**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

---


**Đỗ Minh Hải**

# PHÁT HIỆN TẤN CÔNG ỨNG DỤNG WEB DỰA TRÊN LOG TRUY CẬP SỬ DỤNG BỘ PHÂN LỚP RỪNG NGẪU NHIÊN

**LUẬN VĂN THẠC SĨ KỸ THUẬT**

**(Theo định hướng ứng dụng)**

HÀ NỘI - 2019

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

---

**Đỗ Minh Hải**

# PHÁT HIỆN TẤN CÔNG ỨNG DỤNG WEB DỰA TRÊN LOG TRUY CẬP SỬ DỤNG BỘ PHÂN LỚP RỪNG NGẪU NHIÊN

**CHUYÊN NGÀNH : HỆ THỐNG THÔNG TIN**
**MÃ SỐ: 8.48.01.04**

**LUẬN VĂN THẠC SĨ KỸ THUẬT**
**(Theo định hướng ứng dụng)**

**NGƯỜI HƯỚNG DẪN KHOA HỌC:**
**TS. NGUYỄN NGỌC ĐIỆP**

HÀ NỘI – NĂM 2019

MỤC LỤC

# LỜI CAM ĐOAN

Tôi cam đoan đây là công trình nghiên cứu của riêng tôi.

Các số liệu, kết quả nêu trong luận văn là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Tác giả

Đỗ Minh Hải

# LỜI CẢM ƠN

Lời đầu tiên tôi xin được gửi lời biết ơn chân thành và sâu sắc nhất tới thầy giáo TS. Nguyễn Ngọc Điệp – Khoa Công nghệ thông tin – Học viện Bưu chính Viễn thông, người thầy đã luôn tận tình chỉ bảo, giúp đỡ, hướng dẫn tôi trong suốt quá trình nghiên cứu luận văn này.

Tôi chân thành cảm ơn các thầy, cô giáo trong Khoa Công nghệ thông tin – Học viện Bưu chính Viễn thông đã luôn tận tâm truyền dạy cho tôi những kiến thức bổ ích trong thời gian học tập và nghiên cứu tại nhà trường.

Tôi cũng xin gửi lời cám ơn tới các bạn cùng lớp đã giúp đỡ tôi trong quá trình học tập tại trường.

Học viên

Đỗ Minh Hải

# DANH MỤC CÁC HÌNH VẼ

# MỞ ĐẦU

## 1. Lý do chọn đề tài

Hiện nay, với tốc độ phát triển về công nghệ tin học, truyền thông, thương mại điện tử thì nhu cầu đăng tải, chia sẻ thông tin trên các hệ thống web là rất lớn. Các doanh nghiệp đều sở hữu, sử dụng các ứng dụng web như: webmail, kênh bán hàng trực tuyến, đấu giá, mạng xã hội và nhiều chức năng khác để cung cấp dịch vụ trực tuyến, kết nối với khách hàng, đối tác. Mỗi phút trôi qua lại có một lượng thông tin khổng lồ được đăng tải trên các ứng dụng web, để cung cấp, truyền tải cho người dùng truy cập khai thác. Điều đó dẫn kến nhiều nguy cơ về về sự mất an toàn thông tin, đòi hỏi cần phát triển các công tục hỗ trợ để tăng tính an toàn, bảo mật đối với thông tin được truyền tải trên mạng.

Thực tế, mọi ứng dụng web vẫn luôn tiềm ẩn những nguy cơ mất an toàn thông tin do rất nhiều nguyên nhân kỹ thuật, cả chủ quan cũng như khách quan gây mất mát dữ liệu có giá trị, hay làm gián đoạn việc cung cấp dịch vụ. Việc triển khai trực tuyến ứng dụng web sẽ cho phép người dùng quyền truy cập tự do vào ứng dụng thông qua giao thức HTTP/HTTPS, những truy cập này có khả năng vượt qua hệ thống firewall, các lớp bảo vệ hệ thống và các hệ thống phát hiện xâm nhập vì các mã tấn công đều nằm trong các gói giao thức HTTP hợp lệ, kể cả các ứng dụng Web có độ bảo mật cao sử dụng SSL cũng đều cho phép tất cả các dữ liệu đi qua mà không hề kiểm tra tính hợp lệ của dữ liệu. Các ứng dụng web vẫn luôn tiềm ẩn những lỗ hổng bảo mật do mã nguồn, máy chủ…

Bên cạnh đó, việc tấn công xâm nhập các ứng dụng web của harker ngày càng trở nên đa dạng và vô cùng tinh vi. Tuy nhiên, người quản trị có thể phát hiện được những truy cập bất thường dựa vào cơ chế ghi nhận và lưu trữ tất cả truy cập đến máy chủ web thông qua logfile của máy chủ web. Bằng việc thu thập, phân tích tài nguyên này có thể phát hiện được những truy cập bất thường để chủ động phòng ngừa, ngăn chặn những nguy cơ trong tương lai đối với hệ thống.

Trong phạm vi của luận văn này, tác giả lựa chọn đề tài ***Phát hiện tấn công ứng dụng web dựa trên log truy cập sử dụng bộ phân lớp rừng ngẫu nhiên*** để

nghiên cứu xây dựng, đánh giá mô hình và thử nghiệm kết quả.

## 2. Tổng quan về vấn đề nghiên cứu

Cho đến nay, nhiều hãng công nghệ của Thế giới cũng như Việt Nam đưa ra các giải pháp hỗ trợ an toàn, bảo mật mạng, đã hạn chế và ngăn chặn rất nhiều các cuộc tấn công nhằm vào mạng của các đơn vị, doanh nghiệp. Ví dụ như các phần mềm bảo mật, các chương trình diệt virus với cơ sở dữ liệu các mẫu virus liên tục cập nhật hay hệ thống firewall nhằm ngăn chặn những kết nối không tin cậy, thực hiện mã hóa làm tăng an toàn cho dữ liệu được truyền tải trên mạng. Tuy nhiên, các hình thức phá hoại ứng dụng web ngày càng trở nên tinh vi hơn, phức tạp hơn, có thể vượt qua được các công cụ và phần mềm bảo mật có sẵn. Vì vậy, vẫn cần nghiên cứu thêm các giải pháp hỗ trợ để phát hiện được tối đa những tấn công đang diễn ra trong hệ thống mạng để phòng ngừa, hạn chế những thiệt hại cho người dùng, doanh nghiệp.

Với các máy chủ web, việc thu thập, phân tích các log truy cập là cơ chế quan trọng không thể thiếu, nó sẽ giúp tự động ghi nhận tất cả các truy cập gồm bình thường và bất thường đến ứng dụng web. Từ dữ liệu log thô thu thập được, qua quá trình xử lý, phân tích, người quản trị hệ thống có thể trích xuất được các thông tin quan trọng về các hành vi người dùng trực tuyến, các dấu hiệu truy cập bất thường, các dạng mã độc và các dạng tấn công, xâm nhập để giúp người quản trị quyết định áp dụng các phương án phòng ngừa, hoặc đưa ra các cảnh báo về nguy cơ mất an toàn thông tin đối với hệ thống cho người dùng. Đồng thời cũng như là căn cứ giúp cải thiện chất lượng hệ thống và các dịch vụ đáp ứng tốt hơn nhu cầu người dùng.

Có nhiều phương pháp phân tích log đã được nghiên cứu và triển khai, tuy nhiên việc áp dụng bộ phân lớp rừng ngẫu nhiên để phân tích phát hiện tấn công chưa được sử dụng phổ biến. Vì vậy tác giả lựa chọn sử dụng phương pháp học máy có giám sát, áp dụng bộ phân lớp rừng ngẫu nhiên để phân tích các weblog nhằm phát hiện các truy cập bất thường, giúp người quản trị sớm có biện pháp phòng chống, ngăn chặn các nguy cơ có thể mất an toàn thông tin.

## 3. Mục đích nghiên cứu

Nghiên cứu phương pháp và xây dựng mô hình học máy để phát hiện các tấn công đến ứng dụng web dựa trên log truy cập. Kết quả nghiên cứu sẽ góp phần giúp cho người quản lý website đánh giá và ngăn ngừa được một số hình thức tấn công phổ biến, có thể đưa ra giải pháp tăng cường các lỗ hổng, nguy cơ tiềm ẩn.

## 4. Đối tượng và phạm vi nghiên cứu

Đối tượng phân tích là các log file truy cập được tạo ra trên máy chủ web như Apache, Nginx, IIS thông qua luồng mạng pcap.

## 5. Phương pháp nghiên cứu

Đọc và nghiên cứu tổng quan lý thuyết về ứng dụng web, lý thuyết học máy có giám sát, giải thuật bộ phân lớp rừng ngẫu nhiên.

Xây dựng mô hình học máy phát hiện tấn công ứng dụng web, đánh giá mô hình, thử nghiệm hệ thống dựa trên dữ liệu đã thu thập.

Cài đặt hệ thống để đánh giá.

Cấu trúc của luận văn được tác giả tổ chức thành 4 chương như sau:

Phần 1 – Giới thiệu Cơ sở lý thuyết

1.1. Tổng quan về tấn công Web

1.2. Giới thiệu về Web log

1.3. Phương pháp phát hiện tấn công qua web log sử dụng học máy

Chương 2 – Phương pháp phát hiện tấn công

2.1. Phương pháp phát hiện tấn công

2.2. Tổng quan về thuật toán Random Forest

2.3. Tập dữ liệu huấn luyện (CSIC 2010)

2.4. Phương pháp đánh giá

2.5. Kết quả thử nghiệm

2.6. Kết luận chương

Chương 3: Xây dựng hệ thống thực nghiệm

3.1. Xây dựng hệ thống

3.2. Một số kết quả thử nghiệm hệ thống

Chương 4: Kết luận và kiến nghị

4.1. Những đóng góp của luận văn

4.2. Hướng phát triển luận văn

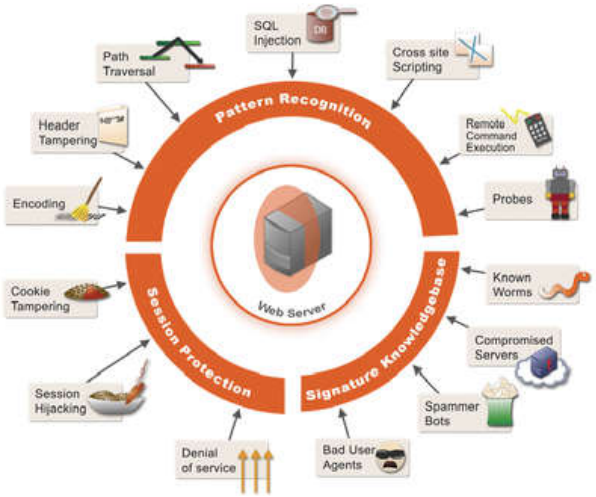
# CHƯƠNG 1 – CƠ SỞ LÝ THUYẾT

## 1.1. Tổng quan về tấn công Web.

Ngày nay, Web chính là kênh truyền thông cơ bản giúp doanh nghiệp tăng cường hình ảnh trực tuyến của mình trên thế giới mạng, giúp xây dựng, duy trì nhiều mối quan hệ với khách hàng tiềm năng. Với xu hướng phát triển công nghệ CNTT và truyền thông hiện nay, Web đã trở thành kênh bán hàng phổ biến đối với hàng nghìn doanh nghiệp lớn nhỏ. Đặc biệt website hiện nay cho phép đóng gói, xử lý, lưu trữ và truyền tải dữ liệu khách hàng với dữ liệu lớn, quan trọng và có giá trị (như thông tin cá nhân, mã số thẻ tín dụng, thông tin bảo mật xã hội …).

Chính những đặc điểm này, các website thường xuyên là mục tiêu tấn công của tin tặc để khai thác đánh cắp các thông tin quan trọng. Một trong những phương thức tấn công phổ biến là khai thác các lỗi bảo mật liên quan đến ứng dụng web. Nhiều điểm yếu nghiêm trọng hay các lỗ hổng cho phép hacker xâm nhập thẳng và truy cập vào cơ sở dữ liệu để trích xuất các dữ liệu nhạy cảm, quan trọng.

Các lỗi bảo mật ứng dụng web là nguyên nhân chủ yếu gây ra các lỗi đối với website. Các harker sau khi phát hiện được các lỗi này, thường sử dụng các kỹ thuật khác nhau để tiến hành xâm nhập, khai thác hoặc đánh sập cả hệ thống đích. Một số kỹ thuật thường được sử dụng như Buffer Overflows, SQL Injection, and Cross-site Scripting… Việc phân loại các kiểu tấn công thành các nhóm khác nhau sẽ giúp cho người quản trị xác định các nguy cơ cũng như biện pháp đối phó hiệu quả hơn.




Hình 1. 1. Một số phương thức tấn công ứng dụng web
(Nguồn: https://securitydaily.net/)

### *1.1.1. Một số khái niệm cơ bản về ứng dụng web*

*a. HTTP Request & HTTP Response*

HTTP header là phần đầu của thông tin mà trình khách và trình chủ gửi cho nhau. Những thông tin trình khách gửi cho trình chủ được gọi là HTTP requests (yêu cầu) còn trình chủ gửi cho trình khách là HTTP responses (phản hồi). Thông thường một HTTP header gồm nhiều dòng, mỗi dòng chứa tên tham số và giá trị. Một số tham số có thể được dùng trong cả Header yêu cầu và Header trả lời, còn số khác thì chỉ được dùng riêng trong từng loại.


Hình 1. 2 - HTTP Request

(Nguồn: http://www.tcpipguide.com/)

*HTTP Request:*

- Dòng đầu của HTTP Request là dòng Request-Line bao gồm các thông tin:

+ Method: là phương thức mà HTTP Request này sử dụng (POST, GET, HEAD, TRACE,...).

+ URI: là địa chỉ định danh của tài nguyên.

+ HTTP version: là phiên bản HTTP đang sử dụng

- Tiếp theo là các trường Header thông dụng như:

+ Accept: Loại nội dung có thể nhận được từ thông điệp phản hồi.

Ví dụ: text/plain,text/html,...

+ Accept-Encoding: Các kiểu nén được chấp nhận.

ví dụ: gzip, xz,...

+ User-Agent: Thông tin về trình duyệt của người dùng

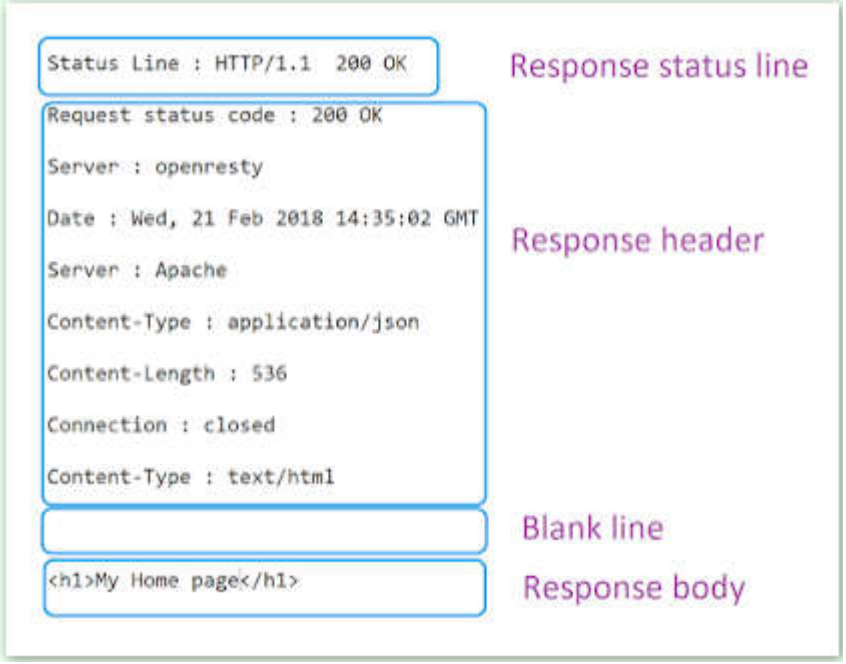
+ Connection: Tùy chọn cho kết nối hiện tại.

Ví dụ: closed, keep-alive, update,..

+ Cookie: Thông tin HTTP Cookie từ máy chủ

- Header của HTTP request sẽ kết thúc bằng một dòng trống

Cấu trúc của HTTP phản hồi gần giống với HTTP yêu cầu, chỉ khác nhau là thay vì Request-Line thì HTTP phản hồi có Status-Line.




Hình 1. 3- HTTP Response

(Nguồn: http://www.way2testing.com)

*HTTP Response:*

- Status-Line có phần chính như sau: HTTP-version là phiên bản HTTP cao nhất mà máy chủ đang hỗ trợ, Status-Code: mã kết quả trả về, Reason-Phrase: mô tả về Status-Code

- Tiếp theo là các tham số và kèm một dòng trống để báo hiệu kết thúc header

- Cuối cùng là phần thân của HTTP response

*b. Session*

Session là khoảng thời gian người sử dụng giao tiếp với một ứng dụng. Session bắt đầu khi người sử dụng truy cập vào ứng dụng lần đầu tiên, và kết thúc khi người sử dụng thoát khỏi ứng dụng. Mỗi session sẽ có một định danh (ID), mỗi session khác nhau sẽ có ID khác nhau. Trong ngữ cảnh ứng dụng web, website sẽ quyết định khi nào session bắt đầu và kết thúc. Trong một session, website có thể lưu trữ một số thông tin như đánh dấu bạn đã login hay chưa, những bài viết nào bạn đã đọc qua…

HTTP là giao thức hướng đối tượng tổng quát, phi trạng thái, nghĩa là HTTP không lưu trữ trạng thái làm việc giữa trình duyệt với trình chủ. Sự thiếu sót này gây khó khăn cho một số ứng dụng web, bởi vì trình chủ không biết trước đó trình duyệt đã có những trạng thái nào. Vì thế, để giải quyết vấn đề này, ứng dụng web đưa ra một khái niệm phiên làm việc (Session). Còn Session ID là một chuỗi để chứng thực phiên làm việc. Một số trình chủ sẽ cung cấp một Session ID cho người dùng khi họ xem trang web trên trình chủ.

Để duy trì phiên làm việc thì Session ID thường được lưu vào:

- Biến trên URL
- Biến ẩn form
- Cookie

Phiên làm việc chỉ tồn tại trong một thời gian cho phép, thời gian này được cấu hình quy định tại trình chủ hoặc bởi ứng dụng thực thi. Trình chủ sẽ tự động giải phóng phiên làm việc để khôi phục lại tài nguyên của hệ thống.

*c. Cookie*

Cookie là những phần dữ liệu nhỏ có cấu trúc được chia sẻ giữa trình chủ và trình duyệt của người dùng.

Các cookie được lưu trữ dưới những file dữ liệu nhỏ dạng text, được ứng dụng tạo ra để lưu trữ/truy tìm/nhận biết các thông tin về người dùng ghé thăm trang web và những vùng mà họ đi qua trong trang. Những thông tin này có thể được bao gồm tên/định danh người dùng, mật khẩu, sở thích, thói quen… cookie được trình duyệt của người dùng chấp nhận lưu trên đĩa cứng của máy tính, tuy nhiên không phải lúc

nào trình duyệt cũng hỗ trợ cookie, mà còn tùy thuộc vào người dùng có chấp nhận chuyện lưu trữ đó hay không. Ở những lần truy cập sau đến trang web đó, ứng dụng có thể dùng lại những thông tin trong cookie (như thông tin liên quan đến việc đăng nhập vào Facebook, gmail…) mà người dùng không phải làm lại thao tác đăng nhập hay cung cấp các thông tin khác.

Cookie được phân làm 2 loại secure/non-secure và persistent/non-persistent do đó ta sẽ có 4 kiểu cookie là:

- Persistent và Secure
- Persistent và Non-Secure
- Non-Persistent và Secure
- Non-Persistent và Non-Secure

Persistent cookie được lưu trữ dưới dạng tập tin .txt trên máy khách trong một khoảng thời gian xác định.

Non-Persistent cookie thì được lưu trữ trên bộ nhớ RAM của máy khách và sẽ bị hủy khi đóng trang web hay nhận được lệnh hủy từ trang web.

Secure cookie chỉ có thể được gửi thông qua HTTPS (SSL).

Non-Secure cookie có thể được gửi bằng cả hai giao thức HTTPS hay HTTP. Thực chất là đối với secure cookie thì trình chủ sẽ cung cấp chế độ truyền bảo mật.

Các thành phần của một cookie bao gồm:

| Domain | Flag | Path | Secure | Expiration | Name | Value |
|---|---|---|---|---|---|---|
| www.acb.vn | FALSE | / | FALSE | 1154029490 | Apache | 64.3.40.151.16018 996349247480 |

Domain: tên miền của trang web đã tạo cookie (ở trên là www. abc.vn)

Flag: mang giá trị TRUE/FALSE – Xác định các máy khác với cùng tên miền có được truy xuất đến cookie hay không.

Path: phạm vi các địa chỉ có thể truy xuất cookie. Ví dụ: Nếu path là “/tracuu” thì các địa chỉ trong thư mục /tracuu cũng như tất cả các thư mục con của nó như /tracuu/baomat có thể truy xuất đến cookie này. Còn nếu giá trị là “/” thì cookie sẽ được truy xuất bởi tất cả địa chỉ thuộc miền trang web tạo cookie.

Secure: mang giá trị TRUE/FALSE – Xác định đây là một secure cookie hay không, nghĩa là kết nối có sử dụng SSL hay không.

Expiration: thời gian hết hạn của cookie, được tính bằng giây kể từ 00:00:00 giờ GMT ngày 01/01/1970. Nếu giá trị này không được thiết lập thì trình duyệt sẽ hiểu đây là non-persistent cookie và chỉ lưu trong bộ nhớ RAM và sẽ xóa nó khi trình duyệt bị đóng.

Name: tên biến (trường hợp này là Apache)

Value: với cookie được tạo ở trên thì giá trị của Apache là 64.3.40.151.16018996349247480, của tên miền http://www.abc.com

Kích thước tối đa của cookie là 4kb. Số cookie tối đa cho một tên miền là 20 cookie. Cookie bị hủy ngay khi đóng trình duyệt gọi là "session cookie".

Một ví dụ về cookie: Giả sử lần đầu tiên bạn vào trang facebook.com thì máy tính của bạn sẽ tải trang này rất lâu vì nó phải tải nội dung trang web về máy của bạn. Sau khi đăng nhập vào hệ thống và sử dụng như bình thường. Sang ngày hôm sau, vào lại trang facebook.com thì vào rất nhanh và nhiều khi cũng không cần phải đăng nhập tài khoản nữa nguyên nhân chính là do trình duyệt đã lưu cookie các thông tin hôm qua bạn đã vào. Cookie là một cao dao hai lưỡi, lợi ích của nó thì bạn có thể thấy được sự tiện lợi là đỡ tốn thời gian tải lại trang web nhưng ngược lại nhược điểm của nó là các Hacker có thể dựa vào các file cookie để lấy các thông tin tài khoản. Rất là nguy hiểm nên tốt nhất không để trình duyệt lưu cookie nhưng đa số người dùng hiện nay đều để chế độ lưu cookie vì người dùng không biết đến sự nguy hiểm của nó hoặc là thấy nó tiện cho công việc của mình.

*d. Proxy*

Hiện nay, người dùng sử dụng Internet đa số là đi Internet trực tiếp nghĩa là người dùng tự mình đi đến máy chủ hỏi xin các yêu cầu. Đi trực tiếp như thế này thì có cái khuyết điểm là băng thông sẽ tốn rất nhiều. Chính vấn đề về băng thông nên mới ra đời khái niệm "proxy".

Proxy là một Internet server làm nhiệm vụ chuyển tiếp thông tin và kiểm soát tạo sự an toàn cho việc truy cập Internet của các máy khách, còn gọi là khách hàng

sử dụng dịch vụ Internet. Trạm cài đặt proxy gọi là proxy server. Proxy hay trạm cài đặt proxy có địa chỉ IP và một cổng truy cập cố định.

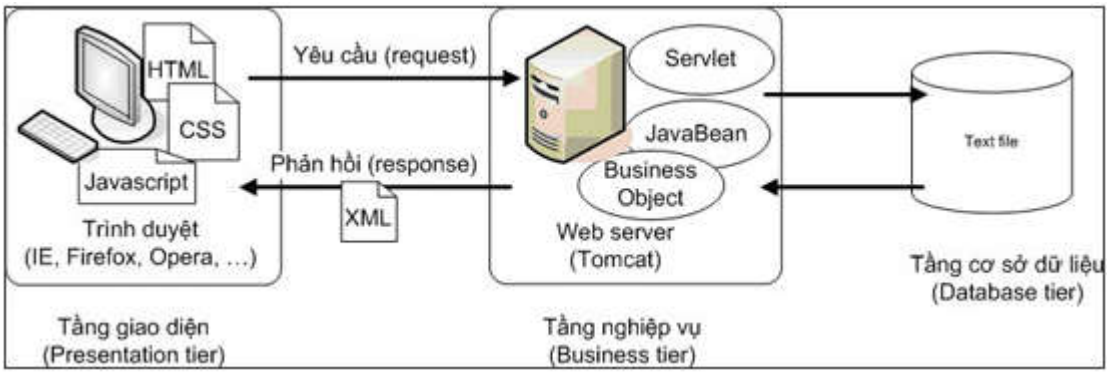
Proxy cung cấp cho người sử dụng truy xuất Internet những nghi thức đặc biệt. Những chương trình máy khách của người sử dụng sẽ qua trung gian máy chủ proxy thay thế cho máy chủ thật sự mà người sử dụng cần giao tiếp.

Máy chủ proxy xác định những yêu cầu từ client và quyết định đáp ứng hay không đáp ứng, nếu yêu cầu được đáp ứng máy chủ proxy sẽ kết nối với máy chủ thật thay cho máy khách và tiếp tục chuyển tiếp những yêu cầu từ máy khách đến máy chủ, cũng như trả lời của máy chủ đến máy khách. Vì vậy máy chủ proxy giống cầu nối trung gian giữa máy chủ và máy khách.

Thường thì máy chủ proxy được xây dựng chủ yếu là trong công ty hay các nhà cung cấp dịch vụ để phục vụ cho nhân viên hay là khách hàng của nhà cung cấp. Khuyết điểm lớn nhất mà proxy mắc phải là bảo mật vì nó làm trung gian nên nó có thể biết hết mọi thứ mà người dùng khai báo với máy chủ đích.

### *1.1.2. Kiến trúc của một ứng dụng web*

Một ứng dụng Web có đầy đủ các thành phần như sau:




Hình 1. 4 - Kiến trúc của một ứng dụng Web

(Nguồn: https://edu.com.vn)

- Trình khách (hay còn gọi là trình duyệt): Internet Explorer, Firefox, Chrome...
- Trình chủ: Apache, IIS,...
- Hệ quản trị cơ sở dữ liệu: SQL Server, MySQL, DB2, Acess...

- Tường lửa: Lớp rào chắn bên ngoài một hệ thống mạng, vai trò kiểm soát luồng thông tin giữa các máy tính
- Proxy xác định những yêu cầu từ trình khách và quyết định đáp ứng yêu cầu hay không, Proxy đóng vai trò cầu nối trung gian giữa máy chủ và máy khách

## 1.2. Giới thiệu về Web log file

Web Log là một hoặc nhiều file log được tạo và lưu trữ bởi một Web server, nó chứa tất cả các hành động mà người truy cập tác động lên trang web.

Một web log sẽ chứa các request tác động đến nó. Các thông tin về request, ví dụ như địa chỉ IP máy khách, ngày/giờ request, trang đã request, mã HTTP, thông tin người dùng,.. Các file này không thể truy cập bởi người dùng thông thường, chỉ dùng cho quản trị viên hoặc admin vì chứa các thông tin quan trọng.

Từ server log để tra cứu lưu lượng người dùng trong ngày, trong tuần, thông tin người dùng.

```
11.222.333.44 - - [11/Dec/2018:11:01:28 -0600] “GET /blog/page-address.htm HTTP/1.1”
200 182 “-” “Mozilla/5.0 Chrome/60.0.3112.113”
```

Hình 1. 5 - Một line của Apache log

Logfile ghi lại liên tục các thông báo về hoạt động của hệ thống hoặc của các dịch vụ được triển khai trên hệ thống, nó cung cấp thông tin cho phép người quản trị phân tích nguyên nhân gốc rễ của một vấn đề phát sinh, giúp cho việc khắc phục sự cố khi phát sinh nhanh chóng hơn, từ đó giúp cho việc dự đoán vấn đề với hệ thống webserver sớm hơn.

Các web server chuẩn như Apache, IIS tạo thông điệp ghi nhật ký theo một chuẩn chung (CLF – common log format)[4]. Tệp nhật ký CLF chứa các dòng thông điệp cho mỗi một gói HTTP request, cấu tạo như sau:

***Host Ident Authuser Date Request Status Bytes***

Trong đó:

- Host: Tên miền đầy đủ của client hoặc IP
- Ident: Nếu chỉ thị IdentityCheck được kích hoạt và client chạy identd, thì đây là thông tin nhận dạng được client báo cáo

- Authuser: Nếu URL yêu cầu xác thực HTTP thì tên người dùng là giá trị của mã thông báo này
- Date: Ngày và giờ yêu cầu
- Request: Dòng yêu cầu của client, được đặt trong dấu ngoặc kép (“”)
- Status: Mã trạng thái (gồm ba chữ số)
- Bytes: số bytes trong đối tượng trả về cho client, ngoại trừ các HTTP header

Mỗi HTTP request có thể chứa các các dữ liệu bổ sung như đường liên kết hoặc chuỗi ký tự của người dùng.

Nếu mã thông báo không có giá trị, thì mã thông báo được biểu thị bằng một dấu gạch ngang (-).

Ví dụ:

127.0.0.1 - frank [10/Oct/2007:13:55:36 -0700] "GET /index.html HTTP/1.0 " 200 2326 "http://www.example.com/links.html" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322)"

Lợi ích lớn nhất của tập tin nhật ký là tính sẵn có tương đối đơn giản. Máy chủ web như Apache mặc định phải cho phép ghi nhật ký. Các ứng dụng thường thực hiện ghi nhật ký để đảm bảo truy xuất nguồn gốc của các hành động của chúng.

## 1.3. Phương pháp phát hiện tấn công qua web log sử dụng học máy

### *1.3.1. Tổng quan về học máy*

Học máy (Machine Learning) là một ngành khoa học nghiên cứu các thuật toán cho phép máy tính có thể học được các khái niệm (concept).

Học máy là một lĩnh vực của trí tuệ nhân tạo liên quan đến việc nghiên cứu và xây dựng các kĩ thuật cho phép các hệ thống "học" tự động từ dữ liệu để giải quyết những vấn đề cụ thể. Học máy là các kỹ thuật giúp cho máy tính có thể tự học hỏi dựa trên dữ liệu đưa vào mà không cần phải được lập trình cụ thể. Thường một chương trình máy tính cần các quy tắc, luật lệ để có thể thực thi được một tác vụ nào đó như dán nhãn cho các email là thư rác nếu nội dung email có chứ từ khoá “quảng cáo”. Nhưng với học máy, các máy tính có thể tự động phân lại các thư rác thành mà không cần chỉ trước bất kỳ quy tắc nào cả.

Một bài toán học máy cần trải qua 3 bước chính:

- Chọn mô hình: Chọn một mô hình thống kê cho tập dữ liệu.

- Tìm tham số: Các mô hình thống kê có các tham số tương ứng, nhiệm vụ lúc này là tìm các tham số này sao cho phù hợp với tập dữ liệu nhất có thể.

- Suy luận: Sau khi có được mô hình và tham số, ta có thể dựa vào chúng để đưa ra suy luận cho một đầu vào mới nào đó.

Một bài toán học máy cần có dữ liệu để huấn luyện, ta có thể coi nó là điều kiện tiên quyết. Dữ liệu sau khi có được cần phải:

- Chuẩn hoá: Tất cả các dữ liệu đầu vào đều cần được chuẩn hoá để máy tính có thể xử lý được. Quá trình chuẩn hoá bao gồm số hoá dữ liệu, co giãn thông số cho phù hợp với bài toán. Việc chuẩn hoá này ảnh hưởng trực tiếp tới tốc độ huấn luyện cũng như cả hiệu quả huấn luyện.

- Phân chia: Việc mô hình được chọn rất khớp với tập dữ liệu đang có không có nghĩa là giả thuyết của ta là đúng mà có thể xảy ra tình huống dữ liệu thật lại không khớp. Vấn đề này trong học máy được gọi là khớp quá (Overfitting). Vì vậy khi huấn luyện người ta phải phân chia dữ liệu ra thành 3 loại để có thể kiểm chứng được phần nào mức độ tổng quát của mô hình. Cụ thể 3 loại đó là:

+Tập huấn luyện (Training set): Dùng để học khi huấn luyện.

+Tập kiểm chứng (Cross validation set): Dùng để kiểm chứng mô hình khi huấn luyện.

+Tập kiểm tra (Test set): Dùng để kiểm tra xem mô hình đã phù hợp chưa sau khi huấn luyện.

### *1.3.2. Các nhóm giải thuật học máy:*

Theo phương thức học, các thuật toán Machine Learning thường được chia làm 4 nhóm:

- Học có giám sát (Supervised learning): Máy tính được xem một số mẫu gồm đầu vào (input) và đầu ra (output) tương ứng trước. Sau khi học xong các mẫu này,

máy tính quan sát một đầu vào mới và cho ra kết quả. Supervised learning là nhóm phổ biến nhất trong các thuật toán Machine Learning.

- Học không giám sát (unsupervised learning) [2]: Thuật toán unsupervised learning sẽ dựa vào cấu trúc của dữ liệu để thực hiện một công việc nào đó, ví dụ như phân nhóm (clustering) hoặc giảm số chiều của dữ liệu (dimension reduction) để thuận tiện trong việc lưu trữ và tính toán. Một cách toán học, Thuật toán học không giám sát là khi chúng ta chỉ có dữ liệu vào X mà không biết nhãn Y tương ứng. Không giống như thuật toán học có giám sát, với học không giám sát, chúng ta không biết câu trả lời chính xác cho mỗi dữ liệu đầu vào.

- Học nửa giám sát: Một dạng lai giữa hai nhóm giải thuật trên. Các bài toán khi chúng ta có một lượng lớn dữ liệu X nhưng chỉ một phần trong chúng được gán nhãn được gọi là Semi-Supervised Learning.

- Học tăng cường (Reinforcement learning): Reinforcement learning là các bài toán giúp cho một hệ thống tự động xác định hành vi dựa trên hoàn cảnh để đạt được lợi ích cao nhất (maximizing the performance). Hiện tại, Reinforcement learning chủ yếu được áp dụng vào Lý thuyết trò chơi (Game Theory), các thuật toán cần xác định nước đi tiếp theo để đạt được điểm số cao nhất .

Việc giám sát thu thập, phân tích các log truy cập hệ thống nói chung và các log truy cập các dịch vụ mạng nói riêng là nhiệm vụ không thể thiếu trong các hệ thống giám sát, phân tích hành vi người dùng, phát hiện bất thường, phát hiện tấn công, xâm nhập hệ thống và mạng. Dữ liệu log có thể cung cấp cho người quản trị nhiều thông tin quan trọng về các hành vi người dùng trực tuyến, cũng như các dấu hiệu của các hành vi truy cập bất thường, các dạng tấn công, xâm nhập để đưa ra các cảnh báo nguy cơ mất an toàn thông tin đối với hệ thống.

Hiện có nhiều phương pháp phát hiện tấn công từ việc thu thập, xử lý, phân tích log truy cập. Trong nội dung của luận văn này, tác giả đi sâu nghiên cứu ứng dụng phương pháp học máy có giám sát, sử dụng bộ phân lớp rừng ngẫu nhiên để phát hiện các tấn công. Để thuận tiện cho quá trình tiền xử lý dữ liệu, trong phạm vi

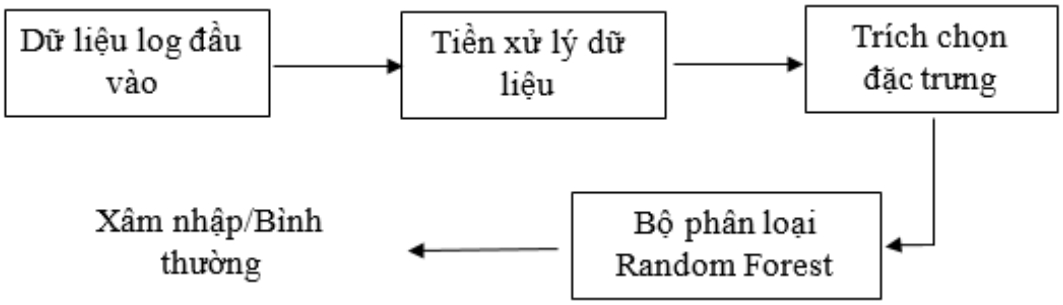
luận văn này sẽ sử dụng đầu vào dữ liệu là các web log máy chủ web được thu thập từ luồng mạng pcap. Trong chương tiếp theo, tác giả sẽ đi sâu nghiên cứu xây dựng mô hình phát hiện tấn công và đánh giá tính hiệu quả của mô hình.

# CHƯƠNG 2: PHƯƠNG PHÁP PHÁT HIỆN TẤN CÔNG

Trong Chương này, tác giả đi sâu nghiên cứu phương pháp phát hiện tấn công dựa vào phương pháp học máy có giám sát, và mô hình cụ thể được sử dụng trong phát hiện tấn công là Rừng ngẫu nhiên (Random Forest).

## 2.1. Phương pháp phát hiện tấn công

### *2.1.1. Mô hình phát hiện tấn công*




**Hình 2. 1- Mô hình hệ thống phát hiện xâm nhập**

Các thành phần trong mô hình Phát hiện mã độc tấn công có chủ đích gồm 4 thành phần chính:

- ***Khối Dữ liệu đầu vào:*** Do cấu trúc dữ liệu weblog rất đa dạng ở các hệ thống khác nhau, trong phạm vi luận văn này chỉ tập trung thu thập dữ liệu weblog thu thập từ luồng pcap tại các máy chủ Apache, Nginx, IIS và cho vào khối tiền xử lý dữ liệu.
- ***Khối tiền xử lý dữ liệu***: Tiền xử lý dữ liệu là bước rất quan trọng trong việc giải quyết bất kỳ vấn đề nào trong lĩnh vực Học Máy. Hầu hết các bộ dữ liệu được sử dụng trong Học Máy đều cần được xử lý, làm sạch và biến đổi trước khi một thuật toán Học Máy có thể được huấn luyện trên những bộ dữ liệu này. Các kỹ thuật tiền xử lý dữ liệu phổ biến hiện nay bao gồm xử lý dữ liệu bị khuyết (missing data), mã hóa các biến nhóm (encoding categorical variables), chuẩn hóa dữ liệu (standardizing data), co giãn dữ liệu (scaling data),…

Trong mô hình hệ thống phát hiện xâm nhập trên, chức năng của khối tiền xử lý dữ liệu là trích xuất lấy các thông tin từ log, các truy cập đến hệ thống máy chủ web. Tất cả các tập tin log sẽ được chuyển về một hệ thống chung để phân tích, chuyển đổi cấu trúc, phân tách các trường đặc trưng.

- ***Khối trích chọn đặc trưng***: Chọn ra những đặt trưng tốt nhất (good feature) của dữ liệu, lược bỏ những đặc trưng không tốt của dữ liệu, gây nhiễu (noise), quyết định chọn bao nhiêu đặc trưng để phân loại được dữ liệu.

Một ví dụ đơn giản của quá trình trích chọn đặc trưng: Nếu muốn xác định xem một người có hạnh phúc hay không, một đặc trưng tiềm năng là xem người đó có cười hay không….

Với bộ dữ liệu HTTP CSIC 2010, để xử lý bộ dữ liệu phù hợp cho mô hình thuật toán Random Forest, quá trình trích chọn đặc trưng sẽ trích chọn các đặc trưng liên quan sau để phát hiện các cuộc tấn công Web: Tên thuộc tính; Độ dài của yêu cầu; Độ dài của các đối số; Số lượng đối số; Chiều dài của đường dẫn; Số ký tự đặc biệt trên đường dẫn; Giá trị byte tối đa trong yêu cầu

- ***Khối bộ phân loại Random Forest:*** Chức năng của khối phân lớp Random Forest được mô tả chi tiết trong mục 2.2.

Có nhiều bộ phân lớp có thể áp dụng để xây dựng mô hình phát hiện tấn công này, như SVM, Decision Tree, Navie Bayers, Random forests…, Random forests được coi là một phương pháp chính xác và mạnh mẽ, có thể làm việc được với dữ liệu thiếu giá trị, và khi Forest có nhiều cây hơn, chúng ta có thể tránh được việc Overfitting với tập dữ liệu vì vậy trong nội dung luận văn, tác giả lựa chọn bộ phân lớp Random Forest để xây dựng mô hình giải quyết bài toán phát hiện tấn công.

- ***Thông báo kết quả***: Sẽ thông báo cho người dùng kết quả phát hiện Xâm nhập/Bình thường.

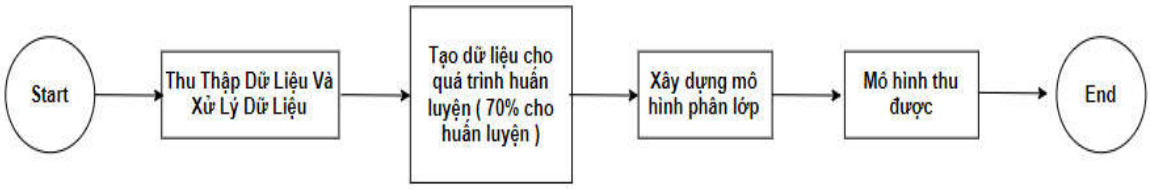
### *2.1.2. Các giai đoạn thực hiện*

Thông thường một ứng dụng của một mô hình học máy thường được chia làm hai giai đoạn đó là : Huấn luyện và kiểm tra mô hình . Tỷ lệ độ chính xác phát hiện ra mã độc của mô hình phụ thuộc rất nhiều vào chất lượng của bộ tập mẫu.

Để giải quyết bài toán phát hiện tấn công dựa trên log file, luồng mạng thì cần thực hiện các giai đoạn sau đây :

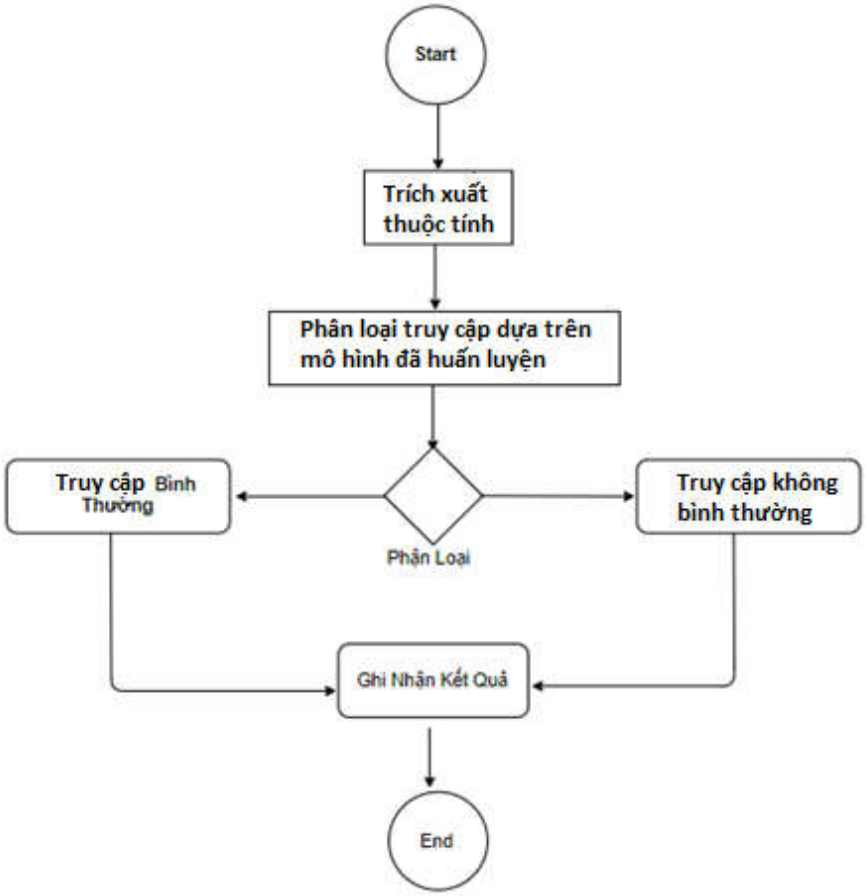
- Giai đoạn 1: Thu thập dữ liệu và tiền xử lý dữ liệu phục vụ cho quá trình học của mô hình.

- Giai đoạn 2: Lựa chọn thuật toán xây dựng mô hình.
- Giai đoạn 3: Huấn luyện mô hình với dữ liệu đã xử lý




Hình 2. 2 - Sơ đồ tạo mô hình phân lớp

- Giai đoạn 4: Kiểm tra huấn luyện trên mô hình mới




Hình 2. 3 - Mô hình phân lớp dữ liệu đầu vào

## 2.2. Tổng quan về thuật toán Random Forest

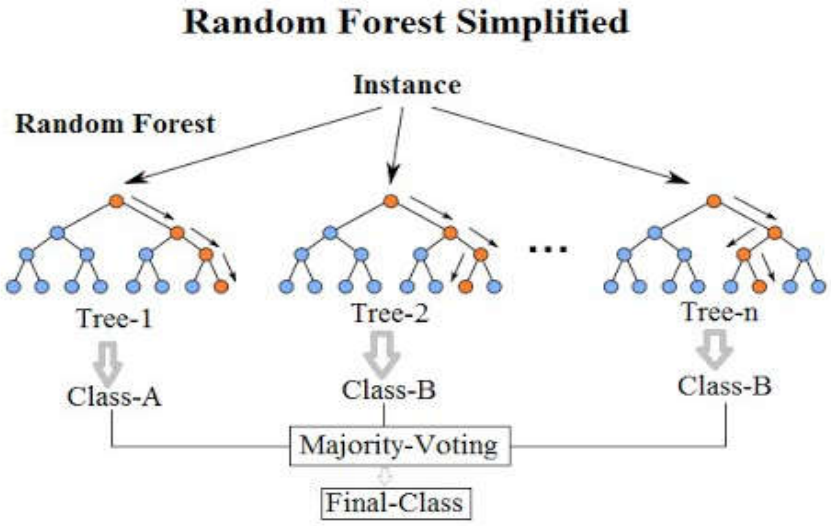
Thuật toán Random Forest [3] lần đầu tiên được đề xuất vào năm 1996 và được giới thiệu chính thức bởi Leo Breiman vào năm 2001, thuật toán đã tạo ra 1 cuộc cách mạng trong Machine Learning, là phương pháp học máy kết hợp, tức là sử dụng cách

kết hợp các phương pháp học máy đơn giản để xây dựng một mô hình có độ chính xác cao hơn. Random Forest có thể được sử dụng để giải cả bài toán phân loại và hồi quy. Nó làm việc bằng cách xây dựng một tập hợp các cây quyết định trong quá trình training, sau đó kết hợp kết quả trả về của mỗi cây đưa ra quyết định dự đoán cuối cùng. Rừng ngẫu nhiên là một thuật toán học có giám sát. Rừng ngẫu nhiên sử dụng các cây (tree) để làm nền tảng. Random Forest là thành viên trong chuỗi thuật toán cây quyết định.

Random Forest (RF) dựa trên cơ sở :

- Random = Tính ngẫu nhiên ;
- Forest = nhiều cây quyết định (decision tree).

Đơn vị của RF là thuật toán cây quyết định, với số lượng hàng trăm. Mỗi cây quyết định được tạo ra một cách ngẫu nhiên từ việc : Tái chọn mẫu (bootstrap, random sampling) và chỉ dùng một phần nhỏ tập biến ngẫu nhiên (random features) từ toàn bộ các biến trong dữ liệu. Ở trạng thái sau cùng, mô hình RF thường hoạt động rất chính xác, nhưng đổi lại, ta không thể nào hiểu được cơ chế hoạt động bên trong mô hình vì cấu trúc quá phức tạp.




Hình 2. 4- Mô hình Random Forest

(Nguồn https://medium.com)

Nhiều cây quyết định được tạo theo các ngẫu nhiên sẽ tạo ra một rừng ngẫu nhiên (random forest). Một rừng ngẫu nhiên phân lớp bao gồm một tổ hợp cây quyết định phân lớp:

$$\{\delta(\varepsilon,\omega_k,k=1,\ldots)\}$$

Với điều kiện $\{\omega_k\}$ là các cây quyết định được tạo độc lập ngẫu nhiên và mỗi cây quyết định sẽ bình chọn cho kết quả lớp phổ biến nhất với giá trị đầu vào x.

### *2.2.1. Cách làm việc của thuật toán*

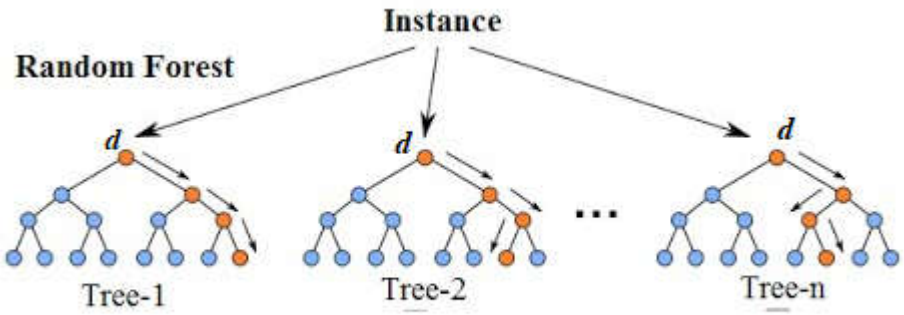
Thuật toán Random Forest bao gồm 2 giai đoạn chính:

- Quá trình tạo ra rừng cây ngẫu nhiên
- Quá trình thực hiện dự đoán dựa trên rừng ngẫu nhiên đã tạo

a. Quá trình tạo ra rừng ngẫu nhiên

Một rừng ngẫu nhiên là một tập hợp của rất nhiều cây quyết định (decision tree) Để tạo mới cây quyết định, thuật toán Random Forest luôn luôn bắt đầu với 1 cây quyết định rỗng. Đó là cây quyết định chỉ có điểm bắt đầu và liên kết thẳng tới câu trả lời. Thuật toán sẽ tìm ra câu hỏi đầu tiên tốt nhất để bắt đầu, và sau đó xây dựng cây quyết định. Mỗi khi thuật toán tìm được 1 câu hỏi tốt để hỏi, nó sẽ tạo ra 2 nhánh (trái và phải) của cây. Khi không còn câu hỏi nào thú vị nữa, thuật toán sẽ dừng lại và kết thúc quá trình xây dựng cây quyết định[5].

Để chắc chắn rằng tất cả các cây quyết định là không giống nhau, Random Forest sẽ tự động thay đổi ngẫu nhiên đối tượng cần theo dõi. Nói một cách chính xác hơn, thuật toán sẽ xóa ngẫu nhiên 1 vài đối tượng, và nhân bản 1 vài đối tượng khác. Tiến trình này được gọi là “bootstrapping”. Ngoài ra để đảm bảo rằng cây quyết định có sự khác biệt, Random Forest sẽ ngẫu nhiên loại bỏ có mục đích một vài câu hỏi khi xây dựng cây quyết định. Trong trường hợp này, nếu câu hỏi tốt nhất không được kiểm tra, thì các câu hỏi khác sẽ được chọn để tạo ra cây. Quá trình được gọi là “attribute sampling”.

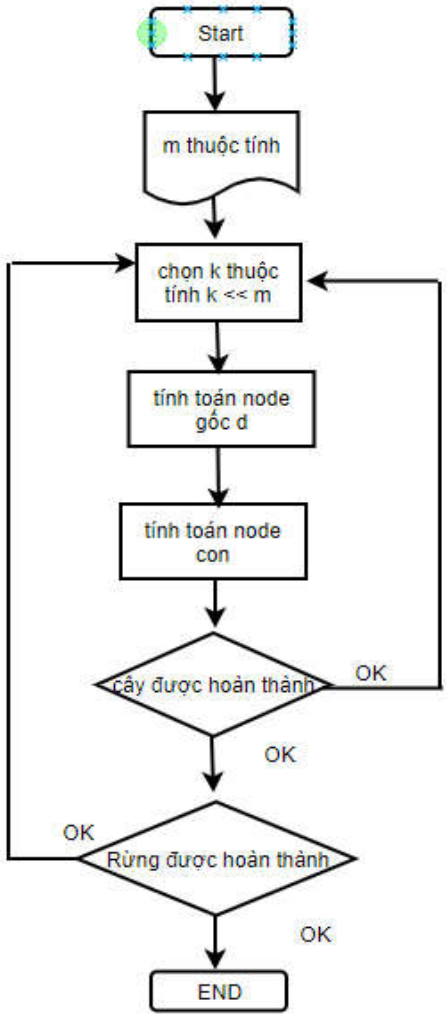



Hình 2. 5- Tạo rừng ngẫu nhiêu trong Random forest

(Nguồn https://medium.com)

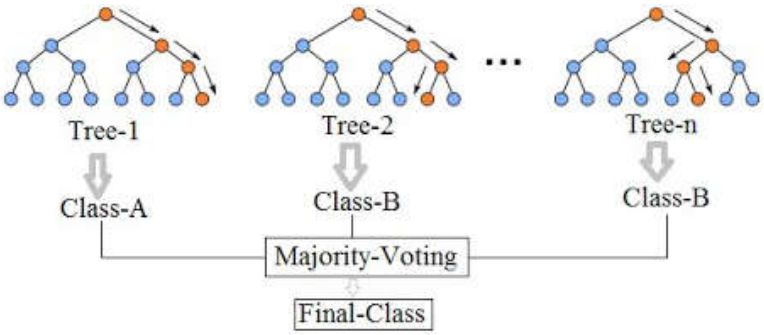
Quá trình tao ra rừng cây ngẫu nhiên được thể hiện qua các bước sau :

- Bước 1 : Chọn ngẫu nhiên k thuộc tính từ tổng m thuộc tính sao cho **k<<m**
- Bước 2 : Trong số ***k*** thuộc tính, tính toán node gốc (root) ***d*** sử dụng phương pháp chọn thuộc tính tốt nhất (best split point).
- Bước 3: Chọn các node trong (internal node) bằng cách sử dụng phương pháp chọn thuộc tính tốt nhất (best split).
- Bước 4: Lặp lại bước 1 đến bước 3 cho đến khi cây được hoàn thành.
- Bước 5: Lặp lại bước 1 dến bước 4 cho đến khi rừng cây được hoàn thành.




Hình 2. 6 - Sơ đồ tạo rừng ngẫu nhiên

b. Quá trình thực hiện dự đoán dựa trên rừng ngẫu nhiên đã tạo




Hình 2. 7 - Quá trình dự đoán trong Random forest

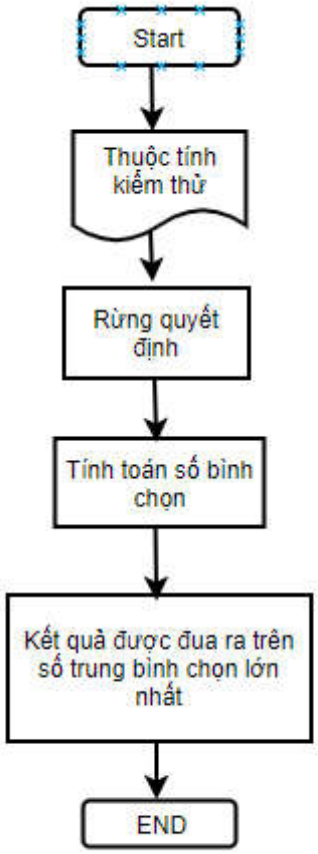
(Nguồn https://medium.com)

Sau quá trình tạo rừng ngẫu nhiên, thuật toán sẽ dự đoán trên rừng đã được tạo các bước cho quá trình dự đoán như sau :

❖ Bước 1: Lấy tập thuộc tính kiểm thử và sử dụng tập luật được tạo ra bởi cây quyết định ngẫu nhiên trong quá trình tạo rừng cây ngẫu nhiên, để dự đoán đầu ra.

❖ Bước 2: Tính toán số phiếu bầu – bình chọn của mỗi cây ngẫu nhiên đưa ra.

❖ Bước 3: Coi số phiếu bầu – bình chọn cao nhất trong cả rừng cây ngẫu nhiên là kết quả cuối cùng




Hình 2. 8 - Sơ đồ dự đoán trên rừng ngẫu nhiên

### *2.2.2. Thuật toán lựa chọn thuộc tính cho Random Forest*

Trong thuật toán Random Forest, để lựa chọn ra thuộc tính nào phù hợp nhất để làm node gốc (root node) và các thuộc tính nào phù hợp để làm các node trong

(internal node) tiếp theo, thì thuật toán Random Forest sử dụng chủ yếu thuật toán Information Gain.

Thuật toán Information Gain là một thuật toán được thực hiện dựa trên việc dùng Entropy làm độ đo.

Công thức tính Entropy như sau:

$$H(X) = E_X[I(x)] = -\sum_{x \in X} p(x) log p(x)$$

*Hàm số Entropy*

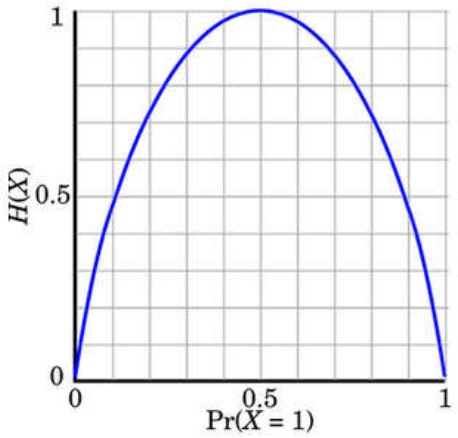
Cho một phân phối xác suất của một biến rời rạc x có thể nhận được n giá trị khác nhau x1,x2,..,xn. Giả sử rằng xác xuất để x nhận các giá trị này là $p_i$ = p(x= xi)

Ký hiệu phân phối này là **p** = ($p_1, p_2, \ldots, p_n$).

Entropy của phân phối này là :

$$H(\mathbf{p}) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Hàm Entropy được biểu diễn dưới dạng đồ thị như sau :




Hình 2. 9 - Đồ thị kết quả entropy

Từ đồ thị ta thấy, hàm Entropy sẽ đạt giá trị nhỏ nhất nếu có một giá trị *pi* =1, đạt giá trị lớn nhất nếu tất cả các *pi* bằng nhau. Hàm Entropy càng lớn thì độ ngẫu nhiên của các biến rời rạc càng cao (càng không tinh khiết).

Với cây quyết định, ta cần tạo cây như thế nào để cho ta nhiều thông tin nhất, tức là Entropy là cao nhất.

Bài toán của ta trở thành, tại mỗi tầng của cây, cần chọn thuộc tính nào để độ giảm Entropy là thấp nhất.

Người ta có khái niệm Information Gain được tính bằng :

$$Gain(S,f) = H(S) - H(f,S)$$

trong đó:

$H(S)$ là Entropy tổng của toàn bộ tập dataset $S$.

$H(f,S)$ là Entropy được tính trên thuộc tính $f$.

Do $H(S)$ là không đổi với mỗi tầng, ta chọn thuộc tính $f$ có Entropy nhỏ nhất để thu được $Gain(S,f)$ lớn nhất.

## 2.3. Tập dữ liệu huấn luyện (CSIC 2010)

Bộ dữ liệu trong luận văn sử dụng được lấy từ tập dữ liệu Bộ dữ liệu HTTP CSIC 2010[1], được phát triển tại Viện An toàn Thông tin thuộc Hội đồng Nghiên cứu Quốc gia Tây Ban Nha, chuyên để thử nghiệm các giải pháp tường lửa ứng dụng web. Trong tập dữ liệu này có tổng cộng 36.000 câu truy vấn an toàn và 25.000 câu truy vấn có tấn công. Tập dữ liệu chứa hầu hết các loại tấn công phổ biến của ứng dụng web, bao gồm các cuộc tấn công như SQL, tràn bộ đệm, thu thập thông tin, tiết lộ tệp, tiêm CRLF, XSS, bao gồm phía máy chủ, giả mạo tham số, v.v.

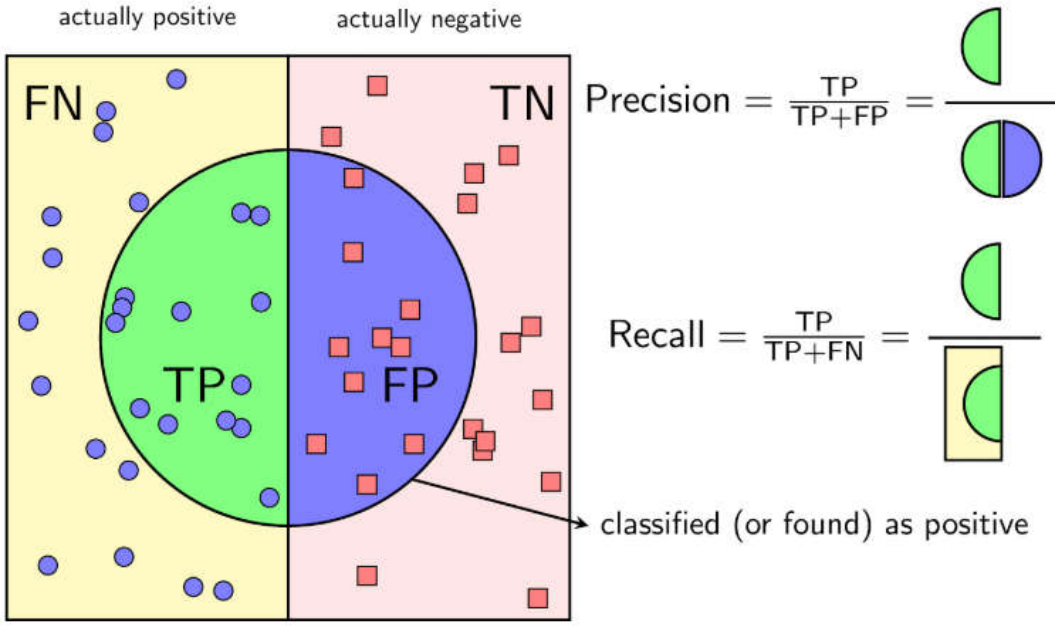
Để xử lý bộ dữ liệu phù hợp cho mô hình thuật toán Random Forest, quá trình trích chọn đặc trưng sẽ trích chọn các trường sau: Host, Method, Content Type (lưu vị trí của các giá trị đó trong mảng), URL, Payload và Content Length (lưu độ dài chuỗi),

## 2.4. Phương pháp đánh giá

Hiệu năng của một mô hình thường được đánh giá dựa trên tập dữ liệu kiểm thử (test data), được phân tách từ tập dữ liệu. Ở bước này thực hiện chia ngẫu nhiên dữ liệu thành 2 phần theo tỉ lệ: 70% dùng cho training và 30% cho testing. Dữ liệu này được chia làm 2 nhãn, với các truy cập bình thường được đánh dấu nhãn 0, còn các truy cập độc hại được đánh dấu nhãn 1.

Có rất nhiều cách đánh giá một mô hình phân lớp. Các phương pháp thường được sử dụng là: accuracy score, confusion matrix, ROC curve… Precision and

Recall, F1 score… Trong luận văn sẽ sử dụng phương pháp Precision and Recall, F1 score do tập dữ liệu của các lớp là chênh lệch nhau nhiều.




Hình 2. 10- Cách tính Precision và Recall

(Nguồn: machinelearningcoban.com)

- TP (true positive) – mẫu mang nhãn dương được phân lớp đúng vào lớp dương.
- FN (false negative) – mẫu mang nhãn dương bị phân lớp sai vào lớp âm.
- FP (false positive) – mẫu mang nhãn âm bị phân lớp sai vào lớp dương.
- TN (true negative) – mẫu mang nhãn âm được phân lớp đúng vào lớp âm.

Từ bốn thông số trên, ta có thể tính được các giá trị:

- Precision (độ chính xác): Trong tất cả các dự đoán thuộc lớp được đưa ra, bao nhiêu dự đoán là chính xác. Được tính bằng tổng số các ví dụ thuộc lớp dương được phân loại chính xác chia cho tổng số các ví dụ được phân loại vào lớp dương. Precision cao đồng nghĩa với việc độ chính xác của các điểm tìm được là cao

$$\text{Precision} = \frac{tp}{tp+fp}$$

- Recall (độ hồi tưởng): Trong tất cả các trường hợp thuộc lớp dương, bao nhiêu trường hợp đã được dự đoán chính xác. Được tính bằng tổng số các ví dụ thuộc lớp

dương được phân loại chính xác chia cho tổng số các ví dụ thuộc lớp dương. Recall cao đồng nghĩa với việc tỷ lệ True Positive cao, tức tỉ lệ bỏ sót các điểm thực sự positive là thấp

$$\text{Recall} = \frac{tp}{tp+fn}$$

- F1-Score: Tiêu chí đánh giá F1 là sự kết hợp của 2 tiêu chí đánh giá Precision và Recall. F1 là một trung bình điều hòa (harmonic mean) của các tiêu chí Precision và Recall. F1 có xu hướng lấy giá trị gần với giá trị nào nhỏ hơn giữa 2 giá trị Precision và Recall. F−1-score có giá trị nằm trong nửa khoảng $(0,1)$, F1 có giá trị lớn nếu cả 2 giá trị Precision và Recall đều lớn.

$$F = 2.\frac{Precision.recall}{Precision+Recall}$$

## 2.5. Kết quả thử nghiệm

Tổng hợp dữ liệu mã độc tấn công có chủ đích từ tập dữ liệu HTTP CSIC 2010. Từ tập dữ liệu này, sau khi tiền xử lý dữ liệu phù hợp với mô hình học máy, trộn và chia ngẫu nhiên dữ liệu thành 2 phần cho training và testing, tách nhãn phân loại cho quá trình training và lưu ra file, cho vào thuật toán học máy tạo model.

Kết quả thực hiện

| | Precision | Recall | F1-score |
|---|---|---|---|
| Bình thường | 0,82 | 0.91 | 0.86 |
| Tấn công | 0.84 | 0.72 | 0.77 |
| Avg/total | 0.83 | 0.83 | 0.83 |

Hình 2. 11 -Kết quả học máy và in ma trận nhầm lẫn

Từ các thông số trên, ta rút ra một số kết luận cho mô hình này:

- Precision: Mô hình dự đoán đúng 83% requests bình thường trong tổng số các requests mà nó phân loại là bình thường.

- Recall : Mô hình dự đoán đúng 83% request tấn công trong tổng số các request mà nó phân loại là tấn công.

## 2.6. Kết luận chương

Trong chương này đã xây dựng được mô hình học máy phát hiện tấn công và thực hiện đánh giá hiệu quả của mô hình dựa trên các tập dữ liệu HTTP CSIC 2010. Qua kịch bản thử nghiệm cho thấy mô hình với dữ liệu từ tập dữ liệu HTTP CSIC 2010 đạt kết quả tốt. Vì vậy, hệ thống sẽ được xây dựng dựa trên mô hình và tập dữ liệu này. Chương kế tiếp sẽ trình bày mô hình thực nghiệm ứng dụng.

# CHƯƠNG III – XÂY DỰNG HỆ THỐNG THỰC NGHIỆM

## 3.1. Xây dựng hệ thống

Đây là một hệ thống phát hiện tấn công dựa trên việc đọc gói tin trực tiếp hoặc đọc file log sử dụng bộ dữ liệu học HTTP CSIC 2010.

Chương trình có các chức năng chính:

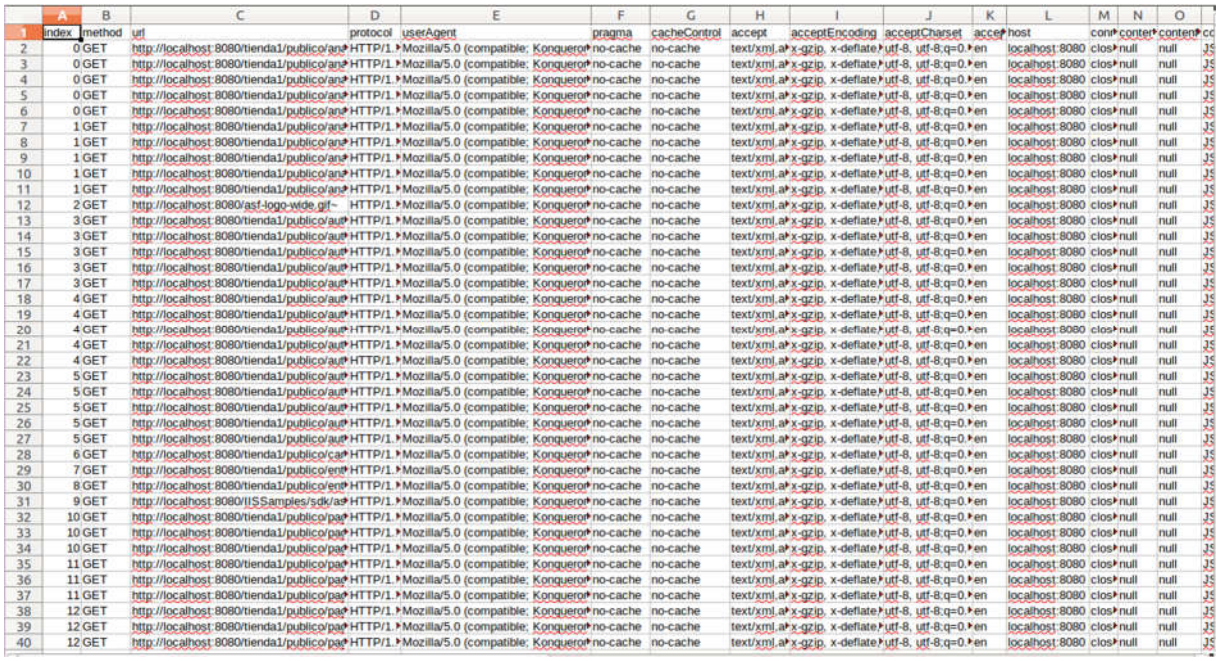
- Đọc trực tiếp request từ pcap: Ở chế độ này, ứng dụng sẽ lọc các http request được gửi tới server, đem phân loại từng request, in kết quả lên màn hình đồng thời lưu ra file log riêng. Chế độ này có ưu điểm là đọc và phân loại tất cả gói tin gửi đến server mà không phân biệt web server nào. Tuy nhiên đối với mô hình mạng lớn, việc đọc tất cả các gói tin khiến cho server xử lý chậm chạp có thể gây crash ứng dụng
- Đọc file pcap ngoại tuyến: Khi các gói tin gửi đến được tổng hợp và lưu dưới dạng file pcap, có thể dùng ứng dụng mở lên đọc các kết nối HTTP và phân loại.
- Đọc trực tiếp file log từ webserver: Mỗi khi có kết nối đến web server được ghi vào file log, ứng dụng sẽ đọc các dòng mới này và đem đi phân loại request. Chế độ này có thể phân loại request thuộc về web server nào, ngoài ra giảm tải cho máy server vì chỉ đọc và phân tích những http request đã được lọc sẵn bởi web server.
- Ngoài ra, ta có thể đọc lại các file log từ Apache, Nginx đã ghi sẵn cho quá trình phân loại và in kết quả.

Chương trình sẽ lấy các dữ liệu tại URL và Payload làm tiền đề để phân loại, vì vậy trước mắt chương trình chỉ có thể phân biệt các tấn công có tác động đến URL và Payload ví dụ như Command Injection, SQL Injection, XSS, Weak Session ID,…

### *3.1.1. Thu thập dữ liệu log và tiền xử lý dữ liệu*

Lấy các thông tin liên quan đến tình trạng hoạt động của máy chủ web, thiết bị mạng. Tất cả các tập tin log sẽ được chuyển về một hệ thống chung để phân tích, chuyển đổi cấu trúc định dạng log, phân tách các trường riêng biệt và phân tích thông tin thu thập được.

Quá trình tiền xử lý dữ liệu vào và lọc dữ liệu, ta cần có các trường Host, Method, Content Type (lưu vị trí của các giá trị đó trong mảng), URL, Payload và Content Length (lưu độ dài chuỗi).



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | index | method | url | protocol | userAgent | pragma | cacheControl | accept | acceptEncoding | acceptCharset | acce | host | con | conter | content | cc |
| 2 | 0 | GET | http://localhost:8080/tienda1/publico/an | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 3 | 0 | GET | http://localhost:8080/tienda1/publico/an | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 4 | 0 | GET | http://localhost:8080/tienda1/publico/an | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 5 | 0 | GET | http://localhost:8080/tienda1/publico/an | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 6 | 0 | GET | http://localhost:8080/tienda1/publico/an | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 7 | 1 | GET | http://localhost:8080/tienda1/publico/an | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 8 | 1 | GET | http://localhost:8080/tienda1/publico/an | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 9 | 1 | GET | http://localhost:8080/tienda1/publico/an | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 10 | 1 | GET | http://localhost:8080/tienda1/publico/an | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 11 | 1 | GET | http://localhost:8080/tienda1/publico/an | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 12 | 2 | GET | http://localhost:8080/asf-logo-wide.gif~ | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 13 | 3 | GET | http://localhost:8080/tienda1/publico/au | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 14 | 3 | GET | http://localhost:8080/tienda1/publico/au | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 15 | 3 | GET | http://localhost:8080/tienda1/publico/au | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 16 | 3 | GET | http://localhost:8080/tienda1/publico/au | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 17 | 3 | GET | http://localhost:8080/tienda1/publico/au | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 18 | 4 | GET | http://localhost:8080/tienda1/publico/au | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 19 | 4 | GET | http://localhost:8080/tienda1/publico/au | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 20 | 4 | GET | http://localhost:8080/tienda1/publico/au | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 21 | 4 | GET | http://localhost:8080/tienda1/publico/au | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 22 | 4 | GET | http://localhost:8080/tienda1/publico/au | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 23 | 5 | GET | http://localhost:8080/tienda1/publico/au | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 24 | 5 | GET | http://localhost:8080/tienda1/publico/au | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 25 | 5 | GET | http://localhost:8080/tienda1/publico/au | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 26 | 5 | GET | http://localhost:8080/tienda1/publico/au | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 27 | 5 | GET | http://localhost:8080/tienda1/publico/au | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 28 | 6 | GET | http://localhost:8080/tienda1/publico/ca | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 29 | 7 | GET | http://localhost:8080/tienda1/publico/en | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 30 | 8 | GET | http://localhost:8080/tienda1/publico/en | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 31 | 9 | GET | http://localhost:8080/IISSamples/sdk/as | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 32 | 10 | GET | http://localhost:8080/tienda1/publico/pa | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 33 | 10 | GET | http://localhost:8080/tienda1/publico/pa | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 34 | 10 | GET | http://localhost:8080/tienda1/publico/pa | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 35 | 11 | GET | http://localhost:8080/tienda1/publico/pa | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 36 | 11 | GET | http://localhost:8080/tienda1/publico/pa | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 37 | 11 | GET | http://localhost:8080/tienda1/publico/pa | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 38 | 12 | GET | http://localhost:8080/tienda1/publico/pa | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 39 | 12 | GET | http://localhost:8080/tienda1/publico/pa | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |
| 40 | 12 | GET | http://localhost:8080/tienda1/publico/pa | HTTP/1. | Mozilla/5.0 (compatible; Konqueror | no-cache | no-cache | text/xml,a | x-gzip, x-deflate | utf-8, utf-8;q=0. | en | localhost:8080 | clos | null | null | JS |

Hình 3. 1 - CSIC Dataset ở định dạng .csv

Với bộ CSIC Dataset ban đầu, ta gộp hai file normal và anomalous, đổi sang định dạng .csv với tất cả các trường. Tiến hành xóa các trường không dùng đến, ta sẽ giữ lại các trường: Method, Host, Index, URL, Payload, ContentLength, Label. Đổi các trường ít giá trị: Method, Host, Label sang mảng và gán số thứ tự.

Đoạn lệnh python sau thực hiện lưu các trường hay gặp dưới dạng thứ tự trong mảng:

```
#Tach 2 bang data va payload
payload = data.loc[:,("index", "payload","label")]
data = data.loc[:,
("index","method","url","host","contentLength","cookie","label")]
data = data.drop_duplicates()
#Thay cac gia tri thuong gap thanh so
data = data.replace(["GET", "POST", "PUT"], [1, 2, 3])
data = data.replace(['localhost:8080', 'localhost:9090'], [0,1])
data = data.replace(["norm", "anom"], [0,1])
#Thay gia tri null
data.loc[:,"contentLength"] =
data["contentLength"].fillna(value=0)
payload.loc[:,"payload"] = payload["payload"].fillna(value="")
```

Tiếp đến xử lí URL và Payload, tại mỗi trường ta cần lưu thêm những thông tin sau: Tổng số ký tự; Số ký tự số; Số ký tự đặc biệt; Số ký tự không phải chữ. Với Payload ta cần thêm trường số lượng đối số (agrument) nhập vào.

Đoạn code python sau thực hiện xử lý URL:

```
url = data["url"]
data.insert(3,"digit_path", url.str.count(r'[0-9]')) #So chu so
data.insert(3,"special_path", url.str.count(r'[^a-zA-Z\d\s\/:\.]')) #Ky tu
dac biet
data.insert(4,"non_an_path", url.str.count(r'[^a-zA-Z\d\s]')) #So ky tu
khong phai chu
data.loc[:,'url'] = data['url'].str.len() #Doi chuoi url thanh so ky tu
url
data.loc[:,'cookie'] = data['cookie'].str.len()
data = data.rename(columns={"url":"url_length", "cookie":"cookie_length"})
```

Tiếp theo là cách xử lý các trường của Payload:

```
#Xu ly payload
arg_anom = payload[payload['label'] == "anom"].loc[:, ("index",
"payload")]
arg_norm = payload[payload['label'] == "norm"].loc[:, ("index",
"payload")]
grouped_norm = arg_norm.groupby("index")
grouped_anom = arg_anom.groupby("index")
#Do dai payload
anom.insert(5, "arg_length", grouped_anom.sum()["payload"].str.len())
norm.insert(5, "arg_length", grouped_norm.sum()["payload"].str.len())
#So luong dau vao
anom.insert(6, "arg_num" , grouped_anom.size())
norm.insert(6, "arg_num" , grouped_norm.size())
arg_sum_anom = grouped_anom.sum()
arg_sum_norm = grouped_norm.sum()
#arg_sum_anom.to_csv("data/arg_sum_anom.csv", index=False)
#arg_sum_norm.to_csv("data/arg_sum_norm.csv", index=False)
anom.insert(7,"digit_in_arg", arg_sum_anom["payload"].str.count(r'[0-9]'))
#So chu so tai args
norm.insert(7,"digit_in_arg", arg_sum_norm["payload"].str.count(r'[0-9]'))
anom.insert(8,"letter_in_arg", arg_sum_anom["payload"].str.count(r'[a-zA-
Z]'))#So chu cai tai args
norm.insert(8,"letter_in_arg", arg_sum_norm["payload"].str.count(r'[a-zA-
Z]'))
```

Sau khi có được bảng dữ liệu mới, tiến hành trộn dữ liệu lên, chia làm hai phần test và train, tách nhãn phân loại cho quá trình train và lưu ra file. Chuẩn bị cho bước học máy tạo model.

Đoạn code python thực hiện lưu kết quả của Features Extract:

```
total = anom.append(norm) # combine anom and norm
total = total.reset_index(drop=True) # Reset Index
#total.to_csv("data/total.csv", index=False)
total = total.sample(frac=1).reset_index(drop=True) #Tron du lieu
#Chia du lieu de train
train = total.iloc[:51065]
test = total.iloc[51065:].reset_index(drop=True)
x_train = train.loc[:, train.columns.values[:-1]]
y_train = train.loc[:, 'label']
x_test = test.loc[:, test.columns.values[:-1]]
y_test = test.loc[:,'label']
x_train.to_csv("../../SVM/data/x_train.csv", index=False)
y_train.to_csv("../../SVM/data/y_train.csv", index=False)
x_test.to_csv("../../SVM/data/x_test.csv", index=False)
y_test.to_csv("../../SVM/data/y_test.csv", index=False)
```

Bảng sau thể hiện kết quả sau quá trình extract Featuré

| 1 | method | url_length | special_path | non_an_path | digit_path | arg_length | arg_num | digit_in_arg | letter_in_arg | host | contentLength |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 47 | 0 | 8 | 5 | 62 | 3 | 7 | 46 | 0 | 64 |
| 3 | 2 | 48 | 0 | 8 | 5 | 63 | 5 | 7 | 47 | 0 | 67 |
| 4 | 2 | 52 | 0 | 8 | 5 | 57 | 5 | 6 | 46 | 0 | 61 |
| 5 | 1 | 48 | 0 | 8 | 5 | 0 | 1 | 0 | 0 | 0 | 0 |
| 6 | 1 | 48 | 0 | 8 | 5 | 0 | 1 | 0 | 0 | 0 | 0 |
| 7 | 1 | 56 | 0 | 8 | 5 | 0 | 1 | 0 | 0 | 0 | 0 |
| 8 | 2 | 57 | 0 | 8 | 5 | 4 | 1 | 1 | 2 | 0 | 4 |
| 9 | 1 | 44 | 0 | 8 | 6 | 0 | 1 | 0 | 0 | 0 | 0 |
| 10 | 1 | 44 | 0 | 8 | 6 | 0 | 1 | 0 | 0 | 0 | 0 |
| 11 | 1 | 49 | 0 | 8 | 5 | 222 | 13 | 42 | 158 | 0 | 0 |
| 12 | 2 | 48 | 0 | 8 | 5 | 72 | 1 | 12 | 44 | 0 | 72 |
| 13 | 2 | 48 | 0 | 8 | 5 | 17 | 1 | 1 | 14 | 0 | 17 |
| 14 | 1 | 47 | 0 | 8 | 5 | 0 | 1 | 0 | 0 | 0 | 0 |
| 15 | 2 | 48 | 0 | 8 | 5 | 18 | 1 | 1 | 15 | 0 | 18 |
| 16 | 1 | 49 | 0 | 8 | 5 | 222 | 13 | 38 | 163 | 0 | 0 |
| 17 | 1 | 52 | 0 | 8 | 5 | 63 | 5 | 1 | 56 | 0 | 0 |
| 18 | 2 | 48 | 0 | 8 | 5 | 14 | 1 | 3 | 9 | 0 | 14 |
| 19 | 1 | 50 | 0 | 8 | 5 | 258 | 13 | 44 | 187 | 0 | 0 |
| 20 | 1 | 48 | 0 | 8 | 5 | 18 | 1 | 1 | 15 | 0 | 0 |
| 21 | 2 | 48 | 0 | 8 | 5 | 33 | 1 | 0 | 31 | 0 | 33 |
| 22 | 1 | 44 | 0 | 8 | 6 | 0 | 1 | 0 | 0 | 0 | 0 |
| 23 | 1 | 58 | 0 | 9 | 5 | 0 | 1 | 0 | 0 | 0 | 0 |
| 24 | 1 | 49 | 0 | 8 | 5 | 0 | 1 | 0 | 0 | 0 | 0 |
| 25 | 2 | 49 | 0 | 8 | 5 | 231 | 13 | 38 | 171 | 0 | 243 |

Hình 3. 2 - Kết quả file training sau quá trình Extract Features

Với chức năng nghe trực tiếp gói tin qua pcap, ta sử dụng scapy sniff để bắt được các gói tin gửi đến server . Với module thu thập và phân loại gói tin tại file live_core.py. Đoạn code python sau thực hiện lọc các http requests trong pcap

```
def sniff_packets(packet_captured):
    """pass the request of it valid http header"""
    global GUI
    if not STOP_EV.is_set():
       if is_http(packet_captured) and IP in packet_captured:
          ip_src = packet_captured[IP].src
          for method in METHODS:
              if method in str(packet_captured):
                  OBTAINED_PAYLOAD['Method'] = str(METHODS.index(method))
```

```
                    classify_live_data(packet_captured.load,
    packet_captured.src, ip_src, GUI, LABEL)
```

Lọc các HTTP requests, sau đó tìm phương thức và gán số thứ tự của phương thức trong mảng vào trường method. Tiếp đó đưa chuỗi tin vào hàm phân loại.

Tại hàm phân loại, ta trích xuất các thông tin Features yêu cầu trong chuỗi. User agent ở giữa chuỗi "User-Agent" và chuỗi ngắt dòng "\r\n". URL ở giữa dấu cách đầu tiên và chuỗi "HTTP". Cách bố trí Payload của scapy sniff cũng tương tự như trong CSIC Dataset. Đối với phương thức GET, payload nằm trong URL sau dấu "?". Các phương thức còn lại payload nằm ở dòng cuối cùng của gói tin.

Đoạn code thực hiện Lọc features từ pcap:

```
def classify_live_data(load, mac_src, ip_src, add_line, lognum):
    """ham phan loai pcap"""
    global input_file
    input_file.flush()
    global STOP_EV
    global countsniff
    method = OBTAINED_PAYLOAD['Method']
    # finding User-Agent
    try:
        srt = load.index('User-Agent:')
    except ValueError:
        srt = 0
    if srt:
        finish = find_user_agent(load, "\r\n")
        # finding URL
        try:
            start = load.index(' ')
        except ValueError:
            start = 0
        if start:
            end = load.index('HTTP')
            if OBTAINED_PAYLOAD['Method'] == '0':
                try:
                    url = 'http://localhost:8080' + str(load[start+1:end])
                    url_length = len(url)
                    special_path = re.findall('[^a-zA-Z\d\s\/:\.]', url) #Len
                    non_an_path = re.findall('[^a-zA-Z\d\s]', url) #Len
                    digit_path = re.findall('[0-9]', url) #Len
                    host = 0
                    try:
                       arg = load[start + 1:end].split('?')[1]
                       arg_num = re.findall('&', load[start + 1:end])
                    except IndexError:
                       arg = ""
                       arg_num = []
                    arg_length = len(arg)
                    digit_in_arg = re.findall('[0-9]', arg) #Len
```

```
                letter_in_arg = re.findall('[a-zA-Z]', arg) #Len
                contentLength = arg_length + len(arg_num)
```

Sau khi có đủ các giá trị, ta truyền dữ liệu vào một mảng numpy, gọi hàm phân loại lấy kết quả, kết hợp ghi ra file log riêng, in ra GUI.

Đoạn code thực hiện phân loại và in kết quả:

```
prediction = classifier.predict(test)
result = list()
for w in prediction:
   result.append(w)
anomalous = True if 1 in result else False
if anomalous:
   classed = 'anomalous'
else:
   classed = 'normal'
print >> input_file, mac_src + "\n" + ip_src + "\n" +
str(load[srt:finish])
print >> input_file, urllib.unquote(str(load[start + 1:end])) + "\n" +
classed
if not STOP_EV.is_set():
    if add_line:
        add_line(mac_src)
        add_line(ip_src)
        add_line(str(load[srt+12:finish]))
        add_line(urllib.unquote(str(load[start + 1:end])))
        add_line(str(classed))
        add_line("end cap")
    if lognum:
        countsniff = countsniff + 1
        lognum(str(countsniff))
```

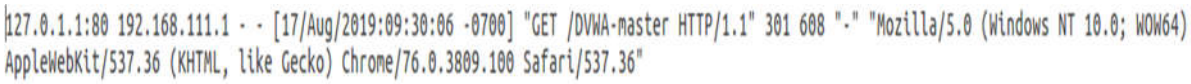
Với chức năng nghe trực tiếp qua web server log (ở đây là apache và nginx), ta mở file server log sinh từ config, chạy một vòng lặp vô hạn để đọc các dòng log mới. Vòng lặp dừng khi ta bấm Stop trên GUI. Mỗi khi đọc được dòng log mới sẽ đưa vài hàm phân loại.

Code python hàm cập nhật dòng apache’s log mới

```
def sniff_apache():
    global TSNIFF
    global workapa
    apachef = open("/var/log/apache2/other_vhosts_access.log")
    apachef.seek(0,2)
    while True:
        if not STOP_EV.is_set():
           line = apachef.readline()
           print line
           if not line:
              time.sleep(0.1)
              continue
```

```
            classify_live_apache(line, TSNIFF, LABEL, -1)
        else:
            apachef.close()
            break
```

Dòng log vào được phân tách các trường bằng chuỗi regex. Chuỗi regex này dùng cho định dạng log chuẩn (Common Log Format). Phân tách các trường như pcap và đưa vào mảng numpy. Gọi hàm phân loại lấy kết quả, kết hợp ghi ra file log riêng và in ra GUI.




Hình 3. 3 - Định dạng log mặc định của Apache

Code để thực hiện lọc Features đối với Apache's log:

```
def classify_live_apache(load, add_line, lognum, count):
    """phan loai apache"""
    global input_file
    input_file.flush()
    global STOP_EV
    load = load.replace("127.0.1.1:80 ","")
    print load
    regex = '(.*?) - - \[(.*?)\] "(.*?)" (\d+) (\d+) "(.*?)" "(.*?)"'
    res = re.match(regex, load)
    if res:
        try:
            for methodd in METHODS:
                if methodd in res.groups()[2].split(' ')[0]:
                    method = METHODS.index(methodd)+1
            res1 = res.groups()[2].split(' ')[1]
            url = 'http://localhost:8080' + res1.split('?')[0]
            url_length = len(url)
            special_path = re.findall('[^a-zA-Z\d\s\/:\.]', url) #Len
            non_an_path = re.findall('[^a-zA-Z\d\s]', url) #Len
            digit_path = re.findall('[0-9]', url) #Len
            host = 0

            try:
                arg = res1.split('?')[1]
                arg_num = re.findall('&', url)
            except IndexError:
                arg = ""
                arg_num = []
            arg_length = len(arg)
            digit_in_arg = re.findall('[0-9]', arg) #Len
            letter_in_arg = re.findall('[a-zA-Z]', arg) #Len
            contentLength = arg_length + len(arg_num)
```

Sau đó, sử dụng code sau để phân loại request và in ra màn hình:

```
test = np.array([[method, url_length, len(special_path),
len(special_path), len(non_an_path), len(digit_path), arg_length,
len(arg_num)+1, len(digit_in_arg), len(letter_in_arg),
contentLength]])
prediction = classifier.predict(test)
print test
result = list()
for w in prediction:
     result.append(w)
anomalous = True if 1 in result else False
if anomalous:
     classed = 'anomalous'
else:
     classed = 'normal'
if not STOP_EV.is_set():
    if (add_line or count==-1) and count!=-2:
       add_line(res.groups()[1].split(' ')[0])
       add_line(res.groups()[0])
       add_line(res.groups()[2].split(' ')[0])
       add_line(res.groups()[6])
       add_line(res1)
       add_line(res.groups()[3])
       add_line(res.groups()[4])
       add_line(classed)
       add_line("end cap")
    if add_line and count==-2:
       add_line(res.groups()[1].split(' ')[0])
       add_line(res.groups()[0])
       add_line(res.groups()[2].split(' ')[0])
       add_line(res.groups()[6])
       add_line(res1)
       add_line(res.groups()[4])
       add_line(classed)
       add_line("end cap")
    if lognum:
       if count!=-1 and count!=-2:
          lognum(str(count))
```

Với chức năng đọc log hoặc đọc file pcap, ta sử dụng FileChooser của Gtk để lấy đường dẫn tới file và gọi hàm đọc từng dòng giống như nghe trực tiếp. Nếu dòng thuộc log Apache sẽ được đưa vào module phân loại apache phía trên, thuộc pcap thì đưa vào hàm phân loại pcap ở phía trên.

Đoạn code python Hàm lấy đường dẫn file log:

```
def open_log(self, widget):
    dialog = Gtk.FileChooserDialog("Please choose a file", self,
```

```
                                        Gtk.FileChooserAction.OPEN,
                                        (Gtk.STOCK_CANCEL,
Gtk.ResponseType.CANCEL,
                                         Gtk.STOCK_OPEN, Gtk.ResponseType.OK))

        response = dialog.run()
        if response == Gtk.ResponseType.OK:
            print("Open clicked")
            print("File selected: " + dialog.get_filename())
            global addl
            global STOP_EV
            global lognum
            STOP_EV.clear()
            self.software_list_store_nginx.clear()
            self.noteb.set_current_page(1)
            worker1 = threading.Thread(target=live_core.start_nginx,
args=[addl, lognum, STOP_EV, dialog.get_filename()])
            worker1.start()
        elif response == Gtk.ResponseType.CANCEL:
            print("Cancel clicked")
        dialog.destroy()
```

### *3.1.2. Cấu trúc thư mục:*

- Dataset: Chứa bộ dữ liệu gốc và bộ dữ liệu mẫu sau khi trích xuất các trường.
- Features Extract: Code để tách lọc các trường cần thiết cho mỗi mô hình
- RF: Chứa các modules phục vụ cho phân tích dữ liệu:
  + *RF.py:* Phần xử lý tạo model và học dữ liệu, tính giá trị độ chính xác, độ phân loại, độ nhạy với mẫu thử. Phân loại log từ bên ngoài.
  + *live_test.py:* Xử lý dữ liệu vào từ gói tin trực tiếp, phân loại và in kết quả.
- Application: Chứa dữ liệu chạy chương trình dạng đồ hoạ

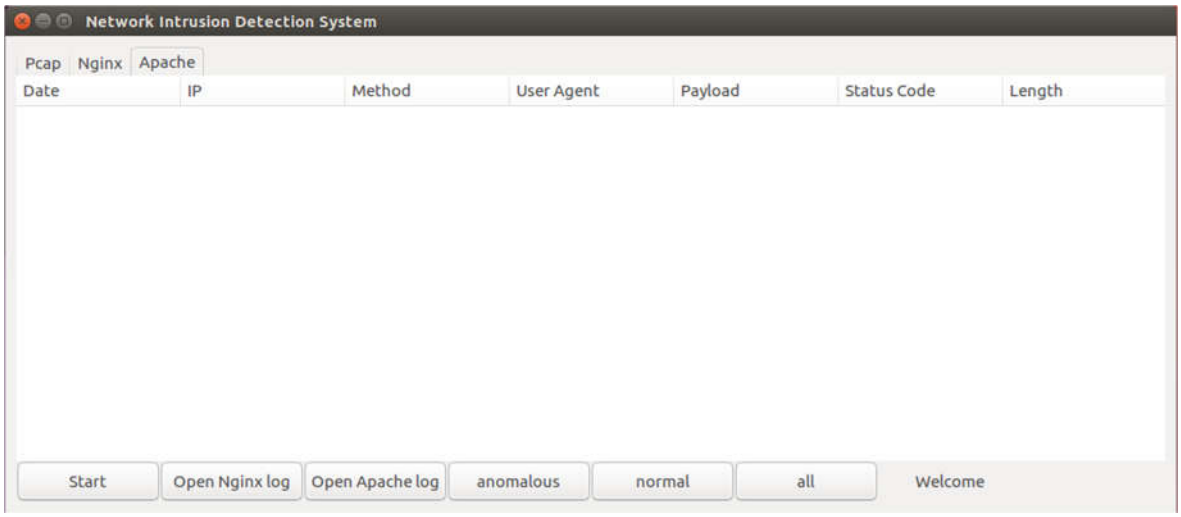
### *3.1.3. Cài đặt hệ thống:*

Chương trình xây dựng chạy trên môi trường Python2. Ứng dụng sử dụng một số thư viện cần thiết như sau:

- Numpy: hỗ trợ cho việc tính toán các mảng nhiều chiều, có kích thước lớn với các hàm đã được tối ưu áp dụng trên các mảng đó. Numpy hữu ích khi thực hiện các hàm liên quan tới đại số tuyến tính.
- Scapy: thư viện xử lý ngôn ngữ tự nhiên với các ví dụ, tài liệu API, hỗ trợ xử lý toàn bộ tài liệu mà không cần phải chia tài liệu thành các cụm từ.

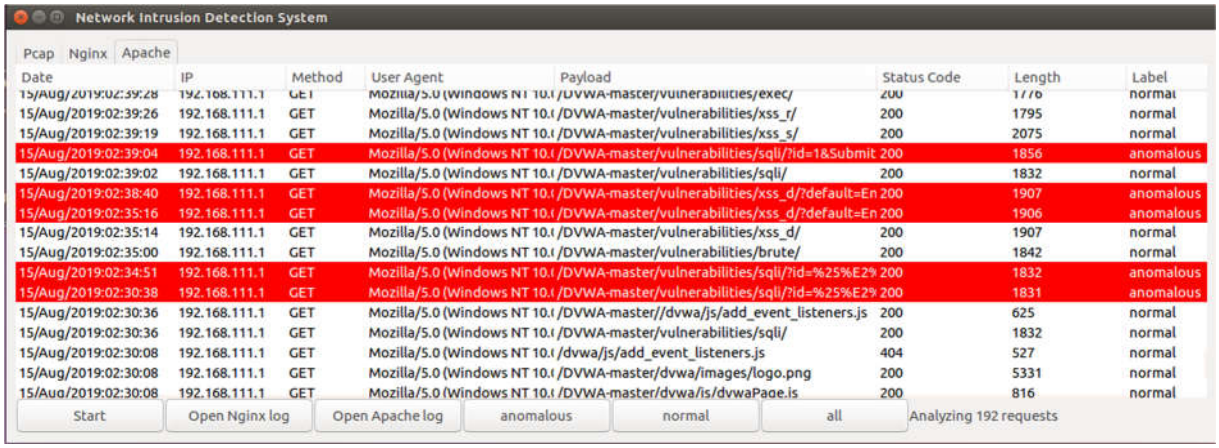
- Scikit-learn: cung cấp các thuật toán cho nhiều nhiệm vụ học tập và khai thác dữ liệu tiêu chuẩn của máy như phân cụm, hồi quy, phân loại, giảm kích thước và lựa chọn mô hình.
- Nltk: cung cấp các thuật toán xử lý ngôn ngữ tự nhiên, có thể xử lý và phân tích văn bản theo nhiều cách khác nhau, mã hóa và gắn thẻ, trích xuất thông tin.
- Gtk

  Chạy chương trình bằng file nids-mod.py trong "Application/gui"

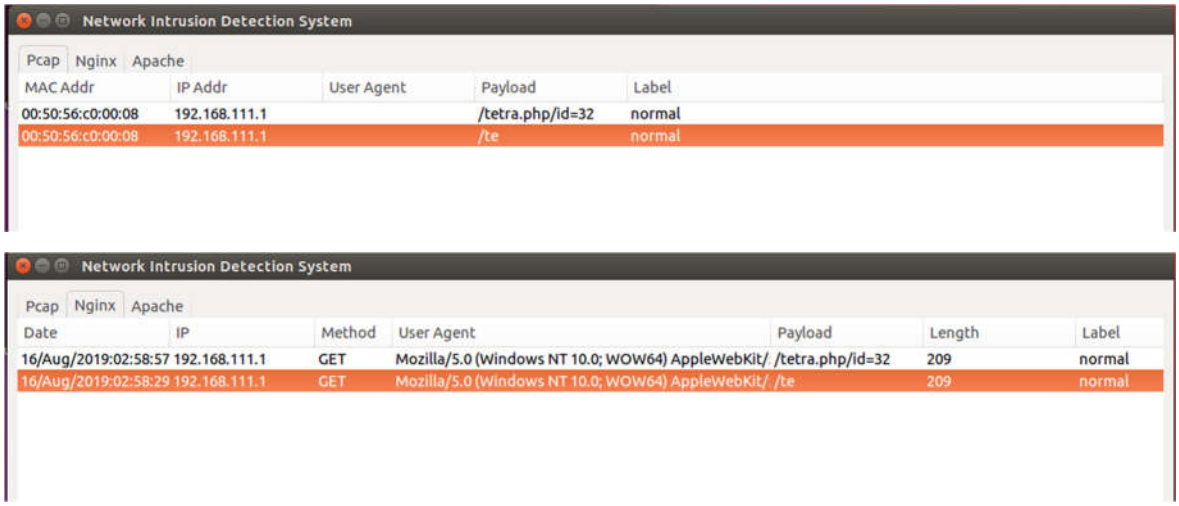



Hình 3. 4 - Giao diện chính chương trình

- Sử dụng giao diện Gtk, phần chính là một notebook với ba list dùng để hiển thị request từ pcap, nginx, apache,… Hiển thị một số trường cơ bản.
- Button Start dùng cho chế độ nghe trực tiếp request từ pcap, nginx log và apache log, đồng thời phân loại normal hoặc anomalous.
- Hai buttons "Open log" để mở các log sinh từ server và phân loại requests.
- Ba buttons tiếp để phân loại requests theo nhãn của chúng.




Hình 3. 5 -Đọc một file log từ Apache và phân loại




Hình 3. 6 - Đọc log trực tiếp từ pcap và trực tiếp từ nginx log

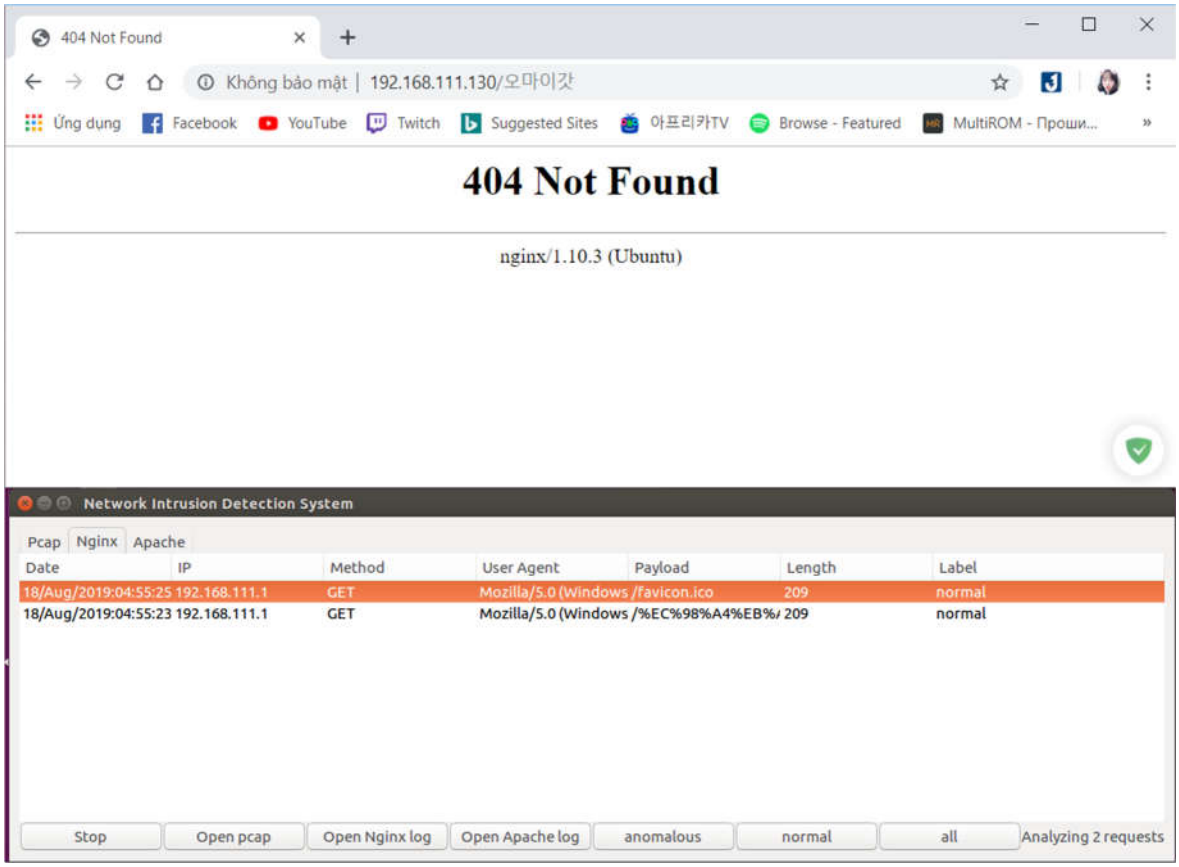
## 3.2. Một số kết quả thử nghiệm hệ thống

Máy server chạy ubuntu có IP 192.168.111.130. Được cài đặt hai web server apache và nginx. Apache được set vitual host nghe kết nối ở cổng 8080, nginx chạy mặc định ở cổng 80

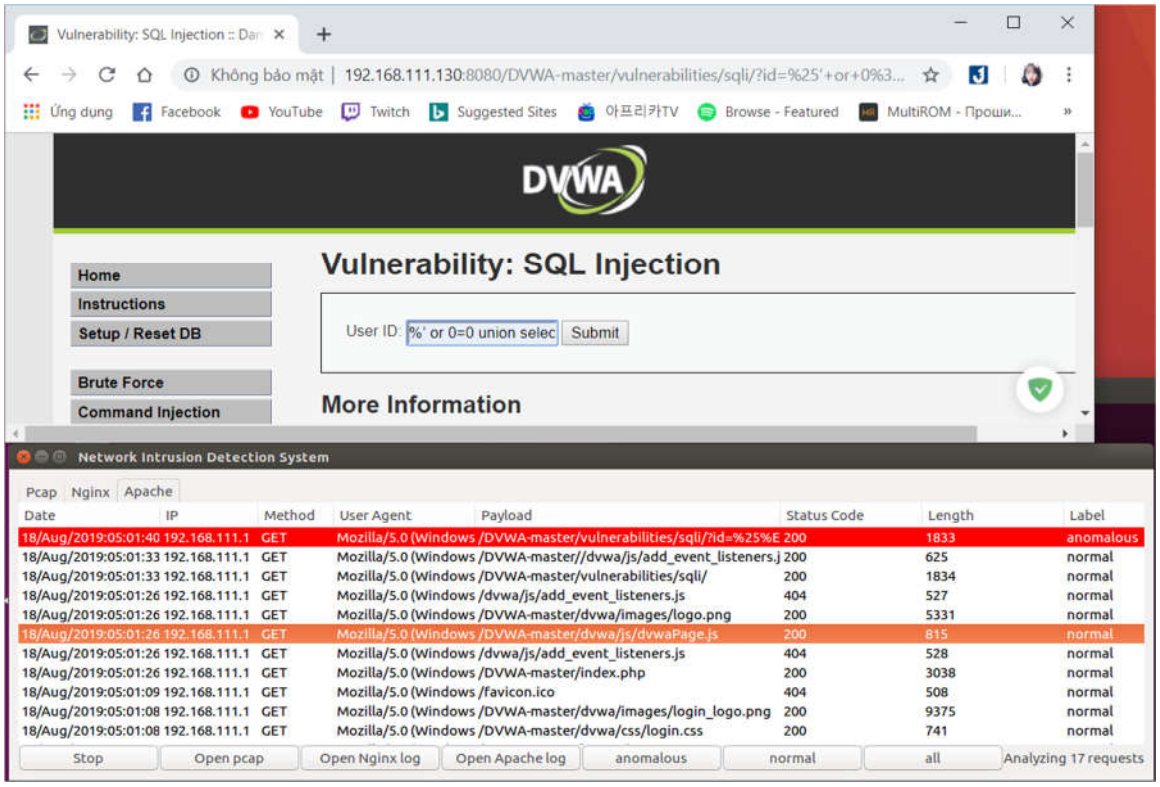

Hình 3. 7 -Cấu hình Apache và Nginx trên server

Mở chương trình bằng file nids.py.

Bắt đầu chế độ nghe bằng nút Start, chương trình sẽ lắng nghe các kết nối từ scapy sniff và file log apache và nginx. Sử dụng một máy bên ngoài truy cập vào nginx thông qua: 192.168.111.130, kết quả sẽ được in ra ở tab pcap và tab nginx




Hình 3. 8 - Nghe gói tin thông qua nginx log

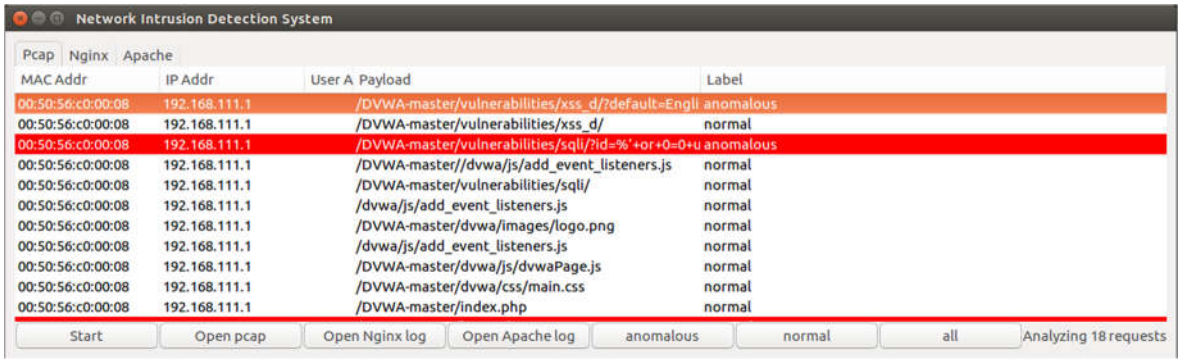



Hình 3. 9 -Nghe gói tin thông qua apache log

Sử dụng máy ngoài truy cập web của apache server thông qua máy chủ có IP 192.168.111.130:8080, khi có kết nối đến, kết quả gói tin được in tại tab pcap và tab apache.

Thử nghiệm lỗi SQL Injection thông qua DVWA được cài trên Apache server sẽ có kết quả như hình trên.

Mở đọc file pcap mới thu được:




Hình 3. 10 - Mở file pcap

Chương trình về cơ bản có thể đọc trực tiếp gói tin, đọc từ file log và phân loại các request tấn công đơn giản theo hướng URL và đối số Payload, hiển thị filter theo nhãn bình thường hay nguy hiểm.

# KẾT LUẬN VÀ KIẾN NGHỊ

## 4.1. Những đóng góp của luận văn

Trong khuôn khổ của luận văn tác giả đã tìm hiểu cơ sở lý thuyết và một số phương thức phát hiện tấn công ứng dụng web dựa trên log truy cập. Tác giả cũng đã tập trung nghiên cứu về thuật toán Random Forest và phương pháp ứng dụng thuật toán áp dụng vào việc phát hiện tấn công. Từ những kết quả thực nghiệm trên bộ dữ liệu HTTP CSIC 2010, chúng ta thấy kết quả tốt. Tuy nhiên phương pháp này có nhược điểm là thời gian chạy chương trình hơi lâu khi phân tích khối lượng dữ liệu lớn.

## 4.2. Hướng phát triển luận văn

Để giải quyết hạn chế của mô hình học máy được đề xuất ở trên, trong thời gian tới tác giả sẽ chú trọng tìm hiểu, cải tiến nhằm tăng tốc độ phân lớp của giải thuật. Đồng thời, tiến hành thử nghiệm phương pháp trên nhiều bộ dữ liệu khác nhau nhằm đánh giá độ chính xác và ổn định của phương pháp đối với từng loại dữ liệu cụ thể.

Tìm hiểu thêm một số phương pháp phân lớp khác như phương pháp hỗ trợ véc tơ (SVM) để so sánh với thuật toán random forest khi đánh giá kết quả dự đoán. So sánh hiệu quả giữa các phương pháp để có thêm lựa chọn khi phát triển các ứng dụng phát hiện tấn công bằng phương pháp phân lớp dữ liệu.

# CÁC TÀI LIỆU THAM KHẢO

[1] M. Zolotukhin, T. Hämäläinen, T. Kokkonen and J. Siltanen, "Analysis of HTTP Requests for Anomaly Detection of Web Attacks," *2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing*, Dalian, 2014, pp. 406-411.

[2] Michie, Donald, David J. Spiegelhalter, and C. C. Taylor. "Machine learning." Neural and Statistical Classification 13.1994 (1994): 1-298.

[3] Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.

[4] Meyer, Roger, and Carlos Cid. "Detecting attacks on web applications from log files." *Sans Institute* (2008).

**Tài liệu tham khảo từ Internet:**

[5] CLASSIFICATION – PART 3, https://tech.3si.vn/2016/03/31/ml-classification-part-3/. Truy cập 18/12/2019