ngược (Backpropagation).

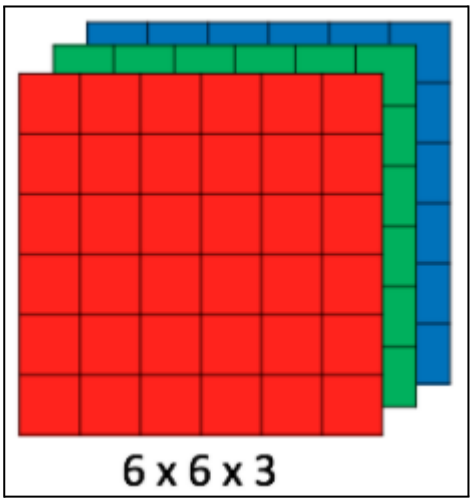
Chương 3: MÔ HÌNH CNN

Trong chương này, chúng ta sẽ tìm hiểu về mạng nơ-ron tích chập (Convolutional Neural Network - CNN), một trong những kiến trúc mạng nơ-ron phổ biến nhất trong lĩnh vực xử lý ảnh và thị giác máy tính. CNN được thiết kế đặc biệt để nhận diện và phân loại các mẫu từ dữ liệu hình ảnh.

3.1. Giới thiệu về CNN

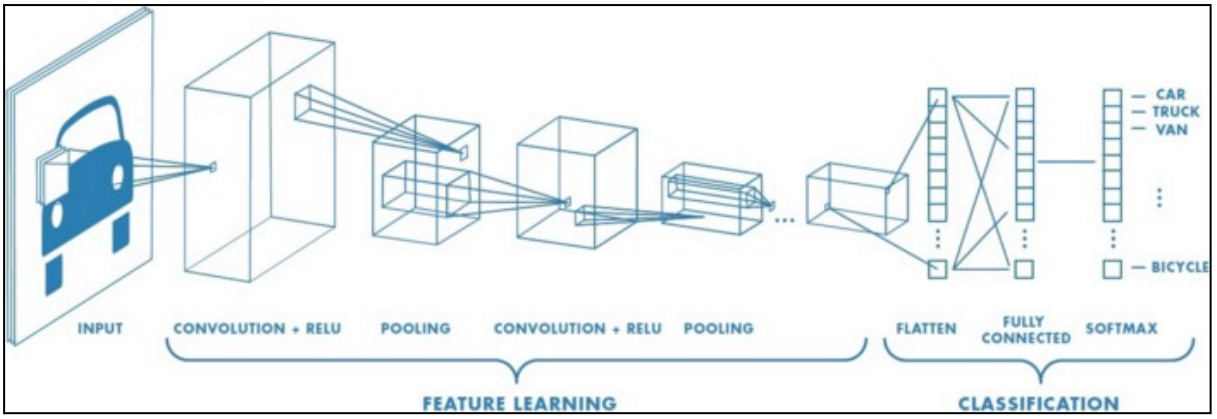
CNN bao gồm một chuỗi các lớp tích chập (convolutional layers), các lớp pooling, và các lớp kết nối đầy đủ (fully connected layers). Các lớp này kết hợp với nhau giúp trích xuất các đặc trưng từ hình ảnh và đưa ra kết quả dự đoán.

CNN phân loại hình ảnh bằng cách lấy 1 hình ảnh đầu vào, xử lý và phân loại nó theo các hạng mục nhất định (Ví dụ: Chó, Mèo, Hổ, ...). Máy tính coi hình ảnh đầu vào là 1 mảng pixel và nó phụ thuộc vào độ phân giải của hình ảnh. Dựa trên độ phân giải hình ảnh, máy tính sẽ thấy H x W x D (H: Chiều cao, W: Chiều rộng, D: Độ dày). Ví dụ



Hình 5: Mảng ma trận RGB 6x6x3 (3 ở đây là giá trị RGB).

Về kỹ thuật, mô hình CNN để training và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernels), tổng hợp lại các lớp được kết nối đầy đủ (Full Connected) và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác suất giữa 0 và 1. Hình dưới đây là toàn bộ luồng CNN để xử lý hình ảnh đầu vào và phân loại các đối tượng dựa trên giá trị.



Hình 6: Toàn bộ luồng CNN xử lý hình ảnh đầu vào, phân loại các đối tượng dựa trên giá trị.

Các thành phần chính của CNN:

- Convolutional Layer: Lớp tích chập
- Activation Function: Hàm kích hoạt
- Pooling Layer: Lớp lấy mẫu (Max Pooling hoặc Average Pooling)
- Flatten Layer: Lớp phẳng hóa
- Fully Connected Layer: Lớp kết nối đầy đủ
- Dropout Layer: Lớp giảm thiểu quá khớp

3.2. Kiến trúc của mô hình CNN:

Dưới đây là một ví dụ về kiến trúc cơ bản của một mô hình CNN dùng cho bài toán phân loại hình ảnh.

Kiến trúc ví dụ:

Lớp	Kích thước đầu ra	Tham số
Input	64 x 64 x 3	Hình ảnh RGB 64x64
Convolutional Layer	32 filters (3x3), ReLU	stride=1, padding=same
Max Pooling Layer	32 filters (2x2)	stride=2
Convolutional Layer	64 filters (3x3), ReLU	stride=1, padding=same
Max Pooling Layer	64 filters (2x2)	stride=2
Flatten Layer	4096	

Fully Connected	128 neurons, ReLU	
Dropout Layer	50%	
Output Layer	10 neurons, Softmax	10 lớp phân loại

Dưới đây là ví dụ về mã nguồn của kiến trúc CNN sử dụng Keras:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
Dropout

# Tạo mô hình CNN
model = Sequential()

# Lớp Convolutional 1
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', padding='same',
input_shape=(64, 64, 3)))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))

# Lớp Convolutional 2
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))

# Lớp Flatten
model.add(Flatten())

# Lớp Fully Connected
model.add(Dense(128, activation='relu'))

# Lớp Dropout
model.add(Dropout(0.5))

# Lớp Output
model.add(Dense(10, activation='softmax'))

# Hiển thị kiến trúc mô hình
model.summary()
```

Giải thích mã nguồn:

- Conv2D: Lớp tích chập với kích thước kernel, hàm kích hoạt và padding được chỉ định.
- MaxPooling2D: Lớp pooling giúp giảm kích thước feature map.
- Flatten: Lớp phẳng hóa chuyển đổi feature map thành vector.
- Dense: Lớp Fully Connected để kết nối các nơ-ron.

- Dropout: Lớp giảm overfitting bằng cách bỏ ngẫu nhiên các nơ-ron trong quá trình huấn luyện.
- Softmax: Dùng cho lớp đầu ra để tính xác suất các lớp.

3.3. Các lớp nơ-ron và chức năng

3.3.1. Convolutional Layer (Lớp tích chập)

Lớp convolution là thành phần quan trọng nhất trong CNN. Nó giúp mạng học các đặc trưng từ dữ liệu đầu vào bằng cách sử dụng bộ lọc (filter/kernel). Các bộ lọc này sẽ trượt qua toàn bộ ảnh và tính toán các giá trị đặc trưng thông qua phép tích chập (convolution operation). Quá trình này giúp mạng tự động trích xuất các đặc trưng như cạnh, góc, hình dạng, và các mẫu phức tạp hơn từ dữ liệu hình ảnh.

Trích xuất đặc trưng cục bộ: Lớp Convolution giúp tự động trích xuất các đặc trưng như cạnh, góc, và đường biên từ hình ảnh. Thay vì yêu cầu con người phải chọn lọc các đặc trưng thủ công, CNN tự học các đặc trưng này từ dữ liệu đầu vào.

Giảm số lượng tham số: Không giống như các lớp dense (fully connected), mỗi nơ-ron trong lớp convolution chỉ kết nối với một vùng nhỏ của ảnh (receptive field), giúp giảm số lượng kết nối và tham số cần học.

Chia sẻ trọng số (Weight Sharing): Các bộ lọc (filter/kernel) sử dụng chung một tập trọng số cho toàn bộ ảnh, giúp mô hình học được các đặc trưng giống nhau ở nhiều vị trí khác nhau của ảnh.

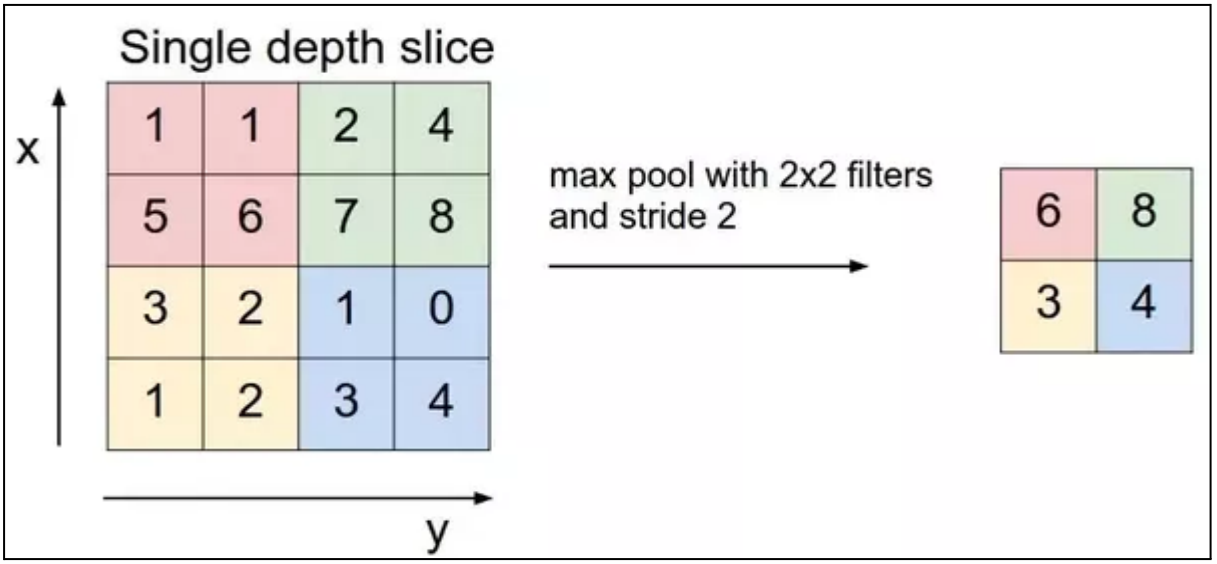
3.3.2. Pooling Layer (Lớp lấy mẫu)

Lớp Pooling trong CNN là một lớp quan trọng giúp giảm kích thước của feature map (tức là giảm độ phân giải của ảnh sau khi đã qua lớp Convolution), đồng thời giúp tăng khả năng tổng quát hóa của mô hình và giảm thiểu overfitting. Các loại pooling phổ biến là Max Pooling và Average Pooling, với Max Pooling là phương pháp thường được sử dụng nhất.

3.3.2.1 Max Pooling

Max Pooling là phương pháp chọn giá trị lớn nhất trong một vùng cụ thể của feature map. Quá trình này giúp giữ lại các đặc trưng mạnh mẽ nhất từ mỗi vùng của

ảnh, giúp mô hình nhận diện các đặc trưng mạnh và nổi bật.



Hình 7: Hình minh hoạ của Max Pooling

Cơ chế hoạt động:

- Kích thước cửa sổ (Window Size): Chọn một vùng nhỏ của feature map (ví dụ: 2×2 hoặc 3×3.
- Stride: Bước di chuyển của cửa sổ pooling. Mỗi lần cửa sổ di chuyển qua một vùng mới, nó sẽ chọn giá trị lớn nhất trong vùng đó.

3.3.2.2 Average Pooling

Average Pooling tương tự như Max Pooling, nhưng thay vì chọn giá trị lớn nhất, nó sẽ tính giá trị trung bình của tất cả các pixel trong vùng pooling.

Cơ chế hoạt động:

- Kích thước cửa sổ: Chọn một vùng nhỏ của feature map (ví dụ: 2×2).
- Stride: Bước di chuyển của cửa sổ qua ảnh. Stride lớn hơn sẽ giảm kích thước của feature map nhanh hơn.

3.3.3. Flatten Layer (Lớp phẳng hóa)

Lớp Flatten là một lớp rất đơn giản nhưng quan trọng trong kiến trúc CNN. Nó có nhiệm vụ chuyển đổi (phẳng hóa) một tensor nhiều chiều thành một vector một chiều để có thể đưa vào các lớp Dense (lớp kết nối đầy đủ) trong mạng neural.

Trong mạng CNN, các lớp Convolution và Pooling sẽ tạo ra các feature map (tensor 3 chiều: chiều rộng, chiều cao, và số lượng kênh). Tuy nhiên, các lớp Dense (Fully Connected Layer) lại yêu cầu đầu vào là một vector một chiều (1D vector). Lớp

Flatten giúp **kết nối** phần feature extraction (trích xuất đặc trưng) của CNN với phần phân loại.

Ví dụ về Flatten Layer: Giả sử đầu vào là một feature map có kích thước (4, 4, 3), trong đó:

- 4 là chiều rộng (width)
- 4 là chiều cao (height)
- 3 là số lượng kênh (channel)

Sau khi qua lớp Flatten, tensor này sẽ được chuyển thành một vector 1D có $4 \times 4 \times 3 = 48$ phần tử.

3.3.4. Fully Connected Layer (Lớp kết nối đầy đủ)

Lớp Fully Connected (Dense Layer), hay còn gọi là lớp Dense, là một trong những lớp phổ biến nhất trong các mô hình mạng neural. Đây là lớp kết nối đầy đủ, nơi tất cả các nơ-ron của lớp hiện tại đều kết nối với tất cả các nơ-ron của lớp tiếp theo. Nó thường được sử dụng ở phần cuối của mô hình CNN để thực hiện quá trình phân loại hoặc dự đoán.

Lớp Fully Connected chịu trách nhiệm tổng hợp các đặc trưng đã được trích xuất bởi các lớp Convolution và Pooling, sau đó đưa ra quyết định dựa trên các đặc trưng đó.

3.3.5. Dropout Layer (Lớp dropout)

Lớp Dropout là một kỹ thuật regularization được sử dụng trong mạng neural nhằm giảm thiểu hiện tượng overfitting. Ý tưởng chính của Dropout là ngẫu nhiên loại bỏ (tắt) một số lượng nơ-ron trong quá trình huấn luyện. Khi loại bỏ nơ-ron, Dropout sẽ ngăn mạng học quá mức từ dữ liệu huấn luyện, giúp mô hình có khả năng tổng quát hóa tốt hơn khi làm việc với dữ liệu mới.

Overfitting xảy ra khi mô hình học quá kỹ các đặc trưng của dữ liệu huấn luyện, dẫn đến việc giảm hiệu suất khi gặp dữ liệu chưa thấy trước đó. Dropout giúp giải quyết vấn đề này bằng cách ép buộc mạng phải học các đặc trưng theo cách không phụ thuộc vào một số nơ-ron nhất định.

Chương 4: TẠO BỘ DỮ LIỆU CHO BÀI TOÁN PHÁT HIỆN TẤN CÔNG XSS

4.1. Các phương pháp để tạo dữ liệu

4.1.1. Sử dụng bộ dữ liệu có sẵn:

Các bộ dữ liệu công khai từ các dự án nghiên cứu:

The XSS Dataset from Kaggle: Một số bộ dữ liệu về XSS có sẵn trên Kaggle, được tạo ra từ các nghiên cứu bảo mật và dự án thực tế.

OWASP Datasets: OWASP cung cấp các bộ dữ liệu bảo mật có liên quan đến các lỗ hổng web, bao gồm cả XSS.

Github Repositories: Nhiều dự án mã nguồn mở trên GitHub có thể có bộ dữ liệu XSS hoặc các tài nguyên liên quan đến bảo mật mà bạn có thể sử dụng.

4.1.2. Tạo dữ liệu từ các ứng dụng và trang web thực tế

Thu thập dữ liệu từ các trang web có khả năng bị tấn công XSS.

Dùng công cụ kiểm tra lỗ hổng bảo mật như Burp Suite hoặc OWASP ZAP để phát hiện các đoạn mã có thể bị tấn công XSS.

Kiểm tra các mẫu dữ liệu từ các dự án mã nguồn mở để thu thập các đoạn mã đã được kiểm tra hoặc dùng trong các thử nghiệm bảo mật.

4.1.3. Sử dụng các công cụ tự động để tạo dữ liệu

Sử dụng các công cụ quét XSS tự động để tìm ra các payload XSS.

XSSer là một công cụ tự động phát hiện các lỗ hổng XSS và có thể giúp tạo ra các mẫu payload.

Burp Suite Extension (XSS Detector): Các tiện ích mở rộng của Burp Suite có thể giúp phát hiện và lưu các mẫu XSS.

4.1.4. Tạo dữ liệu thông qua việc sinh ngẫu nhiên

Sử dụng các script hoặc công cụ để sinh ngẫu nhiên các payload XSS theo một số quy tắc nhất định.

Ví dụ, bạn có thể viết một đoạn Python hoặc JavaScript để tạo ra các đoạn mã XSS ngẫu nhiên:

```
payloads = ["<script>alert('XSS')</script>", "<img src='x'
onerror='alert(1)'>"]
for payload in payloads:
    print(f"<div>{payload}</div>")
```

4.2. Các bước thực hiện cụ thể

Bước 1: Lấy dữ liệu có sẵn từ Kaggle

Truy cập Kaggle

Tìm kiếm “Cross site scripting XSS dataset for Deep learning”

Tải về file “XSS_dataset.csv”

Hình 8: Lấy dữ liệu có sẵn từ Kaggle

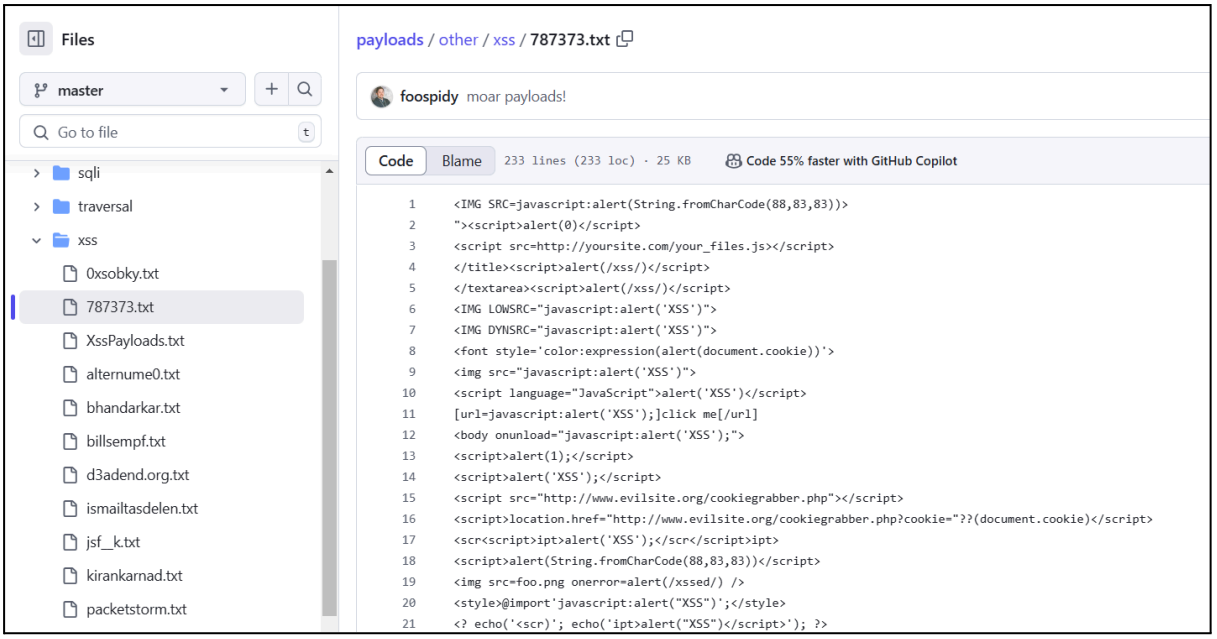
Bước 2: Tiến hành mở rộng bộ dữ liệu hiện có

Tìm một bộ dữ liệu khác từ Github

Nguồn: [https://github.com/foospidy/payloads/tree/master/other/xss]

Chọn các file chứa dữ liệu XSS và gộp thành 1 tệp duy nhất

Đặt tên file “new-xss-data.txt”



Hình 9: Tiến hành mở rộng bộ dữ liệu hiện có

Bước 3: Gộp dữ liệu:

Chuyển đổi file “new-xss-data.txt” thành “new-xss-data.csv”

Tiến hành gộp file “XSS_dataset.csv” và “new-xss-data.csv” thành 1 file duy nhất

Loại bỏ các dòng dữ liệu trùng lắp

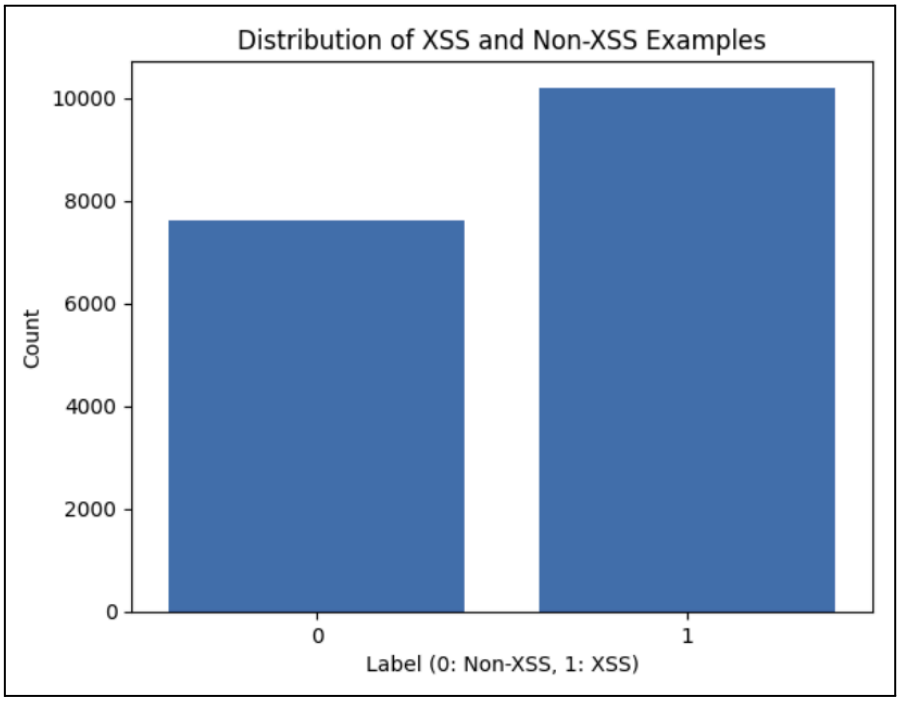
File mới “XSS_dataset_mixed.csv”

```python
import pandas as pd

# Đọc dữ liệu từ hai file CSV
df1 = pd.read_csv('XSS_dataset.csv')
df2 = pd.read_csv('new-xss-data.csv')

# Gộp hai DataFrame lại với nhau
combined_df = pd.concat([df1, df2])

# Loại bỏ các dòng trùng lặp
combined_df = combined_df.drop_duplicates()

# Ghi dữ liệu kết quả vào một file CSV mới
combined_df.to_csv('XSS_dataset_mixed.csv', index=False)

print("Đã gộp file thành công và loại bỏ dòng trùng lặp.")
```

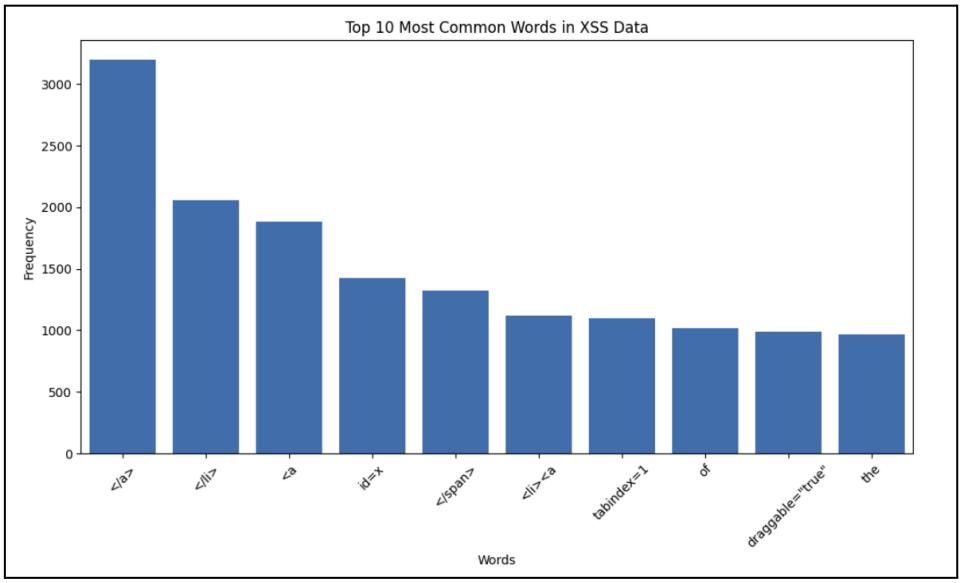
Hình 10 : Code python để gộp file và loại bỏ dữ liệu bị trùng lắp

Kết quả: Từ file dữ liệu ban đầu với 13685 dòng, sau khi thực hiện mở rộng bộ dữ liệu ta được 1 file dữ liệu mới với 17809 dòng.

4.3. Đánh giá dữ liệu



Hình 11:Trực quan hóa phân phối của lớp dữ liệu



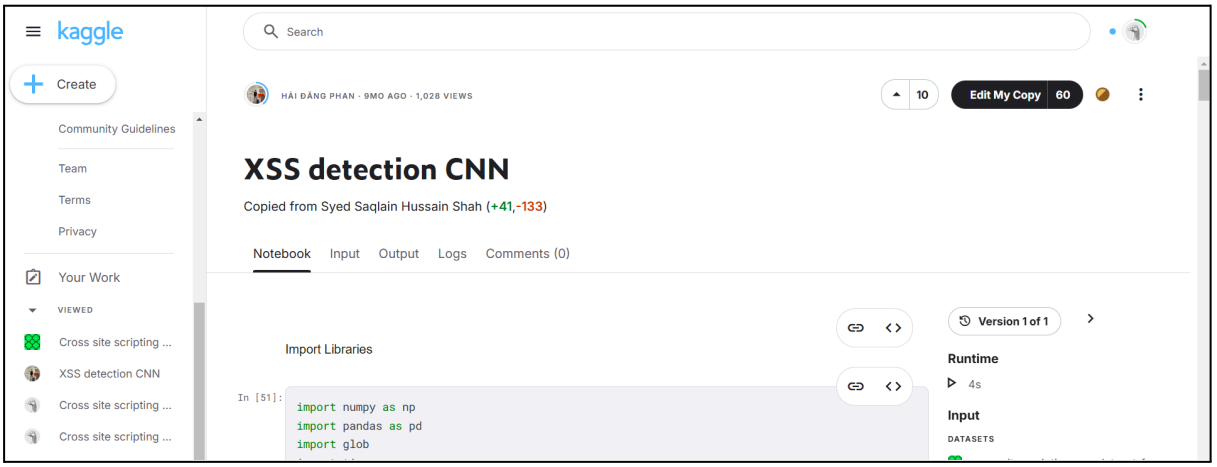
Hình 12:Biểu đồ tần suất các từ/biểu thức được sử dụng trong tấn công XSS

Chương 5: HUẤN LUYỆN MÔ HÌNH VÀ ĐÁNH GIÁ

5.1. Huấn luyện mô hình

Bước 1: Tìm kiếm mô hình huấn luyện có sẵn từ Kaggle

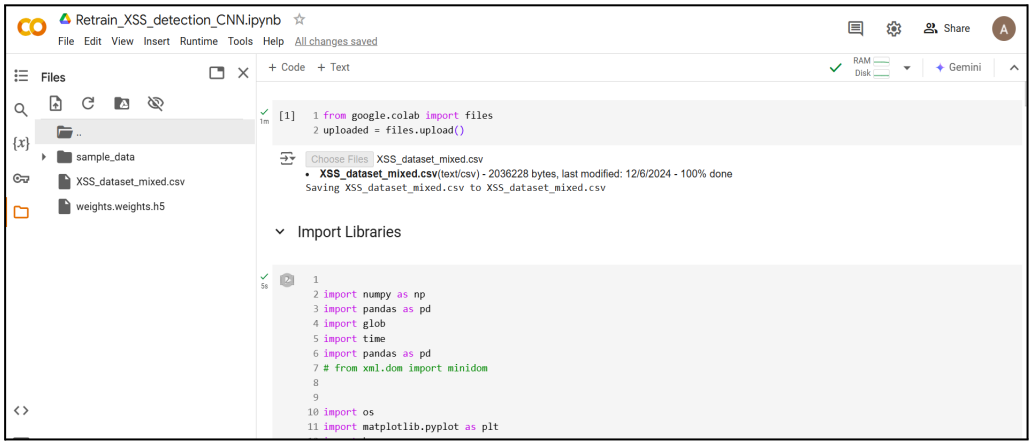
Nguồn: [https://www.kaggle.com/code/hingphan/xss-detection-cnn]



Hình 13: Mô hình huấn luyện có sẵn từ Kaggle

Bước 2: Download code và huấn luyện mô hình với bộ dữ liệu mới

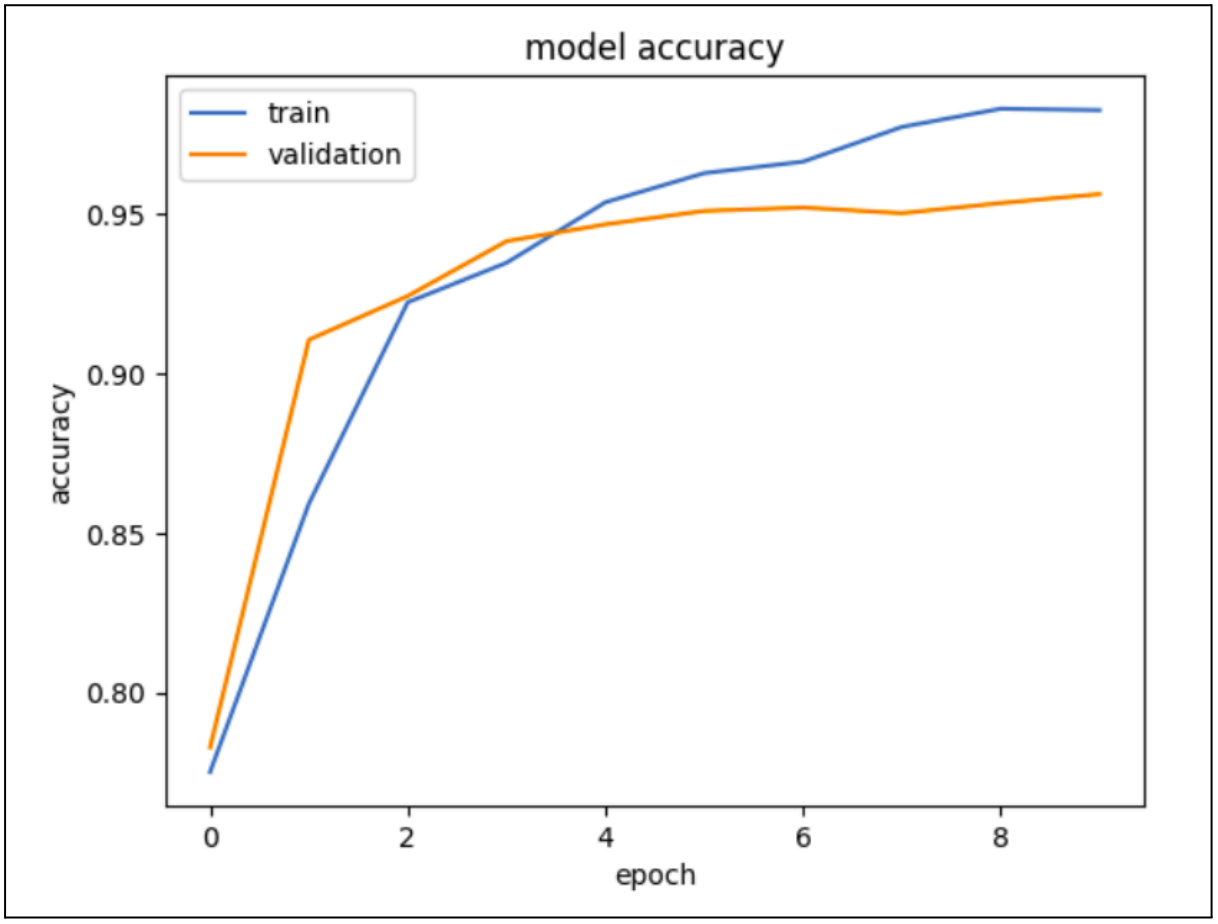
Thực hiện trên Kaggle hoặc Google Colab



Hình 14: Download code và huấn luyện mô hình với bộ dữ liệu mới

5.2. Đánh giá mô hình

5.2.1. Biểu đồ hiển thị quá trình huấn luyện



Hình 15: Biểu đồ hiển thị quá trình huấn luyện

Đường cong cho thấy độ chính xác (accuracy) của tập huấn luyện và tập xác thực (validation) qua các epoch.

Phân tích biểu đồ:

Đường màu xanh (train): Đường này thể hiện độ chính xác trên tập huấn luyện. Chúng ta có thể thấy rằng độ chính xác trên tập huấn luyện tăng đều qua các epoch, cho thấy mô hình đang học tốt từ dữ liệu huấn luyện.

Đường màu cam (validation): Đường này thể hiện độ chính xác trên tập xác thực. Độ chính xác của tập xác thực tăng nhanh đến một thời điểm nào đó rồi bắt đầu giảm nhẹ sau một số epoch. Điều này có thể chỉ ra hiện tượng overfitting, khi mô hình học quá mức từ dữ liệu huấn luyện và không thể tổng quát hóa tốt trên dữ liệu chưa thấy.

Ý nghĩa:

Tăng độ chính xác trên tập huấn luyện: Chứng tỏ mô hình có khả năng học và nắm bắt được các đặc điểm trong dữ liệu huấn luyện.

Độ chính xác của tập xác thực giảm: Gợi ý rằng mô hình có thể đã học quá nhiều chi tiết từ dữ liệu huấn luyện mà không thể áp dụng tốt trên dữ liệu mới, dẫn đến hiện tượng overfitting.

Sự phân kỳ giữa hai đường: Sự chênh lệch giữa độ chính xác của tập huấn luyện và tập xác thực cho thấy mô hình có thể cần được điều chỉnh.

5.2.2. Confusion Matrix (Ma trận nhầm lẫn)

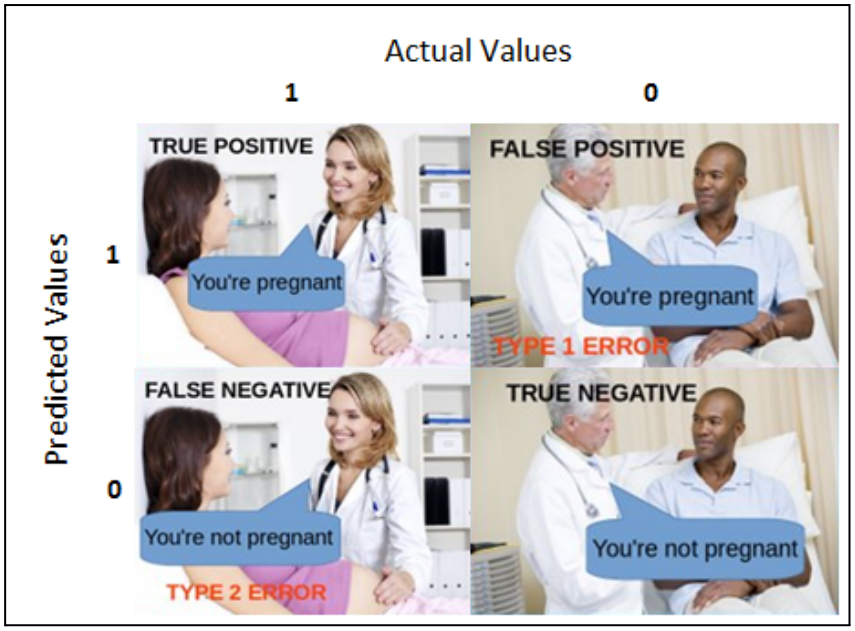
Một ma trận nhầm lẫn cho một bài toán phân loại nhị phân có dạng 2x2, với các thành phần sau:

True Positive (TP): Số lượng mẫu được mô hình dự đoán là dương (positive) và thực tế là dương.

True Negative (TN): Số lượng mẫu được mô hình dự đoán là âm (negative) và thực tế là âm.

False Positive (FP): Số lượng mẫu được mô hình dự đoán là dương nhưng thực tế là âm (còn được gọi là lỗi loại I).

False Negative (FN): Số lượng mẫu được mô hình dự đoán là âm nhưng thực tế là dương (còn được gọi là lỗi loại II).



Hình 16: Confusion Matrix

Ý nghĩa của các thành phần:

Accuracy: Tỉ lệ mẫu dự đoán đúng (TP + TN) trên tổng số mẫu. Công thức là:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision (Độ chính xác): Tỉ lệ mẫu được dự đoán là dương mà thực tế là dương. Công thức là:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall (Độ nhạy): Tỉ lệ mẫu thực tế là dương mà được mô hình dự đoán là dương. Công thức là:

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-Score: Trung bình điều hòa giữa độ chính xác và độ nhạy. Công thức là:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Specificity (Độ đặc hiệu): Tỉ lệ mẫu thực tế là âm mà được dự đoán là âm. Công thức là:

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Confusion Matrix của mô hình:

```
                precision    recall  f1-score   support

           0     0.9797    0.9960    0.9878      1260
           1     0.9966    0.9824    0.9894      1478

    accuracy                         0.9887      2738
   macro avg     0.9881    0.9892    0.9886      2738
weighted avg     0.9888    0.9887    0.9887      2738

[[1255    5]
 [  26 1452]]
      TPR       FPR       FNR
   0.9824    0.0040    0.0176
```

Hình 17: Confusion Matrix

True Positive (TP): 1255 mẫu được dự đoán là lớp 0 và đúng là lớp 0.
True Negative (TN): 1452 mẫu được dự đoán là lớp 1 và đúng là lớp 1.
False Positive (FP): 5 mẫu được dự đoán là lớp 0 nhưng thực tế là lớp 1.
False Negative (FN): 26 mẫu được dự đoán là lớp 1 nhưng thực tế là lớp 0.
TPR (Recall): 0.9824 (98.24%) - Tỷ lệ mẫu thực tế là lớp 1 mà được dự đoán đúng.
FPR (False Positive Rate): 0.0040 (0.40%) - Tỷ lệ mẫu thực tế là lớp 0 nhưng được dự đoán là lớp 1 rất thấp, cho thấy mô hình không bị nhiều lỗi loại I.
FNR (False Negative Rate): 0.0176 (1.76%) - Tỷ lệ mẫu thực tế là lớp 1 nhưng được dự đoán là lớp 0 cũng thấp, cho thấy mô hình không bỏ sót nhiều mẫu lớp 1.

5.2.3. Đáng giá chung:

Accuracy: 0.9887 (98.87%) — Mô hình có độ chính xác rất cao, cho thấy nó phân loại đúng đến 98.87% số mẫu trên tổng số mẫu kiểm tra.

Macro average: Precision = 0.9881, Recall = 0.9892, F1-score = 0.9886 - Tính trung bình đơn giản giữa các lớp cho thấy mô hình vẫn giữ được hiệu suất tốt trên cả hai lớp.

Weighted average: Precision = 0.9888, Recall = 0.9887, F1-score = 0.9887 — Tính trung bình có trọng số, cân nhắc đến số lượng mẫu của từng lớp, cho thấy mô hình cũng đạt hiệu suất cao trên cả hai lớp khi xét đến phân phối lớp.

5.2.4. Đánh giá tổng quát

Mô hình hoạt động rất tốt với độ chính xác và các chỉ số khác đều ở mức cao. Tỷ lệ lỗi rất thấp, đặc biệt là tỉ lệ sai loại I (FPR) và tỉ lệ sai loại II (FNR). Mô hình có khả năng phân loại chính xác cả hai lớp mà không bị thiên lệch nghiêng về lớp nào. Đây là một mô hình rất hiệu quả cho bài toán phân loại tấn công XSS.

Tài liệu tham khảo

1. Dataset - Link: https://github.com/foospidy/payloads/tree/master/other/xss (Truy cập lần cuối: 01/12/2024)
2. Cross site scripting XSS dataset for Deep learning - Link:https://www.kaggle.com/datasets/syedsaqlainhussain/cross-site-scripting-xss-dataset-for-deep-learning (Truy cập lần cuối: 01/12/2024)

3. XSSClassifier: XSS Attack Detection Approach Based on Machine Learning Classifier in Python - Link https://github.com/obarrera/ML-XSS-Detection (Truy cập lần cuối: 01/12/2024)

4. Web Applications vulnerabilities and threats: statistics for 2019 - Link:https://global.ptsecurity.com/analytics/web-vulnerabilities-2020 (Truy cập lần cuối: 01/12/2024)

5. XSS attack detection using CNN - Link:https://www.kaggle.com/code/hingphan/xss-detection-cnn (Truy cập lần cuối: 01/12/2024)