| BAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  A red and yellow circle with a book and key  Description automatically generated |
| --- |
| **PHÁT TRIỂN MÔ-ĐUN TƯỜNG LỬA ỨNG DỤNG**  **WEB TÍCH HỢP HỌC MÁY CHO NGINX** |
| *Nhóm học viên thực hiện:*  **Nguyễn Lê Quốc Đạt - CHAT3P03**  **Hoàng Anh Tuấn – CHAT3P15**    *Người hướng dẫn*:  **TS. Nguyễn An Khương**  Đại học Bách Khoa TP. HCM |
| TP. HCM, 2024 |

LỜI CẢM ƠN

……………………………………………………………………………………………...

MỤC LỤC

Tóm tắt đề tài

[Chương 1. TỔNG QUAN ĐỀ TÀI 1](#_1fob9te)

[1.1. Giới thiệu về đề tài 2](#_3znysh7)

[1.2. Mục tiêu của đề tài 2](#_2et92p0)

[1.3. Đối tượng, phạm vi nghiên cứu 3](#_tyjcwt)

[1.3.1. Đối tượng nghiên cứu: 3](#_3dy6vkm)

[1.4. Tính mới và tính ứng dụng của đề tài 3](#_1t3h5sf)

[Chương 2. CƠ SỞ LÝ THUYẾT 4](#_4d34og8)

[2.1. NGINX 4](#_2s8eyo1)

[2.1.1. Giới thiệu về NGINX 4](#_17dp8vu)

[2.1.2. Nguyên lý hoạt động của NGINX 4](#_3rdcrjn)

[2.1.3. Các tính năng chính của NGINX 4](#_26in1rg)

[2.1.4. Phương pháp tạo và biên dịch mô-đun trong NGINX 5](#_35nkun2)

[2.2. Reverse Proxy 6](#_1ksv4uv)

[2.2.1. Khái niệm Reverse proxy 6](#_44sinio)

[2.2.2. Mô hình Reverse proxy 6](#_2jxsxqh)

[2.3. Web Application Firewall 6](#_3j2qqm3)

[2.4. ModSecurity 7](#_4i7ojhp)

[2.4.1. Giới thiệu về Modsecurity 7](#_2xcytpi)

[2.4.2. ModSecurity Core Rule Set (CRS): 7](#_1ci93xb)

[2.5. Học máy (Machine Learning) 8](#_3whwml4)

[2.5.1. Học sâu (Deep Learning) 9](#_qsh70q)

[2.5.2. Neural Network (mạng nơ-ron) 9](#_1pxezwc)

[2.6. Natural Language Processing (NLP) 14](#_2p2csry)

[2.6.1. Giới thiệu về NLP 14](#_147n2zr)

[2.6.2. Ứng dụng của NLP 14](#_3o7alnk)

[Chương 3. THIẾT KẾ HỆ THỐNG 15](#_23ckvvd)

[3.1. Tổng quan về hệ thống 15](#_ihv636)

[3.2. Tiền xử lí dữ liệu 16](#_1hmsyys)

[3.3. Triển khai các mô hình 19](#_vx1227)

[3.3.1. Kiến trúc mô hình Birectional-LSTM 19](#_3fwokq0)

[3.3.2. Tổng quan mô hình Bidirectional-LSTM: 21](#_1v1yuxt)

[3.4. Word Embedding 21](#_2u6wntf)

[3.5. Chi tiết quy trình hoạt động của hệ thống 24](#_19c6y18)

[3.5.1. Yêu cầu được gửi từ máy tấn công đến Reverse proxy -WAF 24](#_3tbugp1)

[3.5.2. Yêu cầu được xử lý tại máy Reverse proxy – WAF 24](#_28h4qwu)

[3.5.3. Yêu cầu được gửi từ máy Reverse proxy – WAF đến máy chủ web 25](#_nmf14n)

[Chương 4. HIỆN THỰC HỆ THỐNG VÀ THỰC NGHIỆM 27](#_37m2jsg)

[4.1. Hiện thực hệ thống 27](#_1mrcu09)

[4.1.1. Mô hình triển khai 27](#_46r0co2)

[4.1.2. Cấu hình các máy thực nghiệm 27](#_2lwamvv)

[4.1.3. Dữ liệu và tiền xử lí 28](#_111kx3o)

[4.1.4. Cài đặt 29](#_206ipza)

[4.1.5. Tiền xử lý dữ liệu trong NGINX 29](#_4k668n3)

[4.1.6. Triển khai ModSecurity vào NGINX 40](#_1rvwp1q)

[4.1.7. Triển khai mô hình học máy vào NGINX 45](#_4bvk7pj)

[4.1.8. Triển khai các mô hình khác 53](#_1664s55)

[4.2. Thực nghiệm 55](#_1jlao46)

[4.2.1. Tiêu chí đánh giá 55](#_43ky6rz)

[4.2.2. Kết quả thực nghiệm trên Python 55](#_2iq8gzs)

[4.2.3. So sánh kết quả thực nghiệm với các mô hình khác 56](#_3hv69ve)

[4.3. Chạy thực nghiệm trên NGINX 56](#_4h042r0)

[4.3.1. Trường hợp WAF tích hợp Modsecuirty và mô hình Bidirectional-LSTM 56](#_1302m92)

[4.3.2. Kết luận 58](#_upglbi)

[Chương 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 59](#_1tuee74)

[5.1. Kết luận 59](#_4du1wux)

[5.2. Thuận lợi và khó khăn 59](#_2szc72q)

[5.3. Kết quả đạt được 60](#_184mhaj)

[5.4. Hướng phát triển 61](#_3s49zyc)

[References 63](#_279ka65)

DANH MỤC HÌNH

[Hình 1: Các chức năng chính của NGINX 6](#_lnxbz9)

[Hình 2: Mô hình hoạt động của Reverse proxy 8](#_z337ya)

[Hình 3: Mô hình sử dụng Web Application Firewall 9](#_1y810tw)

[Hình 4: Mối quan hệ giữa AI, Machine Learning và Deep Learning. 11](#_2bn6wsx)

[Hình 5: Lịch sử deep learning 12](#_3as4poj)

[Hình 6: Mạng nơ-ron 15](#_49x2ik5)

[Hình 7: Mô hình tổng quan về hệ thống đề xuất 19](#_32hioqz)

[Hình 8: Mô hình Bidirectional LSTM 25](#_4f1mdlm)

[Hình 9: Phân tách đường dẫn dựa vào những kí tự đặc biệt 39](#_2zbgiuw)

[Hình 10: Phân tách dữ liệu dựa vào những kí tự đặc biệt 40](#_1egqt2p)

[Hình 11: Đường dẫn được đưa về dạng thống nhất 42](#_3ygebqi)

[Hình 12: Chuyển đường dẫn về dạng mảng số 43](#_2dlolyb)

[Hình 13: Dữ liệu được chuyển sang dạng thống nhất 47](#_sqyw64)

[Hình 14: Dữ liệu được chuyển sang dạng mảng số 48](#_3cqmetx)

[Hình 15: Mô hình được tải thành công vào NGINX 62](#_2r0uhxc)

[Hình 16: Mô hình Simple model 62](#_3q5sasy)

[Hình 17: Mô hình LSTM 63](#_25b2l0r)

[Hình 18: Mô hình CNN for Text 63](#_kgcv8k)

[Hình 19: Mô hình CNN-LSTM 64](#_34g0dwd)

DANH MỤC BẢNG

[BẢNG 1: CÁC TỪ KHOÁ ĐƯỢC ĐỊNH NGHĨA 22](#_41mghml)

[BẢNG 2: BẢNG ÁNH XẠ CHUYỂN ĐỔI TỪ 23](#_2grqrue)

[BẢNG 3: SỐ LƯỢNG BỘ DỮ LIỆU SAU KHI LỌC 33](#_3l18frh)

[BẢNG 4: KẾT QUẢ DỰ ĐOÁN CỦA MÔ HÌNH TRÊN CÁC BỘ DỮ LIỆU 66](#_xvir7l)

DANH MỤC TỪ VIẾT TẮT

WAF Web Application Firewall

CRS Core Rule Set

NLP Natural Language Processing

XSS Cross Site Scripting

CBOW Continuous Bag-of-Words

NLP Natural Language Processing

LSTM Long Short-Term Memory

CNN Convolutional Neural Network

RNN Recurrent Neural Network

DANH MỤC TỪ TẠM DỊCH

Học máy Machine Learning

Học sâu Deep Learning

Mô hình Model

Mô-đun Module

Nơ ron Neuron

Tường lửa Firewall

Phiên Session

TÓM TẮT ĐỀ TÀI

Các cuộc tấn công mạng đang ngày càng phổ biến, các cuộc tấn công diễn ra dưới rất nhiều kỹ thuật và hình thức khác nhau, cách thức tấn công cũng thay đổi dần nên các hệ thống phát hiện xâm nhập dựa vào dấu hiệu là không hiệu quả, thay vào đó, các hệ thống phát hiện xâm nhập sử dụng học máy thường chứng tỏ ưu thế hơn trong việc phát hiện các cuộc tấn công mới. Để đạt được độ chính xác cao, cần có tập dữ liệu lớn và có thể bao gồm nhiều cuộc tấn công khác nhau. Trong nghiên cứu này, nhóm tác giả đề xuất một hệ thống phát hiện xâm nhập dựa trên học máy, bên cạnh đó, chính hệ thống này sẽ được tích hợp vào NGINX như một module. Mục đích của việc này nhằm tăng được khả năng bảo vệ dành cho tường lửa, nâng cao khả năng phân biệt và ngăn chặn những yêu cầu độc hại.

# TỔNG QUAN ĐỀ TÀI

## Giới thiệu về đề tài

Trong khoảng thời gian 5 năm trở lại gần đây, ứng dụng doanh nghiệp ngày càng có xu hướng chuyển đổi từ ứng dụng desktop sang ứng dụng web. Các cuộc tấn công mạng nhằm vào các máy chủ web và ứng dụng web vẫn đang là một trong những vấn đề chủ chốt cần được chú trọng trong một doanh nghiệp đang sử dụng công nghệ. Các cuộc tấn công vào các ứng dụng web ngày càng trở nên phức tạp và khó bị phát hiện hơn. Do đó, việc triển khai các biện pháp để bảo vệ các ứng dụng web nhằm tránh khỏi các cuộc tấn công là một công việc hết sức cần thiết. Một trong những phương pháp được sử dụng phổ biến nhất hiện nay là Web Application Firewall (WAF).

Các WAF thông thường hoạt động bằng việc sử dụng chữ ký (signature), chúng nhận biết cuộc tấn công thông qua dấu hiệu riêng biệt của nó, điều này dựa trên một cơ sở dữ liệu có sẵn nên nó yêu cầu cơ sở dữ liệu lớn. Bên cạnh đó việc phụ thuộc vào cơ sở dữ liệu và các logic và quy tắc được mã hóa cứng (sử dụng lập trình truyền thống) cũng tồn tại những hạn chế nhất định.

Trong những thập kỷ gần đây, trí tuệ nhân tạo đã trở thành một cuộc cách mạng khoa học và đạt được ưu thế trong việc làm chủ công việc mà con người làm. Các nhà nghiên cứu và chuyên gia bảo mật thông tin đã đặc biệt chuyển sang khai thác khả năng của trí tuệ nhân tạo để phát hiện và chống lại các cuộc tấn công.

Với mong muốn có thể tận dụng tối đa những gì đã có nhưng vẫn có thể nắm bắt được xu hướng hiện tại, trong đề tài này nhóm tác giả sẽ thực hiện triển khai một WAF kết hợp giữa một WAF thông thường và machine learning. Mục đích của việc kết hợp này là có thể đưa ra một WAF có thể hoạt động hiệu quả, không chỉ phát hiện được những cuộc tấn công đã có chữ kí mà còn có khả năng phát triển xa hơn, phát hiện được những cuộc tấn công mới.

## Mục tiêu của đề tài

Mục tiêu của nhóm hướng đến là có thể để có thể xây dựng một mô-đun tích hợp học máy có thể ngăn chặn được các cuộc tấn công Web sử dụng Web Application Firewall như thông thường. Cụ thể, mục tiêu của đề tài như sau:

* Nghiên cứu và sử dụng được một Web Application Firewall thông thường, cụ thể trong đề tài này nhóm sẽ sử dụng ModSecurity.
* Thực hiện tích hợp và triển khai ModSecurity vào Nginx.
* Hiểu được những hạn chế còn tồn tại trong một Web Application Firewall thông thường, nhóm sẽ thực hiện tích hợp các phương pháp hiện đại, cụ thể ở đây là Machine Learning để có thể khắc phục các hạn chế còn tồn đọng.
* Đề xuất một phương pháp không những phát hiện được những cuộc tấn công đã có (những cuộc tấn công đã có signature và được lưu trong database) mà còn có thể phát hiện được những cách tấn công mới.
* Từ phương pháp đã đề xuất, nhóm phát triển một mô-đun cho Nginx có khả năng phát hiện được những cách tấn công web hoạt động.

## Đối tượng, phạm vi nghiên cứu

### Đối tượng nghiên cứu:

* Công cụ Nginx và Reverse Proxy
* Cách viết và biên dịch (compile) một mô-đun Nginx
* Web Application Firewall (ModSecurity)
* Các phương pháp học máy

## Tính mới và tính ứng dụng của đề tài

Trong mục tiêu đề tài của nhóm, chúng em sẽ phát triển một mô đun tường lửa có kết hợp giữa mô hình Web Application Firewall truyền thống và mô hình máy học, đồng thời, đưa ra phương án giải quyết khả quan đảm bảo độ chính xác cao nhưng vẫn giữ được tốc độ xử lí của các hành vi. Nhóm thực hiện nghiên cứu các công bố về vấn đề tương tự, đưa ra cái nhìn tổng thể, sau đó áp dụng vào mô hình hiện tại. Bên cạnh đó, vẫn đảm bảo được tính thực tiễn, có thể áp dụng và dùng như một ứng dụng thực tế.

# CƠ SỞ LÝ THUYẾT

## NGINX

### Giới thiệu về NGINX

NGINX là một web server và reverse proxy server miễn phí, mã nguồn mở. Nó được tạo ra để giải quyết các vấn đề liên quan đến việc quản lý các kết nối đồng thời và hiệu suất truyền tải trên các ứng dụng web tải nặng.

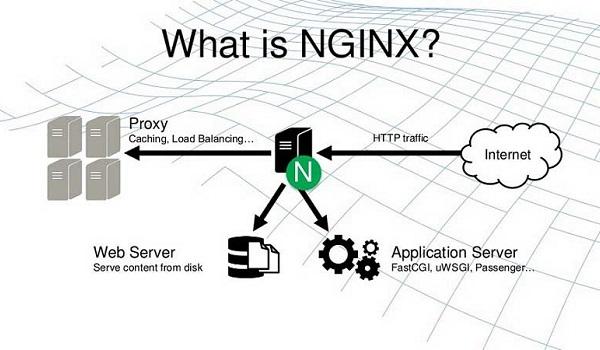
### Nguyên lý hoạt động của NGINX

NGINX hoạt động theo mô hình xử lý bất đồng bộ (asynchronous) và sử dụng cơ chế event-driven để quản lý các kết nối đồng thời. Khi có một yêu cầu truy cập đến server, nginx sẽ tạo ra một tiến trình xử lý (worker process) để xử lý yêu cầu đó..

### Các tính năng chính của NGINX

Nginx có nhiều tính năng chính, bao gồm:

* Web server: Nginx có khả năng xử lý các yêu cầu HTTP và HTTPS truy cập đến server, hỗ trợ nhiều tính năng như server-side includes, reverse proxy, caching, SSL/TLS, và nén dữ liệu để tăng tốc độ truy cập.
* Reverse proxy server: Nginx có thể được sử dụng làm reverse proxy server, giúp chuyển tiếp yêu cầu từ client đến server hoặc giữa các server khác nhau để tăng tốc độ truy cập và cải thiện tính sẵn sàng của hệ thống.
* Load balancing: Nginx cung cấp tính năng cân bằng tải giữa các server, giúp phân phối tải truy cập đến các server khác nhau một cách công bằng và giảm thiểu tình trạng quá tải trên các server.
* High availability: Với tính năng load balancing và chuyển tiếp yêu cầu, Nginx có khả năng tăng tính sẵn sàng của hệ thống bằng cách đảm bảo rằng yêu cầu sẽ được xử lý bởi server khác nếu server chính gặp sự cố.
* Performance: Nginx được thiết kế để xử lý hàng ngàn kết nối đồng thời một cách hiệu quả, giảm thiểu tình trạng blocking và cải thiện hiệu suất truy cập.
* Extensibility: Nginx có khả năng mở rộng thông qua các module để hỗ trợ các tính năng phức tạp và đáp ứng nhu cầu của người dùng.
* Security: Nginx cung cấp tính năng bảo mật như hạn chế truy cập, chống tấn công DDoS, bảo vệ khỏi các cuộc tấn công SQL injection và XSS.



Hình 1: Các chức năng chính của NGINX

Tóm lại, Nginx là một web server và reverse proxy server đa năng và mạnh mẽ, có thể cung cấp nhiều tính năng để giải quyết các vấn đề liên quan đến hiệu suất, tính sẵn sàng, bảo mật và mở rộng của hệ thống web.

### Phương pháp tạo và biên dịch mô-đun trong NGINX

Tùy vào mục đích của mô-đun mà mỗi mô-đun sẽ có những tệp mã nguồn và độ phức tạp khác nhau. Tuy nhiên thì để viết một mô-đun và tích hợp vào NGINX thì cần có tối thiểu 2 tập tin cơ bản:

* File code C: chứa code chính được sử dụng để biên dịch mô-đun.
* File config: Mọi module cần một tệp cấu hình là một Bourne shell file hướng dẫn hệ thống về cách xây dựng một module NGINX. Tùy vào việc compile module dynamic hay static mà tập tin config sẽ khác nhau.

Các bước biên dịch một mô-đun động trong NGINX:

* Biên dịch mô-đun băng cách chạy tập tin configure trước tiên với đối số --with-compat, tạo môi trường xây dựng tiêu chuẩn được hỗ trợ bởi Nginx Open Source và Nginx Plus:

./configure --with-compat --add-dynamic-module=../nginx-hello-world-module

* Tiếp theo chạy câu lệnh bên dưới để xây dựng mô-đun:

make module

## Reverse Proxy

### Khái niệm Reverse proxy

Reverse proxy là một server trung gian nằm giữa client và server, chuyển tiếp các yêu cầu từ client đến server và trả về kết quả từ server cho client. Trong kiến trúc này, client không kết nối trực tiếp với server, mà thay vào đó kết nối với reverse proxy và yêu cầu của client sẽ được chuyển tiếp đến server bên trong mạng nội bộ.

Reverse proxy thường được sử dụng trong các kiến trúc phân tán, mạng lưới và các hệ thống web quy mô lớn. Nginx là một ví dụ phổ biến về reverse proxy server..

### Mô hình Reverse proxy

Bên dưới là một mô hình trong thực tế sử dụng reverse proxy. Reverse proxy sẽ đứng trước máy chủ web, nhận mọi lưu lượng từ Internet đi vào, sau đó thực hiện chuyển những lưu lượng này đến máy chủ web.

Diagram

Description automatically generated

Hình 2: Mô hình hoạt động của Reverse proxy

## Web Application Firewall

Web Application Firewall (WAF) là một loại phần mềm bảo mật đặt tại cổng vào (front-end) của một ứng dụng web, giúp bảo vệ ứng dụng khỏi các cuộc tấn công mạng như tấn công SQL Injection, Cross-site Scripting (XSS), Cross-site Request Forgery (CSRF), và các cuộc tấn công khác.

WAF hoạt động như một bộ lọc tùy chỉnh cho các yêu cầu và phản hồi HTTP/S, chặn các yêu cầu độc hại, và giám sát các hoạt động của ứng dụng web để phát hiện các hành vi bất thường. WAF có thể được cấu hình để thực hiện các chức năng như chặn truy cập từ các địa chỉ IP đáng ngờ, theo dõi các kết nối của người dùng, phân tích độ tin cậy của dữ liệu nhập vào và đưa ra các cảnh báo về các hành vi tấn công mạng.

WAF là một phần quan trọng trong bảo mật ứng dụng web, giúp giảm thiểu rủi ro tấn công và giữ an toàn cho dữ liệu và thông tin của người dùng. Tuy nhiên, việc triển khai WAF cũng cần cân nhắc đến các yếu tố khác như hiệu suất, tương thích và chi phí.

Diagram

Description automatically generated

Hình 3: Mô hình sử dụng Web Application Firewall

## ModSecurity

### Giới thiệu về Modsecurity

ModSecurity (còn được gọi là ModSec) là một Web Application Firewall, nó được triển khai để bảo vệ trang web và ứng dụng ngăn cản nhiều cuộc tấn công phổ biến. Nó thiết lập và ngăn chặn các yêu cầu tấn công trước khi chúng đến được ứng dụng web.

Ban đầu, ModSecurity là một một mô-đun được Ivan Ristic phát triển dành cho máy chủ web Apache và theo thời gian, nó đã phát triển thành một tường lửa ứng dụng web chính thức với sự hỗ trợ cho các nền tảng khác nhau, bao gồm Apache, Nginx và IIS.

Nó có một bộ công cụ quy tắc (rule engine) để thực hiện các hành động từ đơn giản đến phức tạp, đi kèm với nó là Core Rule Set (CRS). Phổ biến nhất trong số đó là OWASP ModSecurity Core Rule Set.

### ModSecurity Core Rule Set (CRS):

CRS là một bộ các quy tắc phát hiện tấn công chung có thể sử dụng cho Modsecurity hoặc tưởng lửa tương thích. CRS có mục đích nhằm bảo vệ ứng dụng web khỏi một loạt các cuộc tấn công, bao gồm cả Top 10 OWASP. CRS cung cấp khả năng chống lại nhiều cuộc tấn công hiện nay phổ biến, bao gồm SQL Injection, Cross-site Scripting, Local File Inclusion, …

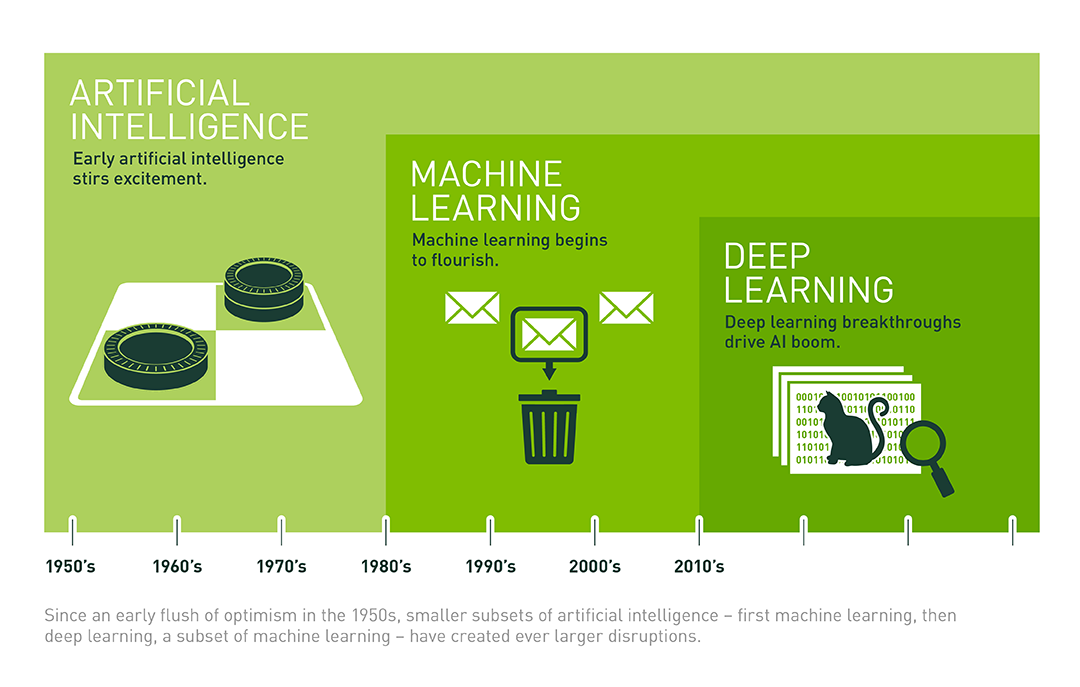
## Học máy (Machine Learning)

Học máy (Machine Learning) là một lĩnh vực của trí tuệ nhân tạo (Artificial Intelligence - AI) nghiên cứu về cách xây dựng các thuật toán và mô hình để máy tính có thể học hỏi và cải thiện kết quả dự đoán của chúng một cách tự động.

Học máy cho phép máy tính tự động học hỏi từ dữ liệu đầu vào và chuyển đổi dữ liệu đó thành các mô hình dự đoán. Để làm được điều này, máy tính sử dụng các kỹ thuật và thuật toán phân tích dữ liệu như regression, clustering, decision trees, neural networks, support vector machines, và deep learning.

Ứng dụng của học máy là rất đa dạng và được sử dụng trong nhiều lĩnh vực như xử lý ngôn ngữ tự nhiên, nhận dạng hình ảnh, phân loại văn bản, thị giác máy tính, điều khiển tự động, dự đoán tín dụng, thị trường tài chính và nhiều lĩnh vực khác.

Học máy đang được sử dụng rộng rãi trong các ứng dụng thực tế và được xem là một công cụ quan trọng trong việc giải quyết các bài toán phức tạp.



Hình 4: Mối quan hệ giữa AI, Machine Learning và Deep Learning.

### Học sâu (Deep Learning)

Học sâu (Deep learning) là một phần của lĩnh vực học máy, sử dụng các mạng nơ-ron nhân tạo (artificial neural networks) để học và hiểu dữ liệu. Khác với các mô hình học máy truyền thống, học sâu có khả năng học và giải quyết các bài toán phức tạp, đặc biệt là trong các lĩnh vực như thị giác máy tính (computer vision) và xử lý ngôn ngữ tự nhiên (natural language processing).

Học sâu được gọi là "sâu" bởi vì các mô hình neural networks được sử dụng có rất nhiều lớp (layers) khác nhau, từ đó giúp nó có khả năng học được các mẫu phức tạp hơn. Các mô hình học sâu có thể được huấn luyện để phát hiện và phân tích các đặc trưng ẩn trong dữ liệu, sau đó áp dụng các phương pháp học tăng cường để cải thiện kết quả dự đoán.

Deep learning được nhắc đến nhiều trong những năm gần đây, nhưng những nền tảng cơ bản đã xuất hiện từ rất lâu.

Cụ thể như hình dưới đây:

Timeline

Description automatically generated

Hình 5: Lịch sử deep learning

### Neural Network (mạng nơ-ron)

Mạng neural (neural network) là một mô hình toán học được lấy cảm hứng từ cấu trúc và hoạt động của hệ thống thần kinh của con người. Mạng neural là một hệ thống tính toán đồng thời trên một tập hợp các "nơ-ron" nhân tạo (artificial neurons) để giải quyết các bài toán phức tạp trong lĩnh vực học máy và trí tuệ nhân tạo.

Mỗi nơ-ron trong mạng neural nhận đầu vào từ các nơ-ron khác, thực hiện một phép tính toán đơn giản, sau đó chuyển kết quả cho các nơ-ron khác. Kết quả cuối cùng được tính toán từ các nơ-ron cuối cùng của mạng. Mạng neural có thể được huấn luyện để tự động học các mối quan hệ và đặc trưng ẩn trong dữ liệu đầu vào, và từ đó áp dụng để giải quyết các bài toán phức tạp như nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên, và dự báo.

#### Tầm quan trọng của mạng nơ-ron

Tầm quan trọng của neural network nằm ở khả năng của chúng trong việc giải quyết các bài toán phức tạp và khó khăn trong các lĩnh vực khác nhau như nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên, dự báo tài chính, và nhiều ứng dụng khác. Mạng nơ-ron có thể hiểu rõ dữ liệu phi cấu trúc và đưa ra các nhận xét chung mà không cần đào tạo cụ thể. Tầm quan trọng của neural network có thể được tóm tắt như sau:

* Xử lý dữ liệu phức tạp: Neural network có thể học và xử lý các dữ liệu phức tạp mà các phương pháp truyền thống khác không thể làm được.
* Khả năng học tập: Neural network có khả năng học tập và cải thiện hiệu suất theo thời gian.
* Giải quyết các bài toán khó khăn: Neural network có thể giải quyết các bài toán phức tạp trong nhiều lĩnh vực như nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên, dự báo tài chính, và nhiều ứng dụng khác.
* Độ chính xác cao: Neural network có khả năng đạt được độ chính xác cao trong các bài toán dự đoán và phân loại.
* Đa dạng về kiến trúc: Neural network có nhiều kiến trúc khác nhau cho phép tùy chỉnh để phù hợp với các bài toán cụ thể.

Tóm lại, tầm quan trọng của neural network nằm ở khả năng của chúng trong việc giải quyết các bài toán phức tạp và khó khăn, đạt được độ chính xác cao và có thể được tùy chỉnh cho các bài toán cụ thể.

#### Công dụng của mạng nơ-ron

Mạng neural (Neural network) là một công nghệ quan trọng trong lĩnh vực machine learning và đã được sử dụng rộng rãi trong nhiều ứng dụng khác nhau. Sau đây là một số công dụng của neural network:

* Nhận dạng hình ảnh: Neural network có thể được sử dụng để nhận dạng hình ảnh, từ việc phát hiện khuôn mặt đến phân tích ảnh y tế để chẩn đoán bệnh.
* Xử lý ngôn ngữ tự nhiên: Neural network có thể được sử dụng để xử lý ngôn ngữ tự nhiên, từ việc phân loại tin nhắn thư rác đến tạo ra các trả lời tự động trong ứng dụng chatbot.
* Dự báo và dự đoán: Neural network có thể được sử dụng để dự báo và dự đoán trong nhiều lĩnh vực khác nhau, từ dự đoán giá chứng khoán đến dự báo thời tiết.
* Tự động hóa: Neural network có thể được sử dụng để tự động hóa nhiều tác vụ, từ việc tìm kiếm thông tin trên mạng đến phân tích dữ liệu cho việc đưa ra quyết định.
* Robotics: Neural network có thể được sử dụng để điều khiển robot và làm cho chúng có thể học và thích nghi với môi trường xung quanh.
* Tối ưu hóa: Neural network có thể được sử dụng để tối ưu hóa các quá trình sản xuất và hoạt động, từ việc quản lý kho hàng đến tối ưu hóa hệ thống cung ứng.

Trên đây chỉ là một vài ví dụ về các ứng dụng của neural network. Với khả năng học tập và xử lý dữ liệu phức tạp, neural network đang trở thành công nghệ quan trọng trong nhiều lĩnh vực và được sử dụng rộng rãi trong các ứng dụng khác nhau.

#### Cách thức hoạt động của mạng nơ-ron

Mạng neural (Neural network) hoạt động dựa trên các phép tính số học được thực hiện trên một số lượng lớn các đơn vị tính toán, được gọi là neuron. Mạng neural bao gồm nhiều lớp neuron liên kết với nhau để thực hiện các tác vụ khác nhau.

Mỗi neuron nhận đầu vào từ các neuron trong lớp trước đó, thực hiện các phép tính và truyền đầu ra đến các neuron trong lớp tiếp theo. Các phép tính này thường bao gồm tổng trọng số các đầu vào và áp dụng một hàm kích hoạt để tính toán đầu ra của neuron.

Khi một tác vụ được đưa vào mạng neural, dữ liệu đầu vào sẽ được đưa vào lớp đầu tiên của mạng neural và truyền qua các lớp neuron để đưa ra kết quả cuối cùng. Mạng neural sẽ sử dụng các kết quả trước đó để tinh chỉnh các trọng số giữa các neuron trong quá trình huấn luyện, từ đó cải thiện khả năng dự đoán và đưa ra kết quả chính xác hơn.

Quá trình này được gọi là "backpropagation" và là một phương pháp quan trọng để huấn luyện mạng neural. Trong quá trình huấn luyện, các đầu vào và đầu ra mong muốn đã biết trước được cung cấp cho mạng neural để tinh chỉnh các trọng số và cấu trúc mạng neural để cải thiện khả năng dự đoán.

Tóm lại, mạng neural hoạt động bằng cách kết nối các lớp neuron để thực hiện các phép tính số học trên các đầu vào và đưa ra kết quả dự đoán. Quá trình huấn luyện mạng neural bao gồm cập nhật các trọng số giữa các neuron để tối ưu hóa kết quả dự đoán.

Diagram, schematic

Description automatically generated

Hình 6: Mạng nơ-ron

#### Những loại mạng nơ-ron

Có nhiều loại mạng neural khác nhau, mỗi loại được thiết kế để giải quyết một loại vấn đề khác nhau. Dưới đây là một số loại mạng neural phổ biến:

* **Mạng neural truyền thẳng (Feedforward Neural Network)**: Đây là loại mạng neural cơ bản nhất, trong đó dữ liệu chỉ truyền một chiều từ đầu vào đến đầu ra mà không có phản hồi. Mạng neural truyền thẳng thường được sử dụng trong các bài toán phân loại và dự đoán.
* **Mạng neural tích chập (Convolutional Neural Network - CNN)**: Được sử dụng rộng rãi trong xử lý ảnh và video, mạng neural tích chập là một loại mạng neural được thiết kế để nhận diện các đặc trưng cục bộ của dữ liệu đầu vào, như các đường biên và các đối tượng trên hình ảnh.
* **Mạng neural tái tạo (Autoencoder Neural Network)**: Được sử dụng để giảm chiều dữ liệu, mạng neural tái tạo là một loại mạng neural có khả năng học cách biểu diễn dữ liệu đầu vào dưới dạng một số lượng nhỏ các đặc trưng, giảm thiểu thông tin không cần thiết và tăng tốc xử lý.
* **Mạng neural phản hồi (Recurrent Neural Network - RNN)**: Loại mạng neural này có khả năng lưu giữ thông tin liên quan đến quá khứ và sử dụng nó để đưa ra dự đoán về tương lai. Mạng neural phản hồi được sử dụng trong các bài toán liên quan đến chuỗi thời gian, như dịch máy hoặc phân tích tín hiệu.
* **Mạng neural tự học (Self-Organizing Maps - SOM)**: Được sử dụng để tạo ra các bản đồ đa chiều của dữ liệu, mạng neural tự học có khả năng phân loại dữ liệu theo các nhóm tương tự nhau và thể hiện chúng trên một bản đồ hai chiều, giúp người dùng dễ dàng quan sát và hiểu dữ liệu.

Ngoài ra còn nhiều loại mạng neural khác như Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Deep Belief Network (DBN), Generative Adversarial Network (GAN), và nhiều loại mạng neural khác được thiết kế để giải quyết các bài toán khác nhau.

#### Đào tạo mạng nơ-ron

Để đào tạo một mạng neural, chúng ta cần thực hiện các bước sau đây:

* Chuẩn bị dữ liệu: Dữ liệu đầu vào là yếu tố quan trọng trong quá trình đào tạo mạng neural. Cần xác định và chuẩn bị dữ liệu đầu vào, cũng như chuẩn bị dữ liệu đầu ra tương ứng cho mỗi dữ liệu đầu vào.
* Chọn kiến trúc mạng neural: Chọn loại mạng neural phù hợp với bài toán cần giải quyết, xác định số lượng các lớp, số lượng neuron trong mỗi lớp, cũng như hàm kích hoạt và thuật toán tối ưu phù hợp.
* Khởi tạo mạng neural: Khởi tạo các trọng số ngẫu nhiên và bias để bắt đầu quá trình đào tạo mạng neural.
* Huấn luyện mạng neural: Sử dụng dữ liệu huấn luyện để điều chỉnh trọng số và bias của các lớp trong mạng neural. Quá trình huấn luyện này bao gồm các bước tiền xử lý dữ liệu, chia dữ liệu thành tập huấn luyện và tập kiểm tra, và sử dụng các thuật toán tối ưu như Stochastic Gradient Descent để tối thiểu hóa hàm chi phí.
* Kiểm tra mạng neural: Kiểm tra độ chính xác của mạng neural trên tập dữ liệu kiểm tra, đánh giá hiệu quả của mạng neural.
* Tinh chỉnh và đánh giá: Nếu kết quả chưa đạt yêu cầu, cần tinh chỉnh các siêu tham số của mạng neural hoặc thay đổi kiến trúc mạng để đạt được kết quả tốt hơn. Sau khi hoàn tất quá trình đào tạo và kiểm tra, cần đánh giá hiệu suất của mạng neural trên dữ liệu mới.

Quá trình đào tạo mạng neural là một quá trình lặp đi lặp lại, đòi hỏi sự tập trung và thử và sai để tìm ra các siêu tham số tối ưu nhất cho bài toán cần giải quyết.

## Natural Language Processing (NLP)

### Giới thiệu về NLP

Natural Language Processing (NLP) là một lĩnh vực của trí tuệ nhân tạo (AI) và khoa học máy tính (Computer Science) tập trung vào việc giúp máy tính hiểu và tạo ra ngôn ngữ tự nhiên (natural language) như con người. Nó liên quan đến khả năng xử lý và hiểu ngôn ngữ phổ biến của con người, bao gồm cả tiếng nói và văn bản.

### Ứng dụng của NLP

NLP được sử dụng rộng rãi trong nhiều ứng dụng, bao gồm:

* Xử lý ngôn ngữ tự nhiên: NLP giúp máy tính hiểu và tạo ra các câu và văn bản tự nhiên, cho phép máy tính tương tác với con người bằng ngôn ngữ tự nhiên.
* Phân tích tâm trạng: NLP có thể được sử dụng để phân tích cảm xúc trong văn bản, giúp các nhà tiếp thị hiểu được phản hồi của khách hàng.
* Dịch máy: NLP có thể được sử dụng để dịch văn bản từ một ngôn ngữ sang một ngôn ngữ khác.
* Tóm tắt văn bản: NLP có thể được sử dụng để tóm tắt các văn bản dài thành các phần tóm tắt ngắn gọn.
* Phân loại văn bản: NLP có thể được sử dụng để phân loại văn bản theo nội dung, chủ đề, hoặc mục đích sử dụng.
* NLP sử dụng các phương pháp xử lý ngôn ngữ tự nhiên và máy học để giúp máy tính hiểu và tạo ra ngôn ngữ tự nhiên như con người. Các phương pháp này bao gồm xử lý ngôn ngữ tự nhiên (NLP) và học sâu (deep learning), và thường sử dụng các thư viện và công cụ như Python Natural Language Toolkit (NLTK) và spaCy.

# THIẾT KẾ HỆ THỐNG

**Tóm tắt chương**

Trong chương này, nhóm tác giả sẽ mô tả về mô hình nhóm tác giả đề xuất. Bao gồm tổng quan mô hình, các luồng hoạt động của mô hình.

## Tổng quan về hệ thống

Nhóm tác giả chúng em đề xuất xây dựng một hệ thống có tích hợp Reverse Proxy – WAF như thông thường, tuy nhiên ngoài việc chỉ thuần sử dụng WAF truyền thống, thường là những WAF phát hiện tấn công bằng việc sử dụng chữ kí số, dựa vào một cơ sở dữ liệu có sẵn, nhóm sẽ tích hợp thêm máy học vào để có thực hiện kiểm tra những yêu cầu độc hại mà các WAF thông thường chưa thể phát hiện được, từ đó có thể tăng khả năng phát hiện và ngăn được những yêu cầu độc hại.

Diagram

Description automatically generated

Hình 7: Mô hình tổng quan về hệ thống đề xuất

Các thành phần trong mô hình sẽ có những chức năng và nhiệm vụ riêng. Cụ thể như sau:

* Attacker: Là máy có nhiệm vụ thực hiện tấn công đến máy web server thông qua reverse proxy.
* Reverse proxy đồng thời là WAF sử có nhiều thành phần với các vai trò cụ thể như sau:

1. WAF thông thường: Được cài đặt trên reverse proxy nhằm giúp chặn những yêu cầu độc hại. Những request không bị chặn sẽ được chuyển tới phần tiếp theo của mô hình để xử lý.
2. Bộ phát hiện tấn công: Đây là một mô hình được nhóm tác giả huấn luyện bên ngoài sử dụng phương pháp máy học. Sau khi mô hình được huấn luyện xong sẽ được lưu lại và tích hợp trực tiếp vào Web server. Những request chưa bị WAF trước đó sẽ được nhóm tác giả tiền xử lý cho phù hợp. Sau đó đưa dữ liệu đã được xử lý qua bộ phát hiện tấn công để tiếp tục kiểm tra dữ liệu đó có an toàn hay không.

Nếu dữ liệu được nhận định là bình thường thì sẽ được chuyển tiếp đến web server, nếu dữ liệu được nhận định là bất thường thì sẽ bị chặn.

* Web server: Nằm phía sau Reverse Proxy, sẽ nhận được lưu lượng từ bên ngoài đi đến thông qua Reverse Proxy.

## Tiền xử lí dữ liệu

Do các loại WAF cũng ngày càng hoàn thiện bộ blacklist rule của mình, đồng thời. Tuy nhiên, các kẻ tấn công luôn xây dựng những URL với các kiểu thay đổi khác nhau, cấu trúc, hoặc tham số của URL để khôn khéo hoá các cuộc tấn công. Nhằm phát hiện các cuộc tấn công web (đặc biệt là SQL Injection, XSS, Command Injection), từ các URL request này với nhiều định dạng, nhóm tác giả quyết định sử dụng phương pháp chuyển đổi các URL này thành định dạng nhất định. Đầu tiên, nhóm tác giả xác định một bộ các từ khoá đặc biệt, bao gồm các dấu hiệu nhận biết của các cuộc tấn công SQL Injection, XSS, … Đặc biệt, các từ khoá này được các chuyên gia chắt lọc nhiều nhất và thường được sử dụng trong các cuộc tấn công hàng ngày. Một số từ khoá được trình bày ở bảng I. Sau đó, nhóm tác giả đã xác định một bảng chuyển đổi được ánh xạ theo từng cột ở bảng II để chuyển đổi các từ khoá này thành các biểu thức giống nhau. Cuối cùng chuyển đổi các URL thành một định dạng bằng cách sử dụng các từ khoá và bảng chuyển đổi. Để trực quan hơn, nhóm tác giả đưa ra ví dụ như sau:

***/tienda1/publico/registro.jsp?modo=registro&login=bob%40%3CSCRipt%3Ealert%28Paros%29%3C%2FscrIPT%3E.parosproxy.org&password=crIn%2BAda&nombre=Aliberta&apellidos=Gochez&email=kubler5%40porqueagel.ps&dni=49372725M&direccion=*Calle*+Barrio+De+La+Torre+163%2C+&ciudad=Alar+del+Rey&cp=22770&provincia=Toledo&ntc=5546440906293369&B1=Registrar***

URL trên sẽ được chuyển đổi thành định dạng của nhóm tác giả như sau:

***/ PathString / PathString / PathString . jsp ? PureString = PureString & PureString = MixString . PureString . PureString & PureString = MixString & PureString = PureString & PureString = PureString & PureString = MixString . PureString & PureString = MixString & PureString = PureString + PureString + PureString + PureString + PureString + MixString + & PureString = PureString + PureString + PureString & cp = Numbers & PureString = PureString & PureString = Numbers & HexString = PureString***

Các từ khoá và dấu câu được nhóm tác giả tách riêng biệt, vì mỗi chúng đều có vai trò và ý nghĩa riêng khi xử lý, trong khi các từ khác và tham số được thay thế theo bảng ánh xạ chuyển đổi trong bảng II. Ngoài ra, nhóm tác giả giữ lại các tên định dạng tệp để đa dạng hoá dataset sau khi chuẩn hoá. Đặc biệt, nhóm xem xét một trường hợp đặc biệt là các từ không xuất hiện trong dữ liệu huấn luyện nhưng lại xuất hiện ở các URL thực tế. Ở đây, nhóm tác giả thay thế bằng từ khoá “MixString” để giải quyết vấn đề này.

Tổng quan quy trình chuẩn hoá dữ liệu được giải thích như sau:

*Algorithm Data Normalization*

*Input: URLs, keywords K1, extension name E1, punctuations P1*

*Output: String of Normalized URLs N*

| for i in range(len(URL)):              e = query[i]              e = e.lower()  if e in E1:                  transformed\_query.append(e)                  continue                if e in P1:                  transformed\_query.append(e)                  continue                if e in K1:                  transformed\_query.append(e)                  continue              if check\_numbers(e):                  transformed\_query.append('Numbers')                  continue                if check\_pure\_string(e):                  transformed\_query.append('PureString')                  continue                if check\_uni\_string(e):                  transformed\_query.append('UniString')                if check\_hex\_string(e):                  transformed\_query.append('HexString')                  continue              transformed\_query.append('MixString')  N = ' '.join(transformed\_query)  return N |
| --- |

**BẢNG 1: CÁC TỪ KHOÁ ĐƯỢC ĐỊNH NGHĨA**

| **Loại** | **Từ khoá** |
| --- | --- |
| SQL | select waitfor delete insert limit replace drop order by create desc group by alter … |
| HTML | doctype base area button audio abbr … |
| XSS | alert br document … |
| … | … |
| punctuation | / + ? & ; = , ( ) < > \* ! $ # | ^ { } \ ~ @ . ` [ ] : \ ' " |

**BẢNG 2: BẢNG ÁNH XẠ CHUYỂN ĐỔI TỪ**

| **Chuyển đổi từ** | **Mô tả** |
| --- | --- |
| PathString | Thay thế phần path trong URL |
| Numbers | Thay thế toàn bộ số [0-9] trong URL |
| PureString | Thay thế toàn bộ chuỗi bao gồm a-z và dấu ‘-‘ |
| UniString | Thay thế toàn bộ chuỗi chứa toàn bộ kí tự Unicode |
| HexString | Thay thế toàn bộ chuỗi chứa kí tự hexa |
| MixString | Thay thế toàn bộ chuỗi không xuất hiện trong dữ liệu huấn luyện, nhưng xuất hiện trong URL thực tế |
| Dấu câu (punctuation) được giữ nguyên | |

## Triển khai các mô hình

Nhóm tác giả bắt đầu với những mô hình LSTM và xây dựng lần lượt các kiến trúc deep learning để giải quyết vấn đề về phân loại các URL. Bao gồm:

* Bi-LSTM
* CNN-LSTM
* …

Có rất nhiều cách tiếp cận để xây dựng các mô hình. Mô hình trình tự (Sequence model) rất giỏi trong việc phân tích ngữ cảnh trong câu trong văn bản, vì nó có thể hiểu nghĩa thay vì sử dụng các kỹ thuật đếm từ dương tính hay âm tính như trong mô hình Bag-of-Words.

Nhóm tác giả đề xuất mô hình Bidirectional-LSTM được mô tả cụ thể dưới đây:

### Kiến trúc mô hình Birectional-LSTM

**Lớp nhúng (Embedding Layer)**: Lớp này chiu trách nhiệm chuyển đổi các kí tự thành biểu diễn vector của chúng được tạo bởi mô hình Word2Vec.

**Các tham số**

* **input\_dim**: Kích thước của bộ từ vựng
* **output\_dim**: Kích thước của ma trận nhúng từ
* **weights**: khởi tạo ma trận nhúng
* **trainable**: có giá trị là True hoặc False, chỉ định lớp này có thể huấn luyện hay không

**Bidirectional**: Bidirectional bao bọc RNNs. Nghĩa là các câu từ được thực hiện cả từ trái sang phải và phải sang trái trong lớp RNN được bao bọc.

**LSTM**: Long Short Term Memory, đây là một biến thể của RNN có ô trạng thái bộ nhớ, mục đích để tìm hiểu ngữ cảnh của các từ nằm xa hơn trong văn bản nhất định và mang nghĩa theo ngữ cảnh, thay vì chỉ các từ lân cận như trong trường hợp của RNN.

**Các tham số**

* **units**: Số chiều của không gian đầu ra
* **dropout**: Tỷ lệ giảm xuống để chuyển đổi tuyến tính của đầu vào
* **return\_sequence**: Trả về đầu ra trong chuỗi đầu ra, hoặc toàn bộ chuỗi

**Conv1D**: Lớp này tạo ra một lõi tích hợp với lớp đầu vào trên một chiều duy nhất để tạo ra thanh đo cho đầu ra.

**Các tham số**

* **filters**: Số chiều của không gian đầu ra (số lượng bộ lọc đầu ra trong tích hợp)
* **dropout**: Tỷ lệ giảm xuống để chuyển đổi tuyến tính của đầu vào
* **return\_sequence**: Trả về đầu ra trong chuỗi đầu ra, hoặc toàn bộ chuỗi

**GlobalAveragePooling1D:** Giảm mẫu biểu diễn đầu vào bằng cách lấy giá trị trung bình trên các chiều khác nhau.

**GlobalMaxPool1D**: Giảm mẫu biểu diễn đầu vào bằng cách lấy giá trị lớn nhất trên các chiều khác nhau.

**Dense**: Lớp dày đặc thêm một lớp kết nối đầy đủ trong mô hình. Đối số được truyền vào chỉ định số lượng nút trong lớp đó.

Lớp cuối (dense) có kích hoạt “Softmax”, được sử dụng để chuyển đổi đầu vào thành một số trong khoảng từ 0 đến 1. Softmax thường được dùng khi chúng ta có 2 phân loại đầu ra.

### Tổng quan mô hình Bidirectional-LSTM:

Diagram

Description automatically generated

Hình 8: Mô hình Bidirectional LSTM

## Word Embedding

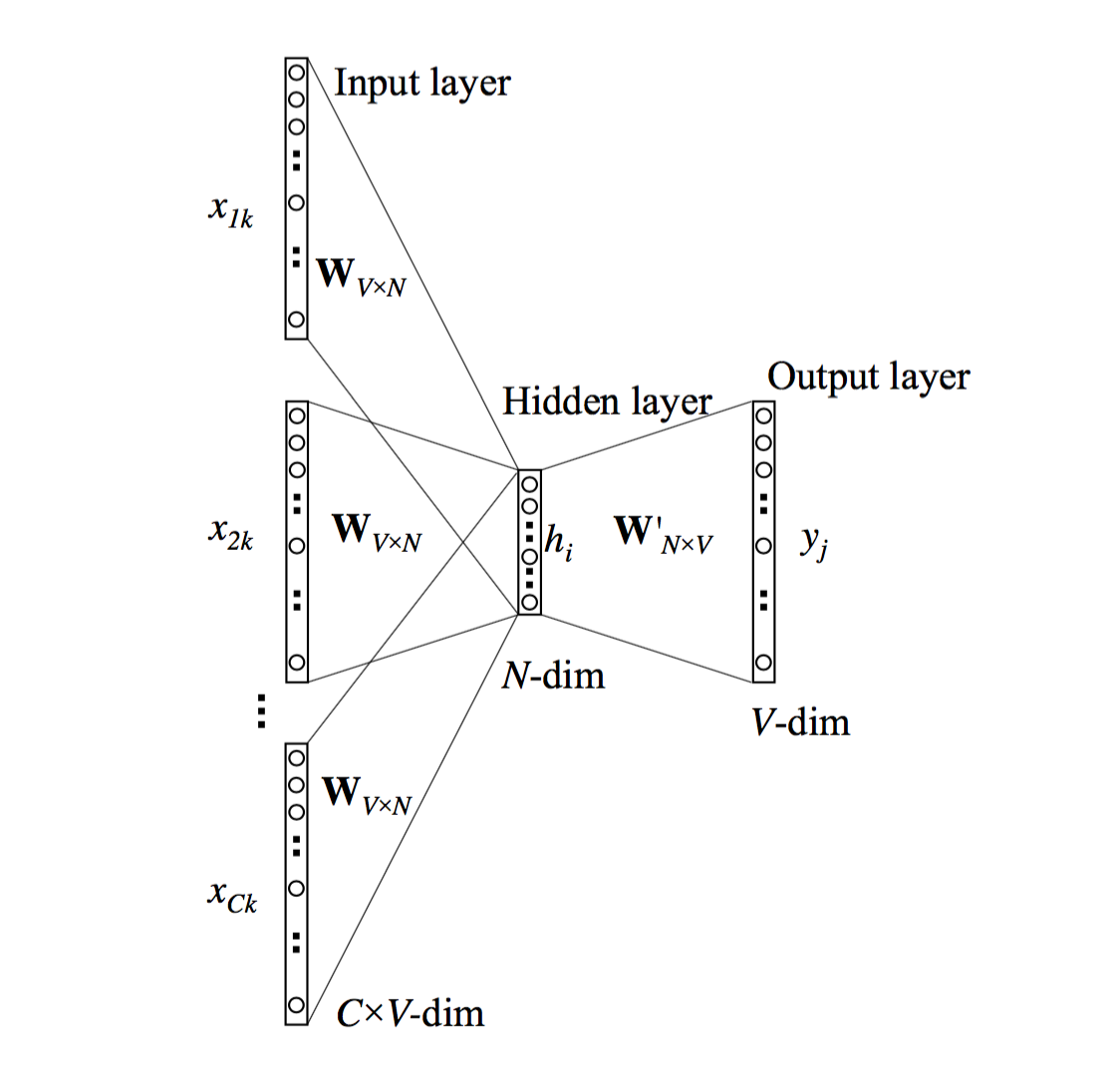
Trong khi triển khai các mô hình, nhóm tác giả có liên hệ đến kỹ thuật nhúng từ (word embedding). Về cơ bản, nhúng từ là một kiểu biểu diễn từ sao cho các từ có nghĩa tương đương nhau thì sẽ được biểu diễn tương đương. Đây là một kiểu biểu diễn phân tán trong các văn bản, và cũng là một trong những bước đặc biệt quan trọng, đưa ra hiệu suất vượt trội của các phương pháp deep learning đối với các vấn đề xử lý ngôn ngữ tự nhiên hiện nay.

Về thực tế, nhúng từ là một loại kỹ thuật mà trong đó, các từ riêng lẻ được biểu diễn dưới dạng các vector chứa giá trị trong một không gian vector được định nghĩa trước. Mỗi từ riêng lẻ đó được ánh xạ đến một vector và các giá trị vector đó được “học” theo cách tương tự như mạng nơ-ron. Do đó, kỹ thuật này thường được gộp vào lĩnh vực học sâu.

Mỗi từ được biểu diễn bằng một vector có giá trị, thường là vài chục hoặc vài trăm chiều. Điều này khác hẳn với vài nghìn hoặc vài triệu chiều cho các loại biểu diễn từ khác, ví dụ như mã hoá one-hot.

Biểu diễn phân tán được học dựa trên việc dùng các từ. Từ đó cho phép các từ được dùng theo những cách tương tự nhau, dẫn đến việc các từ đấy có thể biểu diễn cho nhau, dựa theo nghĩa của chúng một cách tự nhiên nhất. Điều này trái ngược với loại biểu diễn ‘crisp but fragile’ trong mô hình ‘bag of words’, trừ khi được quản lý rõ ràng, các từ khác nhau có cách biểu diễn khác nhau, bất kể các từ đấy được sử dụng ra sao.

Có 3 loại thuật toán nhúng từ: Embedding Layer, Word2Vec, GloVe. Tuy nhiên ở đây, nhóm tác giả sử dụng Word2Vec cho mô hình của mình. Ngoài ra, trong thuật toán Word2Vec, chúng lại có 2 mô hình khác nhau: Continous Bag-of-Words (CBOW) và Continous Skip-Gram Model. Và lựa chọn cuối cùng là CBOW.



Hình 9: Mô hình CBOW

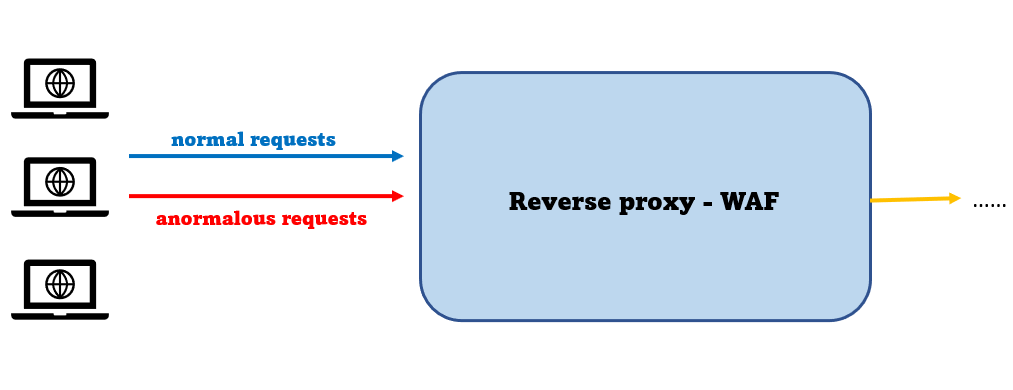
Như được mô tả ở hình 9, mô hình lấy vector 1 chiều cho mọi từ trong một URL làm đầu vào, một vector ẩn và vector một chiều làm đầu ra. Đối với k vector của từ có nghĩa tương đương nhất với từ thứ i sẽ là đầu vào, từ thứ i sẽ là nhãn. Phần đầu tiên là ánh xạ k vectors đầu vào vào một đại diện ẩn thông qua ánh xạ . là ma trận trọng số giữa layer đầu vào và layer ẩn. Ngoài ra, có một ánh xạ khác giữa layer ẩn và layer đầu ra, và .

## Chi tiết quy trình hoạt động của hệ thống

### Yêu cầu được gửi từ máy tấn công đến Reverse proxy -WAF

Máy tấn công có thể là một hoặc nhiều máy. Kịch bản được đưa ra máy tấn công là một máy có nhu cầu truy cập đến máy chủ web để sử dụng dịch vụ web đang được chạy trên máy chủ web, hoặc với mục đích khai thác, phá hoại hệ thống, …

Tuy nhiên trong trường hợp này, máy chủ web của chúng ta lại đang được đặt trong một hệ thống mạng riêng biệt và không có cách nào có thể truy cập trực tiếp từ Internet, một cách duy nhất có thể truy cập thông qua Reverse proxy đứng trước.



Hình 10: Request được gửi từ Attacker đến Reverse proxy

### Yêu cầu được xử lý tại máy Reverse proxy – WAF

Với **WAF A** là một web application thông thường, **Model B** là bộ phát hiện tấn công do nhóm tác giả phát triển. Yêu cầu được xử lý tại máy Reverse proxy – WAF sẽ được thực hiện như mô tả bên dưới.

Tại Reverse Proxy – WAF sẽ thực hiện kiểm tra từng khối. Yêu cầu sẽ được thực hiện kiểm tra ở WAF A trước. Ở WAF A sẽ có hai trường hợp có thể xảy ra:

* Trường hợp 1: WAF A nhận định đây là một yêu cầu độc hại, nó sẽ thực hiện chặn yêu cầu tại đây, không thực hiện chuyển tiếp yêu cầu đến model B để thực hiện kiểm tra.
* Trường hợp 2: WAF A không nhận định đây là một yêu cầu độc hại, nghĩa là yêu cầu đã vượt qua được sự kiểm tra của WAF A, yêu cầu sẽ được tiếp tục chuyển đến Model B để kiểm tra. Tại đây lại tiếp tục có hai trường hợp có thể xảy ra:
* Model B cho ra kết quả đây là một yêu cầu độc hại (anomalous): Yêu cầu sẽ bị chặn ngay tại khối hiện tại và không thực hiện chuyển qua khối tiếp theo
* Model B cho ra kết quả đây là một yêu cầu bình thường (Không phải anomalous), thực hiện chuyển sang khối tiếp theo.

Diagram

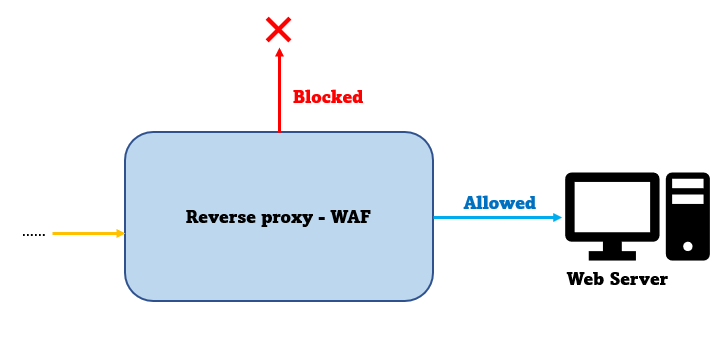
Description automatically generated

Hình 11: Quá trình xử lý yêu cầu trong Reverse proxy

### Yêu cầu được gửi từ máy Reverse proxy – WAF đến máy chủ web

Như nhóm tác giả đã trình bày ở phần trên, sau khi trải qua sự xử lý ở Reverse proxy – WAF sẽ có 2 khả năng có thể xảy ra với một yêu cầu: bị chặn tại Reverse proxy – WAF hoặc là có thể vượt qua được mọi quá trình kiểm tra ở Reverse proxy – WAF.

Với những yêu cầu có thể vượt qua được sự kiểm tra ở Reverse proxy – WAF, được nhận định là một request bình thường sẽ được Reverse proxy – WAF chuyển tiếp nó đến máy chủ web.



Hình 12: Yêu cầu được chuyển từ Reverse proxy đến máy chủ

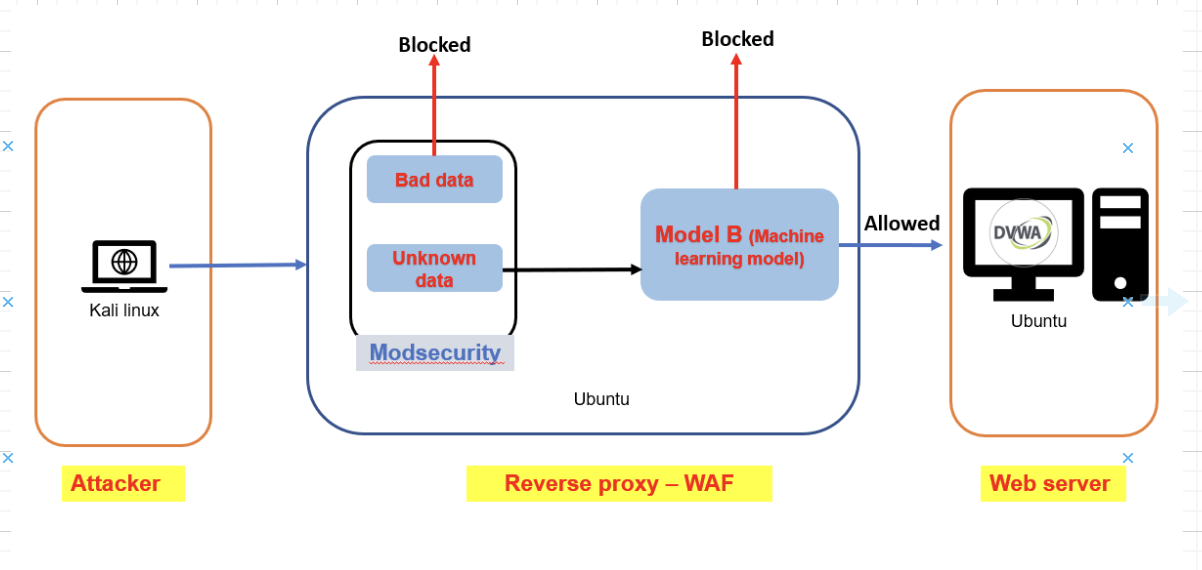
# HIỆN THỰC HỆ THỐNG VÀ THỰC NGHIỆM

**Tóm tắt chương**

Trong chương này, nhóm sẽ trình bày chi tiết các bước để hiện thực hệ thống nhóm đã đề xuất. Tiếp theo sẽ là các kịch bản thực nghiệm với mô hình trên Python và sau khi đã được tích hợp vào NGINX. Cuối cùng là đưa ra kết quả và nhận xét chung cho các thực nghiệm đã đề ra.

## Hiện thực hệ thống

### Mô hình triển khai



Hình 13: Mô hình triển khai

Mô hình mạng thực tế được xây dựng trong mạng LAN, gồm có 3 máy chính được sử dụng:

* Ubuntu : Làm reverse proxy – WAF + Làm máy chủ web (Web server)
* Kali linux : Làm máy Attacker

### Cấu hình các máy thực nghiệm

#### Máy được sử dụng để huấn luyện mô hình

* + Ubuntu 20.04 (32 GB RAM, Intergrated GPU, Intel Xeon CPU E5-2660).
  + Windows 11 (16 GB RAM, RTX 3050, Intel Core i7 11800H).
* Công nghệ, phần mềm được sử dụng:
  + Phần mềm:
    - Anaconda Navigator
    - Jupyter Notebook
    - Visual Studio Code
  + Ngôn ngữ
    - Python
    - C++

#### Máy được sử dụng làm Reverse proxy – WAF

* Hệ điều hành sử dụng: Ubuntu 20.04.5 LTS
* Thông tin phần cứng: 16G RAM, 64G hard disk
* Trên máy cài đặt NGINX làm server và Modsecurity làm Web Application Firewall

#### Máy được sử dụng làm máy chủ web

* Hệ điều hành được sử dụng: Ubuntu 20.04.5 LTS
* Thông tin phần cứng: 4G RAM, 64G hard disk
* Trên máy thực hiện cài đặt DVWA với web server là Apache

#### Máy được sử dụng làm máy tấn công (Attacker)

* Hệ điều hành sử dụng: Kali linux 2024.3
* Thông số phần cứng: 6G RAM, 64G hard disk

Trên máy chủ yếu sử dụng Python để chạy script gửi yêu cầu (request) thực nghiệm và Burpsuite để thực hiện làm proxy bắt lại lưu lượng được gửi đi.

### Dữ liệu và tiền xử lí

Thử nghiệm của nhóm tác giả dựa trên bộ dataset: HTTP Dataset CSIC 2010

Bộ dữ liệu HTTP CSIC 2010 là một tập dữ liệu gồm các gói tin HTTP được sử dụng để phát hiện tấn công web trên các ứng dụng web. Bộ dữ liệu này được tạo ra bởi nhóm nghiên cứu CSIC (Computer Science and Engineering Department) thuộc Đại học Carlos III ở Madrid, Tây Ban Nha. Bộ dữ liệu gồm hơn 36.000 gói tin HTTP được thu thập từ hơn 150 trang web khác nhau, bao gồm cả các trang web thương mại điện tử và các trang web tin tức. Các gói tin này được ghi lại trong khoảng thời gian từ tháng 8 năm 2009 đến tháng 3 năm 2010.

Sau khi xoá dữ liệu trùng lặp và các dữ liệu được gắn nhãn không chính xác ra khỏi bộ dữ liệu, nhóm tác giả còn lại khối lượng dữ liệu được mô tả ở bảng sau:

**BẢNG 3: SỐ LƯỢNG BỘ DỮ LIỆU SAU KHI LỌC**

| Dataset | Normal Requests | Anomalous Requests |
| --- | --- | --- |
| CSIC2010 | 2576 | 4922 |

### Cài đặt

Nhóm tác giả đưa ra những vấn đề cần giải quyết như sau:

1. Chuyển đổi các URL thành vector được chuyển đổi theo 1 định dạng nhất định, tuy nhiên vẫn giữ lại hầu hết thông tin của nó.
2. Chuẩn bị các mô hình học sâu, nhằm tìm ra mô hình khả quan để phát hiện các yêu cầu bất thường.
3. Tối ưu hoá mô hình để đặt độ chính xác cao và phát hiện sai thấp nhất.

Đối với vấn đề đầu tiên, trước khi đưa URL vào để kiểm tra tính độc hại, nhóm tác giả thử với một số phương pháp để chuyển đổi URL thành vector, điển hình như mã hoá bằng mã cp437, mã hoá các ký tự bằng vector của chúng từ mô hình word2vec, mã hoá các từ trong URL như phương pháp được trình bày ở 3.4.1. Sau đó về vấn đề thứ 2, nhóm tác giả tiến hành một số thử nghiệm trên các thuật toán học máy và học sâu, bao gồm Naïve Bayes, Linear Regression, LSTM, …

### Tiền xử lý dữ liệu trong NGINX

Để mô hình machine learning có thể hoạt động được trong NGINX, request được đưa vào cần phải trải qua quá trình tiền xử lý trước khi đưa vào mô hình.

Cụ thể, nhóm sẽ chia thành hai nhóm chính để xử lý và request có bao gồm parameter và request có bao gồm body. Các bước xử lý dữ liệu trong NGINX trước khi đưa vào mô hình machine learning sẽ bao gồm:

* Tách lấy phần path trong request.
* Tách lấy phần parameter và request body trong request.
* Thực hiện phân tích xử lý cho phần path, parameter và request body cho phù hợp với xử lý trong Python.
* Đưa dữ liệu đã phân tích ở trên về dạng mảng cho phù hợp với tập tin token.txt (tập tin này được sinh ra trong quá trình huấn luyện mô hình trên python).

Chi tiết cụ thể các bước thực hiện sẽ được trình bày bên dưới:

#### Tách lấy phần đường dẫn (path) trong yêu cầu

Trong NGINX, ngx\_http\_request\_t là một cấu trúc chứa các chi tiết của request và response.

Uri chứa nội dung đường dẫn của yêu cầu. Cụ thể trong trường hợp này với ngx\_http\_request\_t \*r, thì r->uri.data sẽ chứa nội dung đường dẫn và r->uri.len sẽ chứa độ dài của đường dẫn.

| int len\_path = r->uri.len;      u\_char \*raw\_data\_uri = ngx\_pcalloc(r->pool,len\_path);      ngx\_memcpy(raw\_data\_uri,r->uri.data,len\_path);      u\_char \*data\_uri = malloc(len\_path);      strncpy((char\*)data\_uri,(char\*)raw\_data\_uri,len\_path); |
| --- |

#### Tách lấy phần nội dung tham số (parameter) và body trong yêu cầu:

* Tách lấy phần nội dung tham số:

Chúng ta có thể sử dụng ngx\_http\_request\_t để tách lấy phần tham số trong yêu cầu.

args chứa nội dung tham số của yêu cầu. Cụ thể trong trường hợp này với ngx\_http\_request\_t \*r, thì r->args.data sẽ chứa nội dung tham số và r->args.len sẽ chứa độ dài của phần nội dung tham số.

| int len\_param = r->args.len;      // fprintf(stderr,"\nLen param: %li\n", r->args.len);      u\_char \*raw\_data\_param = ngx\_pcalloc(r->pool,len\_param);      ngx\_memcpy(raw\_data\_param,r->args.data,len\_param);      u\_char \*data\_param = malloc(len\_param+1);      strncpy((char\*)data\_param,(char\*)raw\_data\_param,len\_param);      data\_param[len\_param] = '\0'; |
| --- |

* Tách lấy nội dung của body:

Đối với hoạt động đầu vào, đầu ra, bộ đệm được liên kết theo chuỗi. Chuỗi là một chuỗi các liên kết chuỗi kiểu ngx\_chain\_t:

| typedef struct ngx\_chain\_s  ngx\_chain\_t;  struct ngx\_chain\_s {  ngx\_buf\_t    \*buf;  ngx\_chain\_t  \*next;  }; |
| --- |

Để có thể lấy được nội dung của request body, nhóm tác giả sẽ sử dụng ngx\_chain\_t, cụ thể là ngx\_buf\_t \*buf trong cấu trúc ngx\_chain\_t.

Đối với hoạt động đầu vào/đầu ra, NGINX cung cấp loại bộ đệm (buffer) ngx\_buf\_t. Thông thường, ở đây được sử dụng để giữ dữ liệu được ghi vào đích hoặc đọc từ nguồn. Bộ đệm có thể tham chiếu dữ liệu trong bộ nhớ hoặc trong một tệp và về mặt kỹ thuật, bộ đệm có thể tham chiếu cả hai cùng một lúc. Bộ nhớ cho bộ đệm được phân bổ riêng và không liên quan đến cấu trúc bộ đệm ngx\_buf\_t.

Cấu trúc ngx\_buf\_t có nhiều trường: start, end, pos, last, file, tag, flush, …

Trong phần này, trường được sử dụng sẽ là trường pos.

| ngx\_chain\_t \*cl;  ngx\_buf\_t    \*buf;  cl = r->request\_body.bufs;  for( ;NULL != cl; cl = cl->next )  {      buf = cl->buf;      u\_char \*data = buf->pos;  } |
| --- |

#### Xây dựng hàm urlDecode để thực hiện giải mã nội dung của yêu cầu:

Ý tưởng để xây dựng hàm urlDecode như sau:

* Tạo ra 1 mảng char gồm 2 kí tự (eStr[]) nơi đây sẽ đảm nhiệm lưu trữ các kí tự sau dấu ‘%’ để thực hiện chuyển đổi.
* Vòng while đầu để đảm bảo việc nếu chuỗi không còn kí tự nào để decode thì kết thúc.
* Dùng biến đếm kiểm tra lần lượt 2 kí tự liên tiếp, nếu đó là 2 kí tự hex, tiến hành đưa nó vào eStr để thực hiện chuyển đổi sang unicode.
* Xoá 2 kí tự hex đó đi, đưa kí tự vừa được chuyển đổi vào chuỗi.

| //URL Decode function  u\_char \*urlDecode(u\_char \*str)  {    int d = 0; /\* whether or not the string is decoded \*/    u\_char \*dStr = malloc(strlen((const char\*)str) + 1);    u\_char eStr[] = "00"; /\* for a hex code \*/    strcpy((char\*)dStr, (char\*)str);    while(!d) {      d = 1;      int i; /\* the counter for the string \*/      int len\_dStr = strlen((const char\*)dStr);      for(i=0;i<len\_dStr;++i) {        if(dStr[i] == '%') {          if(dStr[i+1] == 0)            return dStr;          if(isxdigit(dStr[i+1]) && isxdigit(dStr[i+2])) {            d = 0;            /\* combine the next to numbers into one \*/            eStr[0] = dStr[i+1];            eStr[1] = dStr[i+2];            /\* convert it to decimal \*/            long int x = strtol((char\*)eStr, NULL, 16);            /\* remove the hex \*/            memmove(&dStr[i+1], &dStr[i+3], strlen((char\*)&dStr[i+3])+1);            dStr[i] = x;          }        }      }    }    return dStr;  } |
| --- |

#### Thực hiện phân tích nội dung đường dẫn:

Việc thực hiện phân tích đường dẫn sẽ được xử lý bằng hàm handle\_path:

| void handle\_path(char \*parsed\_path, char\* path, int len, char\* punctuations)  {      parsed\_path[0] = '/';      parsed\_path[1] = ' ';      int k = 2;      for(int i=1; i< len; i++)      {          if(isInString(punctuations, 29, path[i]))          {              parsed\_path[k] = ' ';              parsed\_path[k+1] = path[i];              parsed\_path[k+2] = ' ';              k = k+3;          }          else{              parsed\_path[k] = path[i];              k++;          }      }      parsed\_path[k] = ' ';      parsed\_path[k+1] = '?';      parsed\_path[k+2] = '\0';  } |
| --- |

Punctuations sẽ chứa các kí tự đặc biệt: /+?&;=,()<>\*!$#|^{}\~@.`[]:\'", nếu như trong chuỗi có những kí tự này thì chúng sẽ được phân cách bởi khoảng trắng (“ “).

Table

Description automatically generated

Hình 9: Phân tách đường dẫn dựa vào những kí tự đặc biệt

#### Thực hiện phân tích tham số (parameter) và body

Việc thực hiện phân tích parameter và body sẽ được thực hiện bằng hàm handle\_data:

| void handle\_data(char\* parsed\_data, char\* data, int len, char\* punctuations)  {      //char \*parsed\_data = malloc(1000);      //parsed\_data[0] = '?';      parsed\_data[0] = ' ';      int k = 1;      for(int i = 0; i< len; i++)      {          if(isInString(punctuations, 29, data[i]))          {              parsed\_data[k] = ' ';              parsed\_data[k+1] = data[i];              parsed\_data[k+2] = ' ';              k = k+3;          }          else{              parsed\_data[k] = data[i];              k++;          }      }      parsed\_data[k] = '\0';  } |
| --- |

Punctuations sẽ chứa các kí tự đặc biệt: /+?&;=,()<>\*!$#|^{}\~@.`[]:\'", nếu như trong chuỗi có những kí tự này thì chúng sẽ được phân cách bởi khoảng trắng (“ “).

Table

Description automatically generated

Hình 10: Phân tách dữ liệu dựa vào những kí tự đặc biệt

#### Đưa dữ liệu đường dẫn đã xử lý ở trên về dạng mảng số

Việc đưa dữ liệu đường dẫn về dạng mảng các số sẽ được xử lý bằng hàm tokenizer\_path:

| void tokenizer\_path(int \*token\_arr, char\* parsed\_path, char pattern[320][100], int len, char \*punctuations, char ext[2445][100], int len\_ext, int \*len\_path\_arr)  {      //int \*token\_arr = malloc(100);      int i = 0;      int position\_pathstring = find\_position\_of\_str(pattern, "pathstring", len);      //int position\_slash = find\_position\_of\_str(pattern, "/", len);      int position\_purestring = find\_position\_of\_str(pattern, "purestring", len);      char\* token = strtok(parsed\_path," ");      char\* pre\_string = NULL;      while( token != NULL ) {       if((int)strlen(token)==1 && isInString(punctuations, 29, token[0]))          {              token\_arr[i] = find\_position\_of\_str(pattern, token , len);          }       else if(isInArr(token,ext,len\_ext) && pre\_string != NULL && (pre\_string[0]=='.'))       {           int position\_ext = find\_position\_of\_str(pattern, token, len);           if(position\_ext==-1)              position\_ext = find\_position\_of\_str(pattern, token, len);           if(position\_ext==-1)              token\_arr[i] = position\_purestring;           else              token\_arr[i] = position\_ext;       }       else       {          token\_arr[i] = position\_pathstring;       }       i++;       if(pre\_string != NULL)          free(pre\_string);       pre\_string = malloc((int)strlen(token)+1);       pre\_string = memcpy(pre\_string,token,(int)strlen(token));       token = strtok(NULL, " ");      \*len\_path\_arr = i;      }  } |
| --- |

Dữ liệu trong đường dẫn sẽ được chia thành các loại chính: các kí tự đặc biệt nằm trong punctuations, Pathstring, extension. Cụ thể như sau:

* Punctuations: Chứa các kí tự đặc biệt: /+?&;=,()<>\*!$#|^{}\~@.`[]:\'"
* Extension: Phần mở rộng của file: php, exe, jsp, gif, bak, …
* Không nằm trong các dạng trên thì sẽ là Pathstring.

Sau khi trải qua bước ở trên, một chuỗi đường dẫn sẽ có dạng tương tự như bên dưới:

Table

Description automatically generated

Hình 11: Đường dẫn được đưa về dạng thống nhất

Đến bước này, sử dụng hàm strtok() trong C để tách chuỗi trên thành nhiều chuỗi nhỏ dựa trên kí tự khoảng trắng. Với mỗi chuỗi nhỏ sẽ thực hiện tìm vị trí của nó trong tập tin token.txt (vị trí của kí tự đặc biệt, vị trí của Pathstring, vị trí của extension,…), vị trí này sẽ là giá trị của nó khi đưa vào mảng số để đưa vào mô hình học máy.

Chart, diagram

Description automatically generated

Hình 12: Chuyển đường dẫn về dạng mảng số

#### Đưa parameter và body đã được xử lý ở trên về dạng mảng

Việc đưa parameter và body về dạng mảng các số sẽ được thực hiện bằng hàm tokenizer\_data:

| void tokenizer\_data(int \*token\_arr, char\* parsed\_data, char pattern[320][100], int len, int \*len\_path\_arr, char \*punctuations, char KEYWORDS\_IN\_TRAINING[100][20])  {      int i = 0;      int position\_purestring = find\_position\_of\_str(pattern, "purestring", len);      int position\_unistring = find\_position\_of\_str(pattern, "unistring", len);      int position\_hexstring = find\_position\_of\_str(pattern, "hexstring", len);      int position\_number = find\_position\_of\_str(pattern, "numbers", len);      int position\_mixstring = find\_position\_of\_str(pattern, "mixstring", len);      char\* token = strtok(parsed\_data," ");      while(token != NULL ) {          if((int)strlen(token)==1 && isInString(punctuations,29, token[0]))          {              token\_arr[i] =              find\_position\_of\_str(pattern, token, len);              i++;              token = strtok(NULL, " ");              \*len\_path\_arr = i;              continue;          }          else if(isSenString(token,KEYWORDS\_IN\_TRAINING))          {              token\_arr[i] = find\_position\_of\_str(pattern, token, len);              i++;              token = strtok(NULL, " ");              \*len\_path\_arr = i;              continue;          }          else if(isNumber(token,(int)strlen(token)))          {              token\_arr[i] = position\_number;              i++;              token = strtok(NULL, " ");              \*len\_path\_arr = i;              continue;          }          else if(isHexString(token,(int)strlen(token)))          {              token\_arr[i] = position\_hexstring;              i++;              token = strtok(NULL, " ");              \*len\_path\_arr = i;              continue;          }          else if(isPureString(token,(int)strlen(token)))          {              token\_arr[i] = position\_purestring;              i++;              token = strtok(NULL, " ");              \*len\_path\_arr = i;              continue;          }          else if(isUniString(token,(int)strlen(token)))          {              token\_arr[i] = position\_unistring;              i++;              token = strtok(NULL, " ");              \*len\_path\_arr = i;              continue;          }          else          {              token\_arr[i] = position\_mixstring;              i++;              token = strtok(NULL, " ");              \*len\_path\_arr = i;              continue;          }        }  } |
| --- |

Dữ liệu của data và parameter và body là rất phức tạp và đa dạng, nên tất cả các dạng dữ liệu sẽ được đưa về các dạng chính: các kí tự đặc biệt nằm trong punctuations, các từ đặc biệt nằm trong mảng KEYWORDS\_IN\_TRAINING, Purestring, Unistring, Hexstring, Number, Mixstring. Từng dạng dữ liệu sẽ có những đặc điểm riêng, và sẽ có những function riêng để nhận biết. Cụ thể như sau:

* Punctuations: Chứa các kí tự đặc biệt: /+?&;=,()<>\*!$#|^{}\~@.`[]:\'"
* KEYWORKS\_IN\_TRANING: mảng chứa các từ đặc biệt phục vụ cho quá trình training model: ['style', 'sub', 'textarea', 'table', 'ls', 'order', 'rollback', 'alter', 'top', 'basefont', 'tr', 'section', 'cp', 'blockquote', 'by', 'summary', 'group', 'bdi', 'delay', 'desc', 'commit', 'track', 'doctype', 'time', 'where', 'span', 'alert', 'abbr', 'drop', 'bdo', 'delete', 'strong', 'base', 'tbody', 'like', 'thead', 'from', 'small', 'tfoot', 'br', 'xp\_cmdshell', 'dir', 'strike', 'area', 'limit', 'acronym', 'title', 'sup', 'all', 'body', 'pwd', 'savepoint', 'null', 'set', 'transaction', 'article', 'cat', 'create', 'mv', 'tail', 'aside', 'select', 'address', 'truncate', 'into', 'update', 'head', 'button', 'deletescript', 'replace', 'waitfor', 'audio', 'td', 'source', 'applet', 'big', 'insert', 'distinct', 'more' ]
* Purestring: Chỉ gồm các kí tự từ a-z, A-Z và –

| int isPureString(char\* input, int len)  {      char \*pattern = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ-";      for(int i=0; i < len; i++)      {          if(!isInString(pattern, strlen(pattern), input[i]))             return 0;      }      return 1;  } |
| --- |

* Unistring: Các kí tự có giá trị Decimal không nằm trong khoảng từ 32 đến 127:

| int isUniString(char\* input, int len)  {      for(int i=0; i < len; i++)      {          if((int)input[i] < 32)              return 1;      }      return 0;  } |
| --- |

* Hexstring: Là chuỗi có dạng hexa, chỉ bao gồm các kí tự: a-f, A-F, 0-9

| int isHexString(char\* input, int len)  {      char \*pattern = "0123456789abcdefABCDEF";      for(int i=0; i < len; i++)      {          if(!isInString(pattern, strlen(pattern), input[i]))             return 0;      }      return 1;  } |
| --- |

* Number: Là chuỗi chỉ bao gồm chữ số.

| int isNumber(char\* input, int len)  {      char \*pattern = "0123456789";      for(int i=0; i < len; i++)      {          if(!isInString(pattern,10,input[i]))              return 0;      }      return 1;  } |
| --- |

* Mixstring: Chuỗi không thuộc vào bất kì loại nào bên trên.

Sau khi trải qua bước này, chuỗi sẽ có dạng như bên dưới:

Table

Description automatically generated

Hình 13: Dữ liệu được chuyển sang dạng thống nhất

Đến bước này, sử dụng hàm strtok() trong C để tách chuỗi trên thành nhiều chuỗi nhỏ dựa trên khí tự khoảng trắng. Với mỗi chuỗi nhỏ sẽ thực hiện tìm vị trí của nó trong tập tin token.txt, vị trí này sẽ là giá trị của nó khi đưa vào mảng số để đưa vào mô hình học máy.

Chart

Description automatically generated

Hình 14: Dữ liệu được chuyển sang dạng mảng số

### Triển khai ModSecurity vào NGINX

* Cấu hình NGINX làm reverse proxy:
* Trong /etc/nginx/sites-enabled, tạo tập tin custom\_server.conf. Nó cấu hình một virtual server lắng nghe trên port 80, và “proxy” tất cả các yêu cầu tới web server:

Text

Description automatically generated

* Download NGINX source package:
* Kiểm tra phiên bản NGINX đang sử dụng:

Text

Description automatically generated

* Sử dụng câu lệnh bên dưới để download NGINX Source Package với version tương ứng:

| $ wget https://nginx.org/download/nginx-version.tar.gz |
| --- |

* Giải nén tập tin vừa download:

Text

Description automatically generated with medium confidence

* Cài đặt thư viện libmodsecurity3:
* Sử dụng câu lệnh:

git clone --depth 1 -b v3/master --single-branch [https://github.com/SpiderLabs/ModSecurity /usr/local/src/ModSecurity/](https://github.com/SpiderLabs/ModSecurity%20/usr/local/src/ModSecurity/)

* Di chuyển đến cloned directory và thực hiện cài đặt submodules:

| $ sudo git submodule init  $ sudo git submodule update |
| --- |

* Sau đó, thực hiện build environment:

| $ sudo ./build.sh  $ sudo ./configure |
| --- |

* Biên dịch mã nguồn và thực hiện cài đặt các tiện ích khác bằng lệnh make:



* Sau khi hoàn thành, thực hiện cài đặt các thư viện

| $ sudo make -j4 |
| --- |

* Tải và biên dịch ModSecurity v3 Nginx Connector:
* Bước tiếp theo là tải xuống và biên dịch ModSecurity Nginx connector, Modsecurity Nginx connector liên kết thư viện Libmodsecurity với máy chủ web NGINX.
* Sử dụng câu lệnh sau:

| $ git clone --depth 1 https://github.com/SpiderLabs/ModSecurity-nginx.git /usr/local/src/ModSecurity-nginx/ |
| --- |

* Thực hiện di chuyển đến thư mục chứa NGINX Source Package để thực hiện compile ModSecurity NGINX Connector module với –with-compat flag (tùy chọn --with-compat làm cho mô-đun ModSecurity Nginx Connector tương thích với nhị phân với thư viện NGINX hiện tại.

| $ sudo ./configure –with-compat –add-dynamic-module=/usr/local/src/ModSecurity-nginx |
| --- |

* Sau khi hoàn tất, build ModSecurity Nginx Connector bằng cách sử dụng lệnh make:

| $ sudo make module |
| --- |

* Mô-đun vừa được tạo sẽ được lưu trong objs với tên là ngx\_http\_modsecurity\_module.so. Thực hiện sao chép mô-đun này vào /etc/nginx/modules.
* Load Modsecurity Nginx Connector Module:

Trong tập tin /etc/nginx/nginx.conf, thêm dòng:

| load\_module modules/ngx\_http\_modsecurity\_module.so; |
| --- |



* Trong custom\_server.conf thêm dòng:

| modsecurity on;  modsecurity\_rules\_file /etc/nginx/modsec/main.conf; |
| --- |

Điều này enable ModSecurity cho tất cả NGINX virtual host.

Text

Description automatically generated

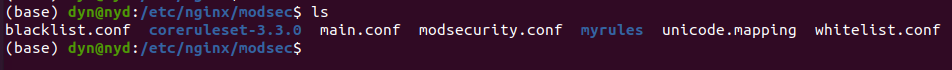
* Tiếp theo, tạo thư mục /etc/nginx/modsec để lưu trữ cấu hình ModSecurity:

Text

Description automatically generated

* Copy tập tin cấu hình ModSecurity vào thư mục trên:

| $ sudo cp /usr/local/src/ModSecurity/modsecurity.conf-recommended /etc/nginx/modsec/modsecurity.conf |
| --- |



* Mở tập tin modsecurity.conf và thực hiện sửa đổi SecRuleEngine DetectionOnly thành SecRuleEngine On

Text

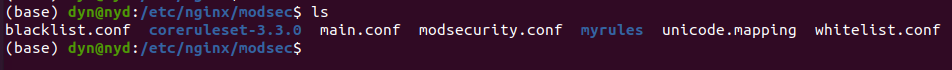
Description automatically generated

* Tiếp theo, tạo tập tin /etc/nginx/modsec/main.conf: Thêm dòng bên dưới để tham chiếu tới tập tin cấu hình /etc/nginx/modsec/modsecurity.conf



* Bên cạnh đó, thực hiện sao chép Unicode mapping tập tin vào /etc/nginx/modsec:

| $ sudo cp /usr/local/src/ModSecurity/Unicode.mapping /etc/nginx/modsec/ |
| --- |



* Download OWASP Corerule Set:
* Download OWASP Core Rule Set from Github với wget command:

| $ wget https://github.com/coruleset/coreruleset/archive/v3.3.0.tar.gz |
| --- |

* Giải nén và di chuyển folder chứa Core Rule Set vào /usr/local/modsec:

| $ mv coreruleset 3.3.0 usr/local/modsecurity-crs |
| --- |

* Đổi tên crs-setup.conf.example thành crs-setup.conf:

| $ sudo mv /urs/local/modsecurity-crs/crs-setup.conf.example/usr/local/modsecurity-crs/crs-setup.conf |
| --- |

* Trở lại tập tin main.conf và thêm dòng như bên dưới:

Text

Description automatically generated

* Lưu và khởi động lại NGINX.

### Triển khai mô hình học máy vào NGINX

Trong phần này, nhóm tác giả sẽ thực hiện đưa trực tiếp mô hình máy học vào trong phần mã nguồn của Modsecurity-nginx connector.

Modsecurity-nginx connector là điểm kết nối giữa NGINX và libmodsecurity (Modsecurity v3). Nói cách khác, nó cung cấp một kênh liên lạc giữa NGINX và libmodsecurity.

#### Xác định vị trí thích hợp để triển khai mô hình học máy vào NGINX

Sau quá trình tìm hiểu về cách viết mô-đun trong NGINX, cũng như tìm hiểu qua về mã nguồn Modsecurity Nginx connector, nhóm tác giả nhận thấy rằng:

Trong tập tin ngx\_http\_modsecurity\_header\_filter.c, sau khi yêu cầu đã được xử lý bằng mô-đun của Modsecurity, kết quả đầu ra sẽ được chuyển tiếp đến Modsecurity-nginx connector, tại dòng 557, thực hiện gọi đến bộ lọc tiêu đề (header filter) tiếp theo bằng cách return ngx\_http\_next\_header\_filter(r);

Graphical user interface, text, application, email

Description automatically generated

Ý tưởng của nhóm là cần phải đưa được mô hình học máy vào vị trí có thể kiểm tra và chặn được yêu cầu độc hại trước khi nó có thể chuyển sang bộ lọc khác. Vậy nên, mã nguồn để tích hợp mô hình học máy sẽ được nhóm đưa vào trước hàm return trong tập tin trên.

Cụ thể, với r hiện là biến đang chứa kết quả sau khi đã được Modsecurity xử lý của yêu cầu hiện tại mà nhóm tác giả đang thực hiện kiểm tra, với r->headers\_out.status sẽ chứa trạng thái của yêu cầu hiện tại, nhóm sẽ kiểm tra giá trị của nó có bằng với “NGX\_HTTP\_FORBIDDEN” hay không. Nếu nó bằng thì thực hiện chuyển sang bộ lọc tiếp theo, bỏ qua sự kiểm tra của mô hình học máy, ngược lại, chuyển yêu cầu qua mô hình học máy để kiểm tra.

#### Phương pháp đưa Tensorflow vào NGINX

Mã nguồn NGINX được viết bằng ngôn ngữ C, TensorFlow cung cấp C API được định nghĩa trong c\_api.h, nên trước tiên chúng ta cần thực hiện cài đặt thư viện Tensorflow cho C.

Libtensoflow packages được xây dựng và tải lên GCS cho tất cả các nền tảng hỗ trợ.

Các nền tảng được hỗ trợ:

* Linux, 64bit, x86
* macOS, Version 10.12.26 (Sierra) hoặc cao hơn
* Windows 64-bit x8

Sau khi cài đặt tensorflow cho code C, trong tập tin mã nguồn mà chúng ta sẽ sử dụng để tải mô hình máy học thêm dòng bên dưới để có thể import thư viện tensorflow vào NGINX mô-đun:

| #include <tensorflow/c/c\_api.h> |
| --- |

Để có thể sử dụng được tensorflow trong NGINX, khi thực hiện chạy tập tin configure compile mã nguồn mô-đun NGINX thành file .so, chúng ta cần thêm option như bên dưới:

| --with-ld-opt=”-ltensorflow -ltensorflow\_framework” |
| --- |

Trong NGINX, --with-ld-opt : tham số đặt các tham số bổ sung sẽ được sử dụng trong quá trình liên kết.

#### Chuẩn bị trước khi tải mô hình học máy:

Khi lưu mô hình học máy, nó sẽ tạo một thư mục với một loạt các tệp bên trong nó. Về cơ bản, nó lưu trữ các trọng số và đồ thị của mô hình. Tensoflow bao gồm một công cụ để đi sâu vào các tệp này và kết nối đầu vào với đầu ra tensors. Nó được gọi là saved\_mode\_cli. Đây là một command line tool được cài đặt cùng khi cài đặt Tensoflow.

Cài đặt tensoflow trong môi trường conda riêng biệt:

| $ conda create -n tf pthon=3.7 tensorflow |
| --- |

Kích hoạt môi trường:

| $ conda activate tf |
| --- |

Lúc này có thể gọi saved\_mode\_cli thông qua command line. Chúng ta cần phải trích xuất tên biểu đồ cho đầu vào và đàu ra tensors và sử dụng những thông tin này sau đó khi chúng ta gọi C API:

| $ saved\_mode\_cli show –dir <path\_to\_saved\_model\_folder> |
| --- |

Khi chạy dòng lệnh này, nhóm tác giả sẽ nhận được đầu ra như bên dưới:

A picture containing text

Description automatically generated

Sử dụng tag-sets để đi sâu hơn vào biểu đồ tensor:

| saved\_model\_cli show --dir <path\_to\_saved\_model\_folder> --tag\_set serve |
| --- |

Text

Description automatically generated

Sử dụng “serving\_default” signature key để in ra nút tensor:

| saved\_model\_cli show --dir <path\_to\_saved\_model\_folder> --tag\_set serve --signature\_def serving\_default |
| --- |

Text

Description automatically generated

serving\_default\_input\_21 và StatefulPartitionedCall sẽ được sử dụng sau trong quá trình tải mô hình. (1)

#### Tải mô hình học máy vào NGINX

Tiếp theo, nhóm tác giả sẽ tải mô hình đã lưu và phiên (session) sử dụng **TF\_LoadSessionFromSavedModel** API.

* Tạo một Graph (đồ thị):

Một biểu đồ tính toán đơn giản được khởi tạo:

| TF\_Graph\* Graph = TF\_NewGraph(); |
| --- |

Tensoflow cũng cung cấp một cấu trúc dữ liệu tiện dụng để gỡ lỗi:

| TF\_Status\* Status = TF\_NewStatus(); |
| --- |

* Chuẩn bị Session:

Một phiên là giao diện người dùng của Tensoflow có thể được sử dụng để thực hiện các tính toán. Trong trường hợp của nhóm, nó sẽ trả về các dự đoán cho đầu vào.

Đầu tiên tạo cấu trúc cần thiết cho các tùy chọn:

| TF\_SessionOptions\* SessionOpts = TF\_NewSessionOptions();  TF\_Buffer\* RunOpts = NULL; |
| --- |

Tiếp theo, cung cấp các thông tin cần thiết về mô hình để thực hiện tải:

| const char\* saved\_model\_dir = <path\_to\_our\_model>;  const char\* tags = "serve";  int ntags = 1; |
| --- |

Cuối cùng, khởi tạo phiên:

| TF\_Session\* Session = TF\_LoadSessionFromSavedModel(SessionOpts, RunOpts, saved\_model\_dir, &tags, ntags, Graph, NULL, Status); |
| --- |

Về mặt kỹ thuật, phiên đã được khôi phục từ mô hình đã được lưu. Đến đây, lưu ý rằng phiên đã được tạo nhưng chưa sẵn sàng để chấp nhận bất cứ đầu vào nào.

#### Phương pháp đưa đầu vào và lấy kết quả đầu ra của mô hình đã đào tạo

* Định nghĩa đầu vào:

Thông tin đầu vào bao gồm: tên nút, kiểu dữ liệu và hình dạng của tensor.

Trong trường hợp của nhóm tác giả, biểu đồ nhận một tensor làm đầu vào:

| int NumInputs = 1;  TF\_Output\* Input = (TF\_Output\*)malloc(sizeof(TF\_Output) \* NumInputs); |
| --- |

Cho biết những nút nào trên biểu đồ sẽ chấp nhận đầu vào, đây là lúc cần sử dụng dữ liệu đã lấy được ở (1), serving\_default\_input\_21 là đầu vào, đồng thời thực hiện đăng kí đầu vào:

| TF\_Output t0 = {TF\_GraphOperationByName(Graph, "serving\_default\_input\_21"), 0};  if(t0.oper == NULL)          printf("ERROR: Failed TF\_GraphOperationByName serving\_default\_input\_21\n");      else      printf("TF\_GraphOperationByName serving\_default\_input\_21 is OK\n");    Input[0] = t0; |
| --- |

* Định nghĩa đầu ra:

Tương tự như đầu vào, nhóm cần phải cho chương trình biết nút nào trong biểu đồ sẽ xuất ra. Trong trường hợp của nhóm tác giả, biểu đồ của nhóm sẽ có một nút đầu ra:

| int NumOutputs = 1;  TF\_Output\* Output = (TF\_Output\*)malloc(sizeof(TF\_Output) \* NumOutputs);  TF\_Output t2 = {TF\_GraphOperationByName(Graph, "StatefulPartitionedCall"), 0};  if(t2.oper == NULL)          printf("ERROR: Failed TF\_GraphOperationByName StatefulPartitionedCall\n");  else      printf("TF\_GraphOperationByName StatefulPartitionedCall is OK\n");    Output[0] = t2; |
| --- |

* Chạy phiên:

Chạy biểu đồ tính toán trên đầu vào được cung cấp:

| TF\_SessionRun(Session, NULL, Input, InputValues, NumInputs, Output, OutputValues, NumOutputs, NULL, 0,NULL , Status);  if(TF\_GetCode(Status) == TF\_OK)  {     printf("Session is OK\n");  }  else  {     printf("%s",TF\_Message(Status));  } |
| --- |

Cuối cùng, lấy giá trị đầu ra từ tensor output bằng cách sử dụng TF\_TensorData để trích xuất dữ liệu từ đối tượng tensor.

| void\* buff = TF\_TensorData(OutputValues[0]);  float\* offsets = (float\*)buff; |
| --- |

Đầu ra của mô hình sẽ sử dụng 2 giá trị là offsets[0] và offsets[1] với offset[0] là phần trăm ra 0 (yêu cầu bất thường) và offsets[1] là phần trăm ra 1 (yêu cầu bình thường). Nhóm tác giả sẽ thực hiện so sánh 2 giá trị trên để đưa ra quyết định xem có thực hiện chặn yêu cầu hay không.

| if(offsets[0] > offsets[1])  {     //ngx\_log\_error(NGX\_LOG\_DEBUG\_HTTP, r->connection->log, 0,"block by module A");      return NGX\_HTTP\_FORBIDDEN;  } |
| --- |

#### Biên dịch mô-đun sau khi đã tích hợp Machine Learning model

Sau khi thực hiện thêm đoạn mã được sử dụng để tải mô hình máy học và lấy kết quả đầu ra từ mô hình máy học, bước tiếp theo chúng ta cần phải thực hiện biên dịch lại mã nguồn của Modsecurity-nginx connector và thực hiện khởi động lại NGINX.

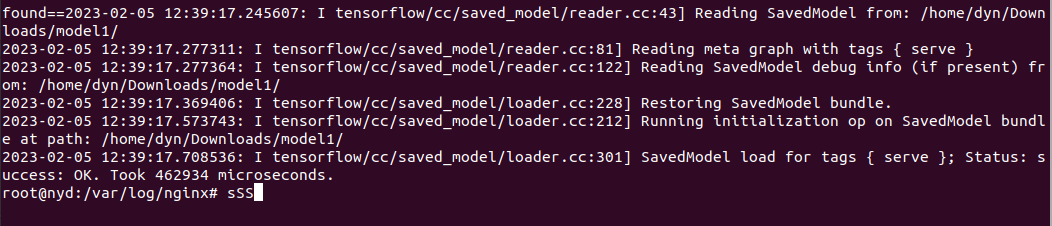
Chạy các lệnh dưới đây để biên dịch lại mã nguồn của Modsecurity-nginx connector:

| $ sudo ./configure –with-compat –with-ld-opt=”-ltensorflow -ltensorflow\_framework” –add-dynamic-module=/usr/local/src/ModSecurity-nginx  $ sudo make module  $ sudo cp objs/ngx\_http\_modsecurity\_module.so /usr/share/nginx/modules/ |
| --- |

Khởi động lại NGINX:

| $ sudo systemctl restart nginx |
| --- |

Kiểm tra lại trong file error.log của NGINX, chúng ta sẽ thấy thông tin rằng mô hình đã được tải thành công:



Hình 15: Mô hình được tải thành công vào NGINX

### Triển khai các mô hình khác

#### LSTM

Diagram

Description automatically generated with medium confidence

Hình 17: Mô hình LSTM

#### CNN-LSTM

Diagram, table

Description automatically generated

Hình 19: Mô hình CNN-LSTM

## Thực nghiệm

### Tiêu chí đánh giá

Sau khi thảo luận, nhóm tác giả quyết định chia các bộ dataset theo tỉ lệ 80-20 (80% dành cho huấn luyện, 20% dành cho thử nghiệm sau huấn luyện). Vì thế, tiêu chí đánh giá dựa trên phần trăm chính xác (**ACC**) sau khi dự đoán trên bộ thử nghiệm. Giá trị độ chụm (precision), khả năng thu hồi (recall) cũng là những chỉ số hiệu suất điển hình để đánh giá các mô hình được sử dụng trong học máy hoặc học sâu. Nhóm tác giả sử dụng **TP** đại diện cho “true positive”, **TN** đại diện cho “true negative”, **FP** đại diện cho “false positive”, **FN** đại diện cho “false negative”.

Khả năng thu hồi (**TPR**) là tỷ lệ các cuộc tấn công thực sự được phát hiện là bất thường trên tất cả các cuộc tấn công được định nghĩa như sau:

Chỉ số **FPR** (False Positive Rate) là tỷ lệ các yêu cầu thông thường được phát hiện là tấn công trên tất cả các yêu câu thông thường:

Độ chính xác (**Precision**) là tỉ lệ dự đoán tấn công đúng:

Đặc biệt, nhóm tác giả xác định tỉ lệ những yêu cầu bình thường được xác định là yêu cầu bình thường (**DRN**):

### Kết quả thực nghiệm trên Python

Dựa theo mô hình nhóm tác giả đề xuất, dưới đây là kết quả sau khi thử nghiệm trên 20% của mỗi bộ dataset:

**BẢNG 4: KẾT QUẢ DỰ ĐOÁN CỦA MÔ HÌNH TRÊN BỘ DỮ LIỆU**

|  | ACC | TPR | DRN |
| --- | --- | --- | --- |
| CSIC2010 | 0.8402 | 0.7887 | 0.9297 |

### So sánh kết quả thực nghiệm với các mô hình khác

**BẢNG 5: KẾT QUẢ SO SÁNH GIỮA MÔ HÌNH ĐỀ XUẤT VÀ CÁC MÔ HÌNH KHÁC**

| Mô hình | ACC (Độ chính xác) | TPR (Độ phủ/Recall) | DRN (Tỷ lệ phát hiện bình thường) |
| --- | --- | --- | --- |
| LSTM | 0.5508 | 0.8623 | 0.0082 |
| CNN-LSTM | 0.8382 | 0.7666 | 0.9628 |
| Bi-LSTM | 0.8402 | 0.7887 | 0.9297 |

Có thể thấy dựa trên bảng 5, mô hình được nghiện cứu nhóm tác giả đề xuất chưa thực sự cho ra kết quả tốt nhất ở bộ dữ liệu CSIC2010 như các bài nghiên cứu đã và đang tham khảo, tuy nhiên với độ chính xác như trên hoàn toàn có thể đưa mô hình này để sử dụng thực tế.

## Chạy thực nghiệm trên NGINX

Dưới đây là kết quả nhóm tác giả chạy thực nghiệm trên NGINX. Với bộ dataset CSIC2010 nhóm tác giả sử dụng 20% bộ dataset để chạy thực nghiệm.

Nhóm tác giả thực hiện chạy thực nghiệm thực hiện kết hợp Modsecurity cùng mô hình máy học do nhóm tác giả phát triển.

### Trường hợp WAF tích hợp Modsecuirty và mô hình Bidirectional-LSTM

#### Bộ CSIC2010

Hình ảnh bên dưới là kết quả chạy thực nghiệm trên 20% dữ liệu **bất thường** của bộ **CSIC2010** trong trường hợp kết hợp Modsecurity **và mô hình học máy**:

Text

Description automatically generated

Hình 26: Thực nghiệm trên 20% bộ dữ liệu bất thường CSIC sử dụng mô-đun mới

* Tổng số yêu cầu bất thường đã gửi: 2176
* Số yêu cầu bất thường mô-đun mới có thể phát hiện được: 1857

Hình ảnh bên dưới là kết quả chạy thực nghiệm trên 20% dữ liệu **bình thường** của bộ **CSIC2010** trong trường hợp kết hợp Modsecurity và mô hình học máy:

Text

Description automatically generated

Hình 27: Thực nghiệm trên 20% bộ dữ liệu bất thường CSIC sử dụng mô-đun mới

* Tổng số yêu cầu bình thường đã gửi: 7199
* Số yêu cầu bình thường bị mô-đun mới nhận định là bất thường: 994

**BẢNG 7: CÁC CHỈ SỐ TRÊN DỮ LIỆU THỬ NGHIỆM KHI ĐÃ TÍCH HỢP MÔ HÌNH HỌC MÁY**

|  |  | CSIC2010 |
| --- | --- | --- |
| Thực nghiệm trên yêu cầu bất thường | TP + FN | 2176 |
| TP | 1857 |
| FP | 994 |
| Thực nghiệm trên cầu bình thường | TN + FP | 7199 |
| TN | 6205 |
| FN | 319 |
| Đánh giá | ACC | 0.8599 |
| TPR | 0.8534 |
| DRN | 0.8619 |

### Kết luận

Sau quá trình thực nghiệm trên bộ dataset CSIC2010, nhóm nhận thấy rằng, mô-đun mới (mô đun kết hợp giữa Modsecurity và mô hình học máy) đã hoạt động tốt trong đa số các trường hợp trong việc phát hiện tấn công.

Tuy nhiên bên cạnh đó, dựa vào việc đo thời gian trung bình của một yêu cầu trong trường hợp có sử dụng mô hình học máy. Kết quả nhóm nhận được cho thấy thời gian trung bình trả về của một yêu cầu khi sử dụng mô hình học máy là khá lâu so với trường hợp không sử dụng mô hình học máy, mỗi yêu cầu thường phải mất hơn 2 giây trong khi đó trong trường hợp chỉ sử dụng Modsecurity mỗi yêu cầu chỉ mất gần 0.2 giây.

**BẢNG 8: SO SÁNH KẾT QUẢ THỰC NGHIỆM GIỮA MODSECURITY VÀ MÔ-ĐUN MỚI**

| Mô-đun | Dataset | ACC | TPR | DRN | Avarage Time (s) |
| --- | --- | --- | --- | --- | --- |
| Modsecurity tích hợp mô hình học máy | CSIC2010 | 0.8599 | 0.8534 | 0.8619 | 2.253 |

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

**Tóm tắt chung**

Ở chương cuối cùng, nhóm tác giả hệ thống lại những gì đã làm được trong đề tài này, đồng thời đưa ra những mục tiêu và các hướng phát triển để có thể cải thiện kết quả, tăng thêm tính khả quan của đề tài khi áp dụng vào thực tiễn.

## Kết luận

Trong những năm gần đây, việc tích hợp học máy vào mô-đun tường lửa trong NGINX là một cách thức mới trong việc tăng khả năng ngăn chặn tấn công từ mạng lưới bên ngoài. Vì cách thức tấn công ngày càng đa dạng, phức tạp và không đồng nhất nên việc tích hợp một mô-đun học máy để hỗ trợ trong việc phát hiện và ngăn chặn tấn công cũng giúp cho máy chủ được tăng khả năng an toàn hơn.

Việc phát triển các mô hình học máy yêu cầu một số lượng lớn dữ liệu huấn luyện từ nhiều nguồn khác nhau. Tuy nhiên dữ liệu được công khai trên mạng còn khá hạn chế, thậm chí còn lỗi thời theo thời gian, dẫn đến khả năng phát hiện tấn công nếu đưa vào thực tế không tốt như thử nghiệm

## Thuận lợi và khó khăn

Trong suốt thời gian thực hiện đề tài, nhóm luôn nhận được sự hướng dẫn, quan tậm tận tình từ phía giảng viên hướng dẫn. Thầy luôn theo sát công việc thực hiện đề tài của nhóm, chỉ ra những chỗ cần phải khắc phục, đưa ra cho nhóm phương pháp giải quyết vấn đề khi nhóm rơi vào khó khăn, đây là một thuận lợi cũng là một may mắn to lớn mà nhóm có được.

Bên cạnh đó, nhờ vào sự tài trợ cơ sở vật chất thiết bị từ phía Phòng thí nghiệm An toàn thông tin, nhóm mới có thể có đủ điều kiện, dễ dàng hơn trong việc thực hiện đề tài của mình.

Đề tài được phát triển từ Đồ án chuyên ngành của nhóm nên khi bắt tay vào làm đề tài nhóm cũng đã trang bị được cho mình một số kiến thức cơ bản và nền tảng, giúp phần nào giúp tiếp kiệm được thời gian cho quá trình thực hiện đề tài.

Trong quá trình làm đề tài, nhóm đã không ngừng tiếp thu những kiến thức mới, cố gắng hoàn thiện mô hình để có thể cho ra kết quả tốt nhất. Tuy nhiên do kiến thức còn hạn hẹp nên nhóm không thể tránh khỏi những sai sót trong khi thực hiện đề tài, vậy nên nhóm rất mong chờ và trân quý những góp ý quý báu từ quý thầy cô.

## Kết quả đạt được

Trong đề tài này, nhóm tác giả đã đề xuất phát triển WAF kết hợp giữa WAF truyền thống và phương pháp học máy để phát hiện các cuộc tấn công mới. Kết quả đạt được cho thấy nhóm đã bước đầu thành công trong việc hiện thực hóa hệ thống WAF nói trên

Nhóm đã thực hiện thu thập dữ liệu và thực hiện huấn luyện trên các mô hình học máy, lựa chọn ra phương pháp phù hợp để đưa ra một mô hình hoạt động hiệu quả trong việc phát hiện những yêu cầu bất thường.

Tiếp theo đó từ việc tìm hiểu cơ sở lý thuyết về Tensoflow cho C API và các bài viết liên quan, nhóm đã thành công trong việc đưa Tensoflow vào NGINX. Theo sau đó là thực hiện tải mô hình đã huấn luyện và thực hiện lấy kết quả đầu ra từ mô hình thành công.

Bên cạnh đó, để có thể kết hợp được Modsecurity với mô hình do nhóm đã đào tạo, nhóm đã thực hiện nghiên cứu mã nguồn của NGINX, Modsecurity-nginx connector, và cách hoạt động của một mô-đun trong NGINX để có thể biên dịch được một mô-đun mới kết hợp giữa Modsecurity và mô hình máy học.

Cuối cùng, để cho mô hình máy học có thể xử lý được yêu cầu trong NGINX, nhóm đã thành công trong việc thực hiện các giai đoạn tiền xử lý dữ liệu trực tiếp trong mã nguồn của Modsecurity-nginx connector đổi với các yêu cầu trước khi đưa vào mô hình máy học.

Trải qua tất cả các bước này, nhóm đã có thể đưa ra một mô-đun là WAF kết hợp giữa WAF truyền thống và mô hình học máy, thành công trong việc phát hiện và chặn những yêu cầu độc hại mới.

Trải qua quá trình thực nghiệm và đánh giá, nhóm đã thu được một số kết quả như sau:

* Mô hình học máy do nhóm phát triển tuy chưa thể được coi là một mô hình có hiệu quả vượt trội tuy nhiên là một mô hình tương đối có thể đưa vào sử dụng trong thực tế.
* Nhóm đã chứng minh được, mô hình sử dụng mô-đun mới (WAF kết hợp Modsecurity và mô hình học máy) do nhóm phát triển có thể phát hiện và chặn được yêu cầu độc hại mà Modsecurity chưa phát hiện được.
* Khi so sánh về tỉ lệ phát hiện yêu cầu bất thường thì mô-đun mới cho ra kết quả tốt hơn, phát hiện được nhiều cuộc tấn công hơn.

## Hướng phát triển

Với việc đạt được kết quả khả quan, tuy nhiên, nghiên cứu trên của nhóm tác giả vẫn còn một vài hạn chế như sau:

* Bộ dữ liệu huấn luyện còn hạn chế, điều này dẫn đến có sai sót tương đối trên thực nghiệm.
* Thời gian để tìm hiểu và phát triểu mô hình học sâu chưa thực sự nhiều, dẫn đến mô hình vẫn còn chưa phức tạp.
* Cần tối ưu hơn thuật toán được sử dụng cho việc tiền xử lý dữ liệu trong NGINX, có thể xử lý được tất cả các trường hợp có thể xảy ra với các loại yêu cầu khác nhau. Tránh được nhiều nhất có thể các trường hợp lỗi trong việc xử lý chuỗi trong NGINX.
* Hiện tại mô hình học máy được đưa vào NGINX có thời gian chạy khá lâu, trong tương lai nhóm mong muốn có thể tìm ra được phương pháp tối ưu hóa thời gian chạy.
* Hiện tại nhóm chưa thể tách được dữ liệu tốt (good data) và dữ liệu không nhận dạng (unknown data) đi ra từ Modsec, do đó hiện tại cả dữ liệu tốt và dữ liệu không nhận dạng của Modsecurity đều được chuyển tới mô hình học máy để kiểm tra. Nhóm sẽ tiếp tục phát triển phương pháp để có thể tách riêng hai loại dữ liệu này, từ đó chỉ là dữ liệu không nhận dạng là sẽ đi qua mô hình học máy để kiểm tra.

Về hướng phát triển lâu dài, nhóm tác giả đề xuất việc thu thập và làm giàu bộ dataset hơn để tăng độ chính xác của mô hình huấn luyện, và trong tương lai, nhóm tác giả sẽ cố gắng cải tiến mô hình học máy để tăng độ chính xác trong việc thực nghiệm và sử dụng thực tế hơn. Bên cạnh đó, tăng hiệu suất khi tải mô hình học máy, giảm được thời gian chạy của mỗi yêu cầu.

TÀI LIỆU THAM KHẢO

# References

| [1] | H. T. Nguyen and K. Franke, "Adaptive Intrusion Detection System via Online Learning," Proc. 12th Int. Conf. Hybrid, 2012. |
| --- | --- |
| [2] | J. Liang, W. Zhao and W. Ye, "Anomaly-Based Web Attack Detection: A Deep Learning Approach," VI International Conference on Network, Communication and Computing, 2017. |
| [3] | Z. Tian, C. Luo, J. Qiu, X. Du and M. Guizani, "A Distributed Deep Learning System for Web Attack Detection on Edge Devices". |
| [4] | M. S. Hacer Karacan, "A Novel Data Augmentation Technique and Deep Learning Model for Web Application Security," 08 11 2021 . [Online]. Available: https://ieeexplore.ieee.org/document/9605636. |
| [5] | A. Joulin, E. Grave, P. Bojanowski and T. Mikolov, "Bag of Tricks for Efficient Text Classification," arXiv:1607.01759 . |
| [6] | A. Tekerek, "A novel architecture for web-based attack detection using convolutional neural network," Comput. Secur, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404820303692. |
| [7] | D. Madhugiri, "Visualize Deep Learning Models using Visualkeras," [Online]. Available: https://www.analyticsvidhya.com/blog/2022/03/visualize-deep-learning-models-using-visualkeras/. [Accessed 8 March 2022]. |
| [8] | E. BARRIL, "Text Classification using Neural Networks," 2019. [Online]. Available: https://www.kaggle.com/code/eliotbarr/text-classification-using-neural-networks#A-complex-model-:-LSTM. |
| [9] | Z. Wang, X. Ren and S. Li, "A Malicious URL Detection Model Based on Convolutional Neural Network," 2021. |
| [10] | H. Liu, B. Lang, M. Liu and H. Yan, "CNN and RNN based payload classification methods for attack detection," Knowledge-Based Systems, 2019. |
| [11] | M. Ito and H. Iyatomi, "Web application firewall using character-level convolutional neural network," Penang, Malaysia, 2018. |
| [12] | T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space," 2013. |
| [13] | KERAS, "Keras API reference," [Online]. Available: https://keras.io/api/. |
| [14] | P. R. McWhirter, K. Kifayat, Q. Shi and B. Askwith, "SQL Injection Attack classification through the feature extraction of SQL query strings using a Gap-Weighted String Subsequence Kernel," [Online]. Available: https://doi.org/10.1016/j.jisa.2018.04.001. |
| [15] | MR\_KNOWNOTHING, "Deep Learning For NLP: Zero To Transformers & BERT," 2020. [Online]. Available: https://www.kaggle.com/code/tanulsingh077/deep-learning-for-nlp-zero-to-transformers-bert. |
| [16] | P. MEGRET, "Gensim Word2Vec Tutorial," 2019. [Online]. Available: https://www.kaggle.com/code/pierremegret/gensim-word2vec-tutorial. |
| [17] | SpiderLabs, [Online]. Available: https://github.com/SpiderLabs/ModSecurity-nginx. |
| [18] | Emiller, "Emiller’s Guide To Nginx Module Development," [Online]. Available: https://www.evanmiller.org/nginx-modules-guide.html. |
| [19] | AmirulOm, "Deploying Tensorflow as C/C++ executable," [Online]. Available: https://github.com/AmirulOm/tensorflow\_capi\_sample. |
| [20] | "Install TensorFlow for C," [Online]. Available: https://www.tensorflow.org/install/lang\_c. |
| [21] | V. Dovgalecs, "Undocumented Tensorflow C API," [Online]. Available: https://medium.com/@vladislavsd/undocumented-tensorflow-c-api-b527c0b4ef6. |
| [22] | O. G. o. F5, "Compiling Third-Party Dynamic Modules for NGINX and NGINX Plus," [Online]. Available: https://www.nginx.com/blog/compiling-dynamic-modules-nginx-plus/. |
| [23] | J. Kiarie, "How to Install ModSecurity for Nginx on Debian/Ubuntu," [Online]. Available: https://www.tecmint.com/install-modsecurity-nginx-debian-ubuntu/. |
| [24] | [Online]. Available: https://stackoverflow.com/questions/32991429/compiling-nginx-module-with-dependant-library. |