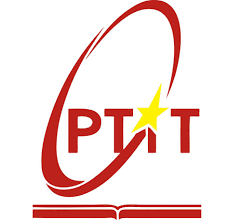
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA AN TOÀN THÔNG TIN**

**—----------------------------------------**



**BÀI TẬP LỚN**

**CƠ SỞ AN TOÀN THÔNG TIN**

**Đề tài: Phát hiện tấn công qua mạng**

|  |  |
| --- | --- |
| Giảng viên: | Đỗ Xuân Chợ |
| Nhóm: | 07 Lớp 02 |

Thành viên nhóm:

|  |  |
| --- | --- |
| Vũ Thành Long | B21DCAT012 |
| Nguyễn Duy Đạt | B21DCAT056 |

**Hà Nội - 2023**

**MỤC LỤC**

[**Chương 1: Tổng quan về phát hiện tấn công qua mạng 5**](#_30j0zll)

[**1.Tổng quan về phát hiện tấn công qua mạng 5**](#_1fob9te)

[**2. Nghiên cứu Suricata 5**](#_3znysh7)

[**3. Nghiên cứu ELK 5**](#_2et92p0)

[**Chương 2: CƠ SỞ LÝ THUYẾT VỀ SURICATA 6**](#_tyjcwt)

[**1. Giới thiệu về Suricata 6**](#_3dy6vkm)

[**2. Nhu cầu ứng dụng 6**](#_1t3h5sf)

[**3. Công nghệ áp dụng 6**](#_4d34og8)

[**3.1 Hệ thống phát hiện xâm nhập mạng IDS 6**](#_2s8eyo1)

[**3.2 Hệ thống ngăn chặn xâm nhập mạng IPS 7**](#_17dp8vu)

[**3.3 Công nghệ giám sát an ninh mạng NSM 7**](#_3rdcrjn)

[**3.4 Sử dụng PCAP log lại thông tin của lưu lượng dữ liệu mạng 7**](#_26in1rg)

[**4. Luật trong Suricata 8**](#_lnxbz9)

[**4.1. Giới thiệu 8**](#_35nkun2)

[**4.2. Rule Header 9**](#_1ksv4uv)

[**4.3. Rule Option 11**](#_44sinio)

[**5. Kiến trúc xử lý của Suricata 21**](#_2jxsxqh)

[**6. Phân tích giá trị từng thành phần của mỗi quá trình 23**](#_z337ya)

[**6.1 Packet Sniffer (Decoder) 23**](#_3j2qqm3)

[**6.2 Preprocessors 24**](#_1y810tw)

[**6.3 Detection Engine 24**](#_4i7ojhp)

[**6.4 Thành phần cảnh báo/logging 25**](#_2xcytpi)

[**Chương 3: CƠ SỞ LÝ THUYẾT VỀ ELK 26**](#_1ci93xb)

[**1.Một số khái niệm 26**](#_3whwml4)

[***a.Sự kiện (Event)* 26**](#_2bn6wsx)

[***b.Dữ liệu log sự kiện (Event log)* 27**](#_qsh70q)

[***c.Sự cố (Incident)* 27**](#_3as4poj)

[**2.Lựa chọn công nghệ 27**](#_1pxezwc)

[**3. Tìm hiểu nền tảng công nghệ ELK 32**](#_49x2ik5)

[**3.1. Giới thiệu ELK 32**](#_2p2csry)

[**3.2. ElasticSearch 32**](#_147n2zr)

[**Chương 4: Thực nghiệm 40**](#_3o7alnk)

[**Chủ đề 1: Xây dựng một website để phân tích các gói pcap và xem log. 40**](#_23ckvvd)

[**Chủ đề 2: Thực hiện quản lý, phát hiện tấn công các máy trong mạng LAN 45**](#_ihv636)

[**1.Mô hình mạng 45**](#_32hioqz)

[**2.Thực hiện tấn công, phát hiện trên suricata xem log và xem log trên elasticsearch 45**](#_1hmsyys)

[**TÀI LIỆU THAM KHẢO 50**](#_41mghml)

# Chương 1: Tổng quan về phát hiện tấn công qua mạng

## **Tổng quan về phát hiện tấn công qua mạng**

Phát hiện tấn công qua mạng là trụ cột quan trọng trong việc bảo vệ hệ thống và dữ liệu trước những mối đe dọa ngày càng tinh vi và đa dạng. Khả năng nhận biết và ngăn chặn các cuộc tấn công, hay các hình thức tấn công khác là yếu tố quyết định giữa sự an toàn và rủi ro trong môi trường kết nối mạng hiện đại. Quá trình phát hiện tấn công qua mạng không chỉ giúp bảo vệ tính toàn vẹn và an ninh của dữ liệu quan trọng, mà còn mang lại khả năng ngăn chặn kịp thời các mối đe dọa tiềm ẩn trước khi chúng gây hậu quả lớn. Sự đáp ứng nhanh chóng trong việc phát hiện và đối phó với các tấn công mạng không chỉ giúp giữ cho hệ thống hoạt động ổn định mà còn giảm thiểu rủi ro và tổn thất cho tổ chức. Nhờ vào khả năng liên tục theo dõi và phân tích dữ liệu mạng, việc phát hiện tấn công trở thành một phần quan trọng của chiến lược an ninh thông tin toàn diện, đảm bảo sự an toàn và ổn định cho môi trường kết nối mạng ngày nay.

Chủ đề phát hiện tấn công qua mạng sẽ cung cấp cho chúng ta phương pháp để phát hiện các đợt tấn công, biết cách ngăn chặn các cuộc tấn công vào hệ thống mà ta muốn bảo vệ. Trong chủ đề này ta sẽ sử dụng hệ thống phát hiện xâm nhập Suricata, tìm hiểu cách thức hoạt động và tính bảo mật của hệ thống Suricata và bài toán quản lý log với ELK.

## **2. Nghiên cứu Suricata**

Nghiên cứu cấu trúc, các chức năng của Suricata và các cài đặt, cấu hình trên hệ điều hành Linux. Nghiên cứu về các luật của Suricata, cấu trúc luật và các thành phần trong luật của Suricata. Viết một số luật đơn giản trong Suricata và thực thi các luật. Nghiên cứu, phân tích dấu hiệu của một số cuộc tấn công, hình thành các luật tương ứng với đặc điểm của các dạng tấn công và xâm nhập đó.

## **3. Nghiên cứu ELK**

Trong bài toán quản lý log chúng ta sẽ tập trung nghiên cứu giải pháp công nghệ ELK do công nghệ này hài hòa giữa chi phí đầu tư và khả năng đáp ứng các tiêu chí như mã nguồn mở, hỗ trợ mở rộng theo chiều ngang (Clusters), có một nền tảng tìm kiếm mạnh mẽ (ElasticSearch), hỗ trợ phân tích số liệu thu thập được (Analytic), quản lý log tập trung, tìm kiếm và thông báo lỗi một cách tự động.

# Chương 2: CƠ SỞ LÝ THUYẾT VỀ SURICATA

## **1. Giới thiệu về Suricata**

Suricata là một hệ thống phát hiện xâm nhập dựa trên mã nguồn mở. Nó được phát triển bởi Open Information Security Foundation (OISF). Công cụ này được phát triển không nhằm cạnh tranh hay thay thế các công tụ hiện có, nhưng nó sẽ mang lại những ý tưởng và công nghệ mới trong lĩnh vực an ninh mạng. Suricata là công cụ IDS/IPS (Intrusion Detection System / Intrusion Prevention System) phát hiện và ngăn chặn xâm nhập dựa trên luật để theo dõi lưu lượng mạng và cung cấp cảnh báo đến người quản trị hệ thống khi có sự kiện đáng ngờ xảy ra. Nó được thiết kế để tương thích với các thành phần an ninh mạng hiện có. Bản phát hành đầu tiên chạy trên nền tảng linux 2.6 có hỗ trợ nội tuyến (inline) và cấu hình giám sát lưu lượng thụ động có khả năng xử lý lưu lượng lên đến gigabit. Suricata là công cụ IDS/IPS miễn phí trong khi nó vẫn cung cấp những lựa chọn khả năng mở rộng cho các kiến trúc an ninh mạng phức tạp nhất. Là một công cụ đa luồng, Suricata cung cấp tăng tốc độ và hiệu quả trong việc phân tích lưu lượng mạng. Ngoài việc tăng hiệu quả phần cứng (với phần cứng và card mạng giới hạn), công cụ này được xây dựng để tận dụng khả năng xử lý cao được cung cấp bởi chip CPU đa lõi mới nhất.

## **2. Nhu cầu ứng dụng**

Với sự phát triển ngày càng nhiều các hình thức tấn công của tội phạm mạng (Cyber Crimes) hiện nay thì đối với một hệ thống mạng của doanh nghiệp, vấn đề an ninh phải được đặt lên hàng đầu. Sử dụng một tường lửa với những công nghệ hiện đại và các chức năng có thể ngăn chặn các cuộc tấn công của các hacker là một giải pháp hữu hiệu cho các doanh nghiệp trong việc phát hiện, ngăn chặn các mối đe dọa nguy hiểm để bảo vệ hệ thống mạng

## **3. Công nghệ áp dụng**

**3.1 Hệ thống phát hiện xâm nhập mạng IDS**

IDS (Intrusion Detection System) là hệ thống giám sát lưu thông mạng (có thể là phần cứng hoặc phần mềm), có khả năng nhận biết những hoạt động khả nghi hay những hành động xâm nhập trái phép trên hệ thống mạng trong tiến trình tấn công, cung cấp thông tin nhận biết và đưa ra cảnh báo cho hệ thống, nhà quản trị. IDS có thể phân biệt được các cuộc tấn công từ nội bộ hay tấn công từ bên ngoài.  
IDS phát hiện dựa trên các dấu hiệu đặc biệt về nguy cơ đã biết hay dựa trên so sánh lưu thông mạng hiện tại với baseline (thông số chuẩn của hệ thống có thể chấp nhận được) để tìm ra các dấu hiệu bất thường.

### **3.2 Hệ thống ngăn chặn xâm nhập mạng IPS**

IPS (Intrusion Prevention System) là một hệ thống có thể phát hiện và ngăn chặn sự xâm nhập từ bên ngoài vào các hệ thống máy tính. IPS là một phương pháp tiếp cận an ninh mạng bằng cách ưu tiên sử dụng các công nghệ tiên tiến để phát hiện và ngăn chặn các nỗ lực xâm nhập vào hệ thống máy tính. IPS kiểm tra các luồng lưu lượng ra/ vào một hệ thống máy tính hoặc mạng máy tính nhằm mục đích vi phạm an ninh. Nếu phát hiện mối đe dọa thì nó sẽ có những hành động bảo vệ như ngăn chặn gói tin hoặc ngắt toàn bộ kết nối. IPS kiểm tra, ghi chép lại một cách chi tiết các hành động đang đăng nhập vào hệ thống và gửi cảnh báo cho hệ thống hoặc quản trị mạng. Các IPS khác nhau có phương thức khác nhau trong việc kiểm tra các luồng dữ liệu để phát hiện các mối đe dọa, xâm nhập.

### **3.3 Công nghệ giám sát an ninh mạng NSM**

Giám sát an ninh mạng (Network Security Monitoring) là việc thu thập các thông tin trên các thành phần của hệ thống, phân tích các thông tin, dấu hiệu nhằm đánh giá và đưa ra các cảnh báo cho người quản trị hệ thống.

Đối tượng của giám an ninh mạng là tất cả các thành phần, thiết bị trong hệ thống mạng:  
- Các máy trạm  
- Cơ sở dữ liệu  
- Các ứng dụng  
- Các server  
- Các thiết bị mạng

### **3.4 Sử dụng PCAP log lại thông tin của lưu lượng dữ liệu mạng**

PCAP (packet capture) bao gồm những giao diện lập trình ứng dụng (API) dùng để chặn bắt và phân tích các lưu lượng dữ liệu trên mạng. PCAP thực hiện các chức năng lọc gói dữ liệu theo những luật của người dùng khi chúng được truyền tới ứng dụng, truyền những gói dữ liệu thô tới mạng, thu thập thông tin thống kê lưu lượng mạng. Đối với các hệ thống thuộc họ Unix ta có thư viện libpcap, còn đối với Window ta có thư viện được port từ libpcap là winpcap

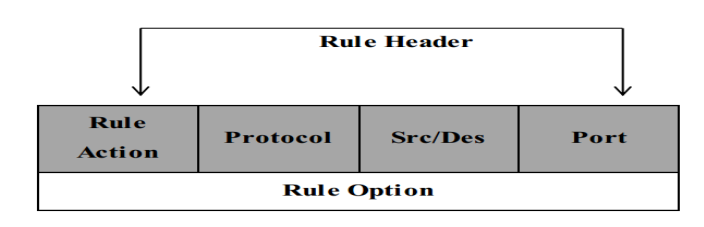
## **4. Luật trong Suricata**

**4.1. Giới thiệu**

“Luật” (Rule) trong Suricata ta có thể hiểu một cách đơn giản nó giống như các quy tắc và luật lệ trong thế giới thực. Nghĩa là nó sẽ có phần mô tả một trạng thái và hành động gì sẽ xảy ra khi trạng thái đó đúng. Một trong những điểm đáng giá nhất của Suricata đó là khả năng cho phép người sử dụng có thể tự viết các luật của riêng mình hoặc tùy biến các luật có sẵn cho phù hợp với hệ thống mạng của mình. Ngoài một cơ sở dữ liệu lớn mà người sử dụng có thể download từ trang chủ của Suricata, người quản trị có thể tự  
phát triển các luật cho hệ thống của mình. Thay vì phải phụ thuộc vào nhà cung cấp, một cơ quan bên ngoài, hoặc phải cập nhật khi có một cuộc tấn công mới hay một phương pháp khai thác lỗ hổng mới được phát hiện. Người quản trị có thể viết riêng một luật dành cho hệ thống của mình khi nhìn thấy các lưu lượng mạng bất thường và so sánh với bộ luật được cộng đồng phát triển. Ưu điểm của việc tự viết các luật là có thể tùy biến và cập nhật một cách cực kỳ nhanh chóng khi hệ thống mạng có sự bất thường.

Để biết cách viết một luật từ các dữ liệu của hệ thống ta cần phải hiểu cấu trúc của luật trong Suricata như thế nào. Một luật trong Suricata được chia thành hai phần đó là phần ***rule header*** và ***rule options***. Phần rule header bao gồm: rule action, protocol, địa chỉ ip nguồn, địa chỉ ip đích, subnet mask, port nguồn, port đích. Phần options bao gồm các thông điệp cảnh báo, thông tin các phần của gói tin sẽ được kiểm tra để xác định xem hành động nào sẽ được áp dụng.

Để biết cách viết một luật từ các dữ liệu của hệ thống ta cần phải hiểu cấu trúc của luật trong Suricata như thế nào. Một luật trong Suricata được chia thành hai phần đó là phần ***rule header*** và ***rule options***. Phần rule header bao gồm: rule action, protocol, địa chỉ ip nguồn, địa chỉ ip đích, subnet mask, port nguồn, port đích. Phần options bao gồm các thông điệp cảnh báo, thông tin các phần của gói tin sẽ được kiểm tra để xác định xem hành động nào sẽ được áp dụng.



*Hình 2.1: Cấu trúc luật trong Suricata*

**4.2. Rule Header**

**4.2.1. Rule Action**

Phần Header sẽ chứa các thông tin xác định ai, ở đâu, cái gì của một gói tin, cũng như phải làm gì nếu tất cả các thuộc tính trong luật được hiện lên.

Mục đầu tiên trong một luật đó chính là phần rule action, rule action sẽ nói cho Suricata biết phải làm gì khi thấy các gói tin phù hợp với các luật đã được quy định sẵn. Có 4 hành động mặc định trong Suricata đó là: pass (cho qua), drop (chặn gói tin), reject, alert (cảnh báo).  
∙ ***Pass:*** nếu signature được so sánh trùng khớp và chỉ ra là pass thì Suricata sẽ thực hiện dừng quét gói tin và bỏ qua tất cả các luật phía sau đối với gói tin này.  
∙ ***Drop:*** nếu chương trình tìm thấy một signature hợp lệ và nó chỉ ra là drop thì gói tin đó sẽ bị hủy bỏ và dừng truyền ngay lập tức, khi đó gói tin không thể đến được nơi nhận.  
∙ ***Reject:*** là hành động bỏ qua gói tin, bỏ qua ở cả bên nhận và bên gửi. Suricata sẽ tạo ra một cảnh báo với gói tin này.  
∙ ***Alert:*** nếu signature được so sánh là hợp lệ và có chứa một alert thì gói tin đó sẽ được xử lý giống như với một gói tin không hợp lệ. Suricata sẽ tạo ra một cảnh báo.

**4.2.2. Protocol**

Trường tiếp theo trong luật đó là protocol. Các giao thức mà Suricata hiện đang phân tích các hành vi bất thường đó là TLS, SSH, SMTP (tải thư điện tử qua mạng internet), IMAP (đặt sự kiểm soát email trên mail server), MSN, SMB (chia sẻ file), TCP, UDP, ICMP và IP, DNS.

**4.2.3. IP Address**

Mục tiếp theo của phần header đó là địa chỉ IP. Các địa chỉ này dùng để kiểm tra nơi đi và nơi đến của một gói tin. Địa chỉ ip đó có thể là địa chỉ của một máy đơn hoặc cũng có thể là địa chỉ của một lớp mạng. Từ khóa “any” được sử dụng để định nghĩa một địa chỉ bất kỳ.

Một địa chỉ ip sẽ được viết dưới dạng *ip\_address/netmask*. Điều này có nghĩa là nếu netmask là /24 thì lớp mạng đó là lớp mạng C, /16 là lớp mạng B hoặc /32 là chỉ một máy đơn. Ví dụ: địa chỉ 192.168.1.0/24 có nghĩa là một dải máy có địa chỉ IP từ 192.168.1.1-192.168.1.255. Trong hai địa chỉ IP trong một luật Suricata thì sẽ có một địa chỉ IP nguồn và một địa chỉ IP đích. Việc xác định đâu là địa chỉ nguồn, đâu là địa chỉ đích phụ thuộc vào “→”. Ngoài ra toán tử phủ định có thể được áp dụng cho việc định địa chỉ IP. Có nghĩa là khi sử dụng toán tử này thì Suricata sẽ bỏ qua việc kiểm tra địa chỉ của gói tin đó. Toán tử đó là “!”. Ngoài ra ta có thể định nghĩa một danh sách các địa chỉ IP bằng cách viết liên tiếp chúng cách nhau bởi một dấu “,”.  
Ví dụ:

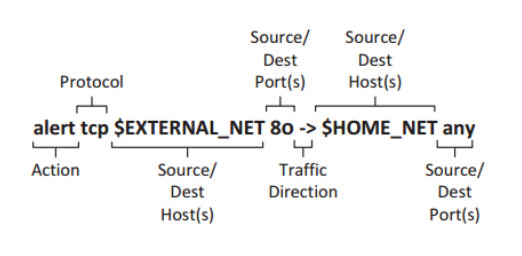
*Alert TCP any any → ![192.168.1.0/24, 172.16.0.0/16] 80 (msg: “Access”)*

**4.2.4. Port**

Port có thể được định nghĩa bằng nhiều cách. Với từ khóa “any” giống như địa chỉ IP để chỉ có thể sử dụng bất kỳ port nào. Gán một port cố định, ví dụ như gán kiểm tra ở port 80 http hoặc port 22 ssh. Ngoài ra ta cũng có thể sử dụng toán tử phủ định để bỏ qua một port nào đó hoặc liệt kê một dải các port.  
Ví dụ:  
*Alert UDP any any → 192.168.1.0/24 1:1024 - port bất kỳ tới dãy port từ 1 - 1024.  
Alert UDP any any → 192.168.1.0/24 :6000 - port bất kỳ tới dãy port nhỏ hơn 6000.  
Alert UDP any any → 192.168.1.0/24 !6000:6010 - port bất kỳ tới bất kỳ port nào, bỏ qua dải port từ 6000 – 6010.*

*4.2.5. Điều hướng*

Toán tử hướng “→” chỉ ra đâu là hướng nguồn, đâu là hướng đích. Phần địa chỉ IP và port ở phía bên trái của toán tử được coi như là địa chỉ nguồn và port nguồn, phần bên phải được coi như địa chỉ đích và port đích. Ngoài ra còn có toán tử “<>” Suricata sẽ xem cặp địa chỉ/port nguồn và đích là như nhau. Nghĩa là nó sẽ ghi/phân tích ở cả hai phía của cuộc hội thoại.



*Hình 2.2: Thứ tự ưu tiên phân tích gói tin cho hệ thống xử lý*

Ví dụ:

*Alert TCP !192.168.1.0/24 any <> 192.168.1.0/24 23*

### **4.3. Rule Option**

Rule options chính là trung tâm của việc phát hiện xâm nhập. Nội dung chứa các dấu hiệu để xác định một cuộc xâm nhập. Nó nằm ngay sau phần Rule Header và được bọc bởi dấu ngoặc đơn “()”. Tất cả các rule options sẽ được phân cách nhau bởi dấu chấm phẩy “;”, phần đối số sẽ được tách ra bởi dấu hai chấm “:”. Có 4 loại rule options chính bao gồm:  
***- General:*** Tùy chọn này cung cấp thông tin về luật đó nhưng không có bất cứ ảnh hưởng nào trong quá trình phát hiện.  
***- Payload:*** Tùy chọn liên quan đến phần tải trong một gói tin.  
***- Non-payload:*** Bao gồm các tùy chọn không liên quan đến phần tải của gói tin (header).  
***- Post-detection:*** Các tùy chọn này sẽ gây ra những quy tắc cụ thể sau khi một luật đã được kích hoạt.

*Các thành phần khác trong Rule:*

**4.3.1. General**

* ***Msg***

Msg (Message): được dùng để cho biết thêm thông tin về từng signature và các cảnh báo. Phần đầu tiên sẽ cho biết tên tập tin của signature và phần này quy ước là phải viết bằng chữ in hoa. Định dạng của msg như sau:

*msg: “..........”;*

* ***Sid***

Sid (signature id): cho ta biết định danh riêng của mỗi signature. Định danh này được bắt đầu với số. Định dạng của sid như sau:

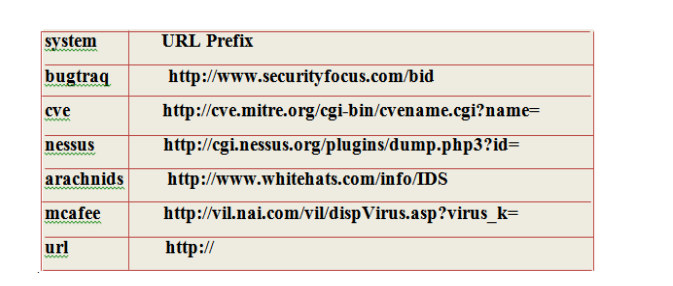
*sid:123;*

* ***Rev***Rev (revision): mỗi sid thường đi kèm với một rev. Rev đại diện cho các phiên bản của signature. Mỗi khi signature được sửa đổi thì số rev sẽ được tăng lên bởi người tạo ra. Định dạng của rev như sau:

*rev:123;*

* ***Reference***Reference: cung cấp cho ta địa chỉ đến được những nơi chứa các thông tin đầy đủ về signature. Các tham chiếu có thể xuất hiện nhiều lần trong một signature. Ví dụ về một tham chiếu như sau:

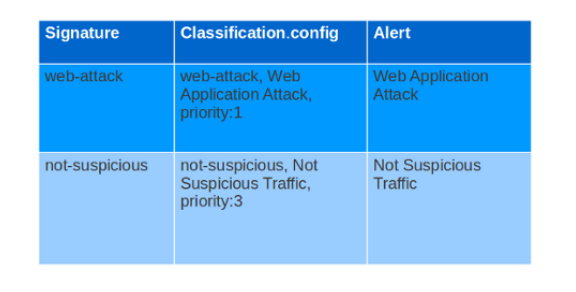
*reference: url,* [*www.info.nl*](http://www.info.nl)



*Hình 2.3: Bảng các tuỳ chọn của Reference*

* ***Classtype***Classtype: cung cấp thông tin về việc phân loại các lớp quy tắc và cảnh báo. Mỗi lớp bao gồm một tên ngắn gọn, một tên đầy đủ và mức độ ưu tiên.  
  Ví dụ:

*config classification: web-application-attack,Web Application Attack,1  
config classification: not-suspicious,Not Suspicious Traffic,3*



*Hình 2.4: Thông tin phân loại lớp quy tắc*

* ***Priority***Priority: chỉ ra mức độ ưu tiên của mỗi signature. Các giá trị ưu tiên dao động từ 1 đến 255, nhưng thường sử dụng các giá trị từ 1 -> 4. Mức ưu tiên cao nhất là 1. Những signature có mức ưu tiên cao hơn sẽ được kiểm tra trước. Định dạng như sau:

*priority:1;*

* ***Metadata***Metadata: Suricata sẽ bỏ qua những gì viết sau metadata. Định dạng như sau: *metadata:......;*

**4.3.2. Payload**

* ***Content***Content: thể hiện nội dung chúng ta cần viết trong signature, nội dung này được đặt giữa 2 dấu nháy kép. Nội dung là các byte dữ liệu, có 256 giá trị khác nhau (0-255). Chúng có thể là các ký tự thường, ký tự hoa, các ký tự đặc biệt, hay là các mã hex tương ứng với các ký tự và các mã hex này phải được đặt giữa 2 dấu gạch dọc. Định dạng của một nội dung như sau:

*content: ”............”;*

* ***Nocase***Nocase: được dùng để chỉnh sửa nội dung thành các chữ thường, không tạo ra sự khác biệt giữa chữ hoa và chữ thường. Nocase cần được đặt sau nội dung cần chỉnh sửa. Ví dụ:

*content: “abC”; nocase;*

* ***Depth***Depth: sau từ khóa depth là một số, chỉ ra bao nhiêu byte từ đầu một payload cần được kiểm tra. Depth cần được đặt sau một nội dung. Ví dụ: ta có một payload : abCdefghij. Ta thực hiện kiểm tra 3 byte đầu của payload.

*content: “abC”; depth:3;*

* ***Offset***Offset: chỉ độ lệch byte trong tải trọng sẽ được kiểm tra. Ví dụ: độ lệch là 3 thì sẽ kiểm tra từ byte thứ 4 trong tải trọng.

*content: “def”; offset:3;*

Ví dụ:  
*Alert TCP 192.168.1.0/24 any -> any any (content: \"HTTP"; offset: 4; depth: 40; msg: "HTTP matched";)*

* ***Distance***Distance: xác định khoảng cách giữa các nội dung cần kiểm tra trong payload. Khoảng cách này có thể là một số âm. Ví dụ:

*content: “abC”; content: “efg”; distance:1;*

* ***Within***Within: được dùng cùng với distance, để chỉ độ rộng của các byte cần kiểm tra sau một nội dung với khoảng cách cho trước đó. Ví dụ:

*content:"GET"; depth:3 content:"download"; distance:10 \within:9;*

Luật có nghĩa là tìm “GET” trong 3 byte đầu tiên của trường dữ liệu, di chuyển thêm 10 byte bắt đầu từ “GET” và tìm khớp “download”. Tuy nhiên, “download” phải xuất hiện trong 9 byte tiếp theo.

* ***Dsize***Dsize: được dùng để tìm một payload có độ dài bất kỳ.

*dsize:min<>max;*

* ***Rpc***Rpc (Remote Procedure Call): là một ứng dụng cho phép một chương trình máy tính thực hiện một thủ tục nào đó trên một máy tính khác, thường được sử dụng cho quá trình liên lạc. Định dạng của rpc như sau:

*rpc:<application number>, [<version number>|\*], [<procedure number>|\*]>;*

* ***Replace***Replace được dùng để thay đổi nội dung của payload, điều chỉnh lưu lượng mạng. Việc sửa đổi nội dung của payload chỉ có thể được thực hiện đối với gói dữ liệu cá nhân. Sau khi thực hiện thay đổi nội dung xong thì Suricata sẽ thực hiện tính toán lại trường checksum.

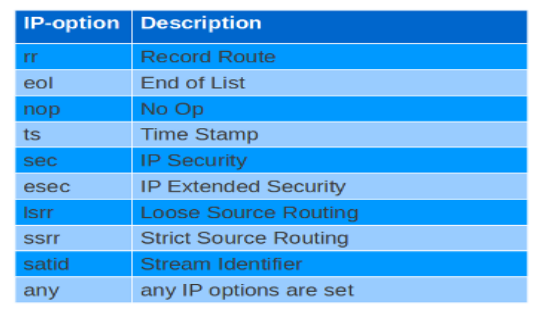
**4.3.3. Non-Payload**

**a. IP**

* ***ttl***Được sử dụng để kiểm tra về thời gian sống, tồn tại tên mạng của một địa chỉ IP cụ thể trong phần đầu của mỗi gói tin. Giá trị time-to-live (thời gian sống), xác định thời gian tối đa mà mỗi gói tin có thể được lưu thông trên hệ thống mạng. Nếu giá trị này về 0 thì gói tin sẽ bị hủy bỏ. Thời gian sống được xác định dựa trên số hop, khi đi qua mỗi hop/router thì thời gian sống sẽ bị trừ đi 1. Cơ chế này nhằm hạn chế việc gói tin lưu thông trên mạng vô thời hạn. Định dạng của một ttl như sau:

*ttl:<number>;*

* ***ipopts***Chúng ta có thể xem và tùy chỉnh các tùy chọn cho việc thiết lập các địa chỉ IP. Việc thiết lập các tùy chọn cần được thực hiện khi bắt đầu một quy tắc. Một số tùy chọn có thể sử dụng:



*Hình 2.5: Một số tùy chọn của Ipopts*

Định dạng của một ipopts như sau:

*ipopts: <name>;*

* ***sameip***

Mỗi gói tin sẽ có một địa chỉ IP nguồn và đích. Chúng ta có thể sử dụng sameip để kiểm tra xem địa chỉ IP nguồn và đích có trùng nhau hay không. Định dạng của sameip như sau:

*sameip;*

* ***Ip\_proto***

Được dùng để giúp ta lựa chọn giao thức. Ta có thể chọn theo tên hoặc số tương ứng với từng giao thức. Có một số giao thức phổ biến sau:

*1 ICMP Internet Control Message*

*6 TCP Transmission Control Protocol*

*17 UDP User Datagram*

*47 GRE General Routing Encapsulation*

*50 ESP Encap Security Payload for IPv6*

*51 AH Authentication Header for Ipv6*

*58 IPv6-ICMP ICMP for Ipv6*

Định dạng của ip\_proto như sau:

*ip\_proto:<number/name>;*

* ***Id***Được sử dụng để định danh cho các phân mảnh của gói tin được truyền đi. Khi gói tin truyền đi sẽ được phân mảnh, và các mảnh của một gói tin sẽ có ID giống nhau. Việc này giúp ích cho việc ghép lại gói tin một cách dễ dàng.  
  Định dạng như sau:

*id:<number>;*

* ***Geoip***Cho phép xác định địa chỉ nguồn, đích để gói tin lưu thông trên mạng.
* ***Fragbits***Được dùng để kiểm tra các phân mảnh của gói tin. Nó bao gồm các cơ  
  chế sau:

*M - More Fragments*

*D - Do not Fragment*

*R - Reserved Bit*

*+ match on the specified bits, plus any others*

*\* match if any of the specified bits are set*

*! match if the specified bits are not set*

Định dạng của một Fragbits như sau:

*fragbits:[\*+!]<[MDR]>;*

* ***Fragoffset***Kiểm tra sự phù hợp trên các giá trị thập phân của từng mảnh gói tin trên trường offset. Nếu muốn kiểm tra phân mảnh đầu tiên của gói tin, chúng ta cần kết hợp frag offset 0 với các tùy chọn fragment khác. Các tùy chọn fragment như sau:

*< match if the value is smaller than the specified value> match if the value is greater than the specified value! match if the specified value is not present*

Định dạng của fragoffset:

*fragoffset:[!|<|>]<number>;*

**b. TCP**

* ***Sed***

Là một số ngẫu nhiên được tạo ra ở cả bên nhận và bên gửi gói tin để kiểm tra số thứ tự của các gói tin đến và đi. Máy khách và máy chủ sẽ tự tạo ra một số seq riêng của mình. Khi một gói tin được truyền thì số seq này sẽ tăng lên 1. Seq giúp chúng ta theo dõi được những gì diễn ra khi một dòng dữ liệu được truyền đi.

* ***Ack***Được sử dụng để kiểm tra xem gói tin đã được nhận bởi nơi nhận hay chưa trong giao thức kết nối TCP. Số thứ tự của ACK sẽ tăng lên tương ứng với số byte dữ liệu đã được nhận thành công.
* ***Window***Được sử dụng để kiểm tra kích thước của cửa sổ TCP. Kích thước cửa sổ TCP là một cơ chế dùng để kiểm soát các dòng dữ liệu. Cửa sổ được thiết lập bởi người nhận, nó chỉ ra số lượng byte có thể nhận để tránh tình trạng bên nhận bị tràn dữ liệu. Giá trị kích thước của cửa sổ có thể chạy từ 2 đến 65.535 byte.

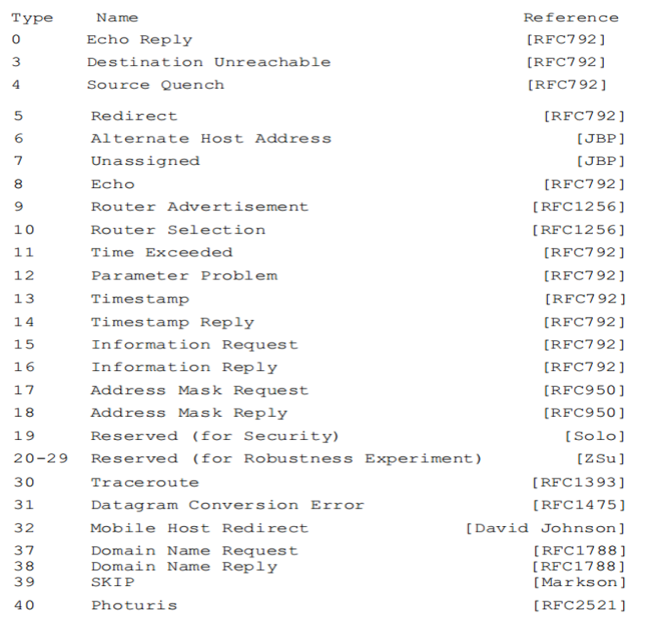
**c. ICMP**

* ***Itype***Cung cấp cho việc xác định các loại ICMP. Các thông điệp khác nhau sẽ được phân biệt bởi các tên khác nhau hay các giá trị khác nhau.

Định dạng của itype như sau:

*itype:min<>max;*

*itype:[<|>]<number>;*



*Hình 2.6: Bảng Type của ICMP Header*

* ***Icode***Cho phép xác định mã của từng ICMP để làm rõ hơn cho từng gói tin ICMP. Định dạng của icode như sau:

*icode:min<>max;  
icode:[<|>]<number>;*

* ***Icmp\_id***Mỗi gói tin ICMP có một giá trị ID khi chúng được gửi. Tại thời điểm đó, người nhận sẽ trả lại tin nhắn với cùng một giá trị ID để người gửi sẽ nhận ra và kết nối nó đúng với yêu cầu ICMP đã gửi trước đó. Định dạng của một icmp\_id như sau:

*icmp\_id:<number>;*

* ***Icmp\_seq***Được sử dụng để kiểm tra số thứ tự của ICMP. Định dạng của icmp\_seq như sau:

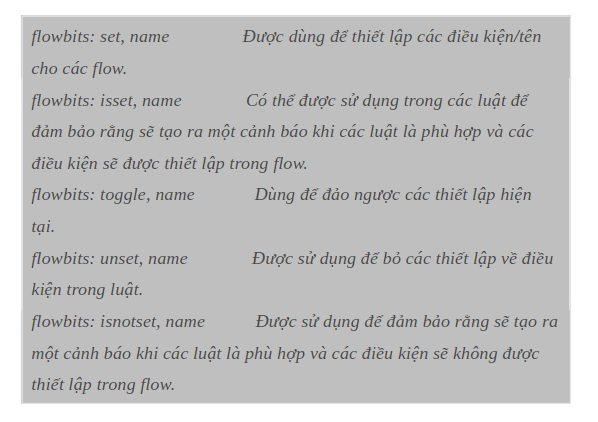
*icmp\_seq:<number>;*

**d. HTTP**

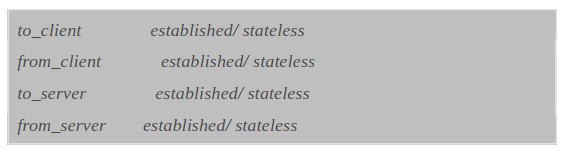
* ***http\_method***Chỉ ra các phương thức được áp dụng với các request http. Các phương thức http: GET, POST, PUT, HEAD, DELETE, TRACE, OPTIONS, CONNECT và PATCH.
* ***http\_uri và http\_raw\_uri***Chỉ ra đường dẫn tới nơi chứa nội dung yêu cầu.
* ***http\_header***Chỉ ra phương thức sử dụng, địa chỉ cần truy cập tới và tình trạng kết nối.
* ***http\_user\_agent***Là một phần của http\_header, chỉ ra thông tin về trình duyệt của người dùng.
* ***http\_client\_body***Chỉ ra các yêu cầu của máy trạm.
* ***http\_stat\_code***Chỉ ra mã trạng thái của server mà máy trạm yêu cầu kết nối tới.
* ***http\_stat\_msg***Các dòng tin thông báo về tình trạng máy chủ, hay tình trạng về việc đáp ứng các yêu cầu kết nối của máy trạm.
* ***http\_server\_body***Chỉ ra nội dung đáp trả các yêu cầu từ máy trạm của máy chủ.
* ***File\_data***Chỉ ra nội dung, đường dẫn tới file chứa dữ liệu được yêu cầu.

**e. Flow**

* ***Flowbits***Gồm 2 phần, phần đầu mô tả các hành động được thực hiện, phần thứ 2 là tên của flowbit. Các hành động của flowbit:



* ***Flow***Có thể được sử dụng để kết nối các thư mục chứa các flow lại với nhau. Các flow có thể được đi từ hoặc đến từ Client/Server và các flow này có thể ở trạng thái được thiết lập hoặc không. Việc kết nối các flow có thể xảy ra các trường hợp sau:



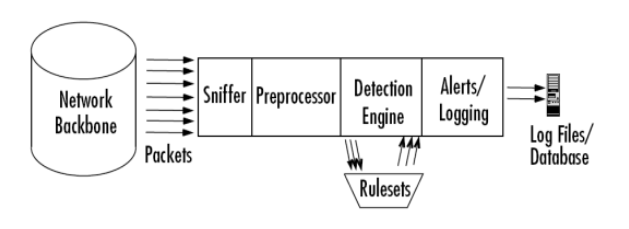
## 

## **5. Kiến trúc xử lý của Suricata**

* 1. **Các quá trình xử lý**

Suricata được phát triển dựa trên Snort nên nó vẫn giữ nguyên kiến trúc bên trong của Snort. Kiến trúc của nó có nhiều thành phần, với mỗi thành phần có một chức năng riêng.  
Kiến trúc của Suricata gồm 4 phần cơ bản sau:  
∙ The Sniffer (Packet Decoder) .  
∙ The Preprocessors.  
∙ The Detection Engine.  
∙ The Output: gồm hai modules

* Modul Alert/ Logging
* Modul kết xuất thông tin



*Hình 2.9: Kiến trúc của Suricata*

**5.2 Ý nghĩa của từng quá trình**

***5.2.1 Packet Sniffer (Decoder)***

- Khi Suricata hoạt động nó sẽ thực hiện lắng nghe và thu bắt tất cả các gói tin nào di chuyển qua nó.  
- Các gói tin sau khi bị bắt được đưa và module Sniffer, tại đây các gói tin sẽ được giải mã.

***5.2.2 Preprocessors***

- Tiếp theo gói tin sẽ được đưa vào module Preprocessor, tại đây gói tin sẽ được phân tích một cách chi tiết và đầy đủ theo các cách khác nhau.

***5.2.3 Detection Engine***

-Sau khi phân tích xong các gói tin được đưa vào module Detection. Tại đây, tùy theo việc có phát hiện được xâm nhập hay không mà gói tin có thể được lưu thông tiếp hay được đưa vào module Alert/ Logging để xử lý.

***5.2.4 Thành phần cảnh báo/logging***

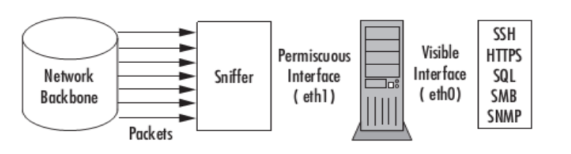
- Khi các cảnh báo được xác định module kết xuất thông tin sẽ thực hiện việc đưa cảnh báo ra theo đúng định dạng mong muốn.

Preprocessors, detection engine và alert system đều là các plug-ins. Điều này giúp cho việc chỉnh sửa hệ thống theo mong muốn của người quản trị một cách dễ dàng

## **6. Phân tích giá trị từng thành phần của mỗi quá trình**

### **6.1 Packet Sniffer (Decoder)**

Packet Sniffer là một thiết bị phần cứng hoặc phần mềm được đặt vào trong mạng. Chức năng của nó tương tự như việc nghe lén trên điện thoại di động, nhưng thay vì hoạt động trên mạng điện thoại nó nghe lén trên mạng dữ liệu. Bởi vì trong mô hình mạng có nhiều giao thức cao cấp như TCP, UDP, ICMP... nên công việc của packet sniffer là nó phải phân tích các giao thức đó thành thông tin mà con người có thể đọc và hiểu được. Packet Sniffer có thể được sử dụng với các mục đích như:  
∙ Phân tích mạng và troubleshooting.  
∙ Performance network and benchmarking.  
∙ Nghe lén mật khẩu clear-text và những dữ liệu khác.

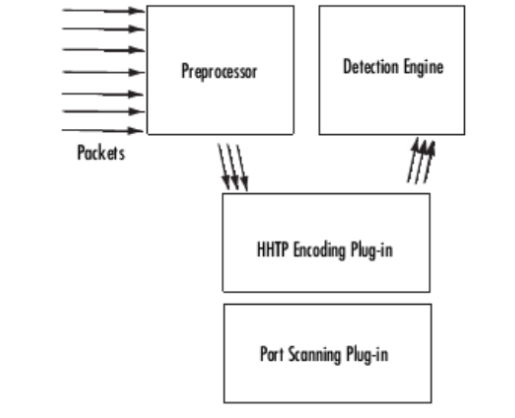


*Hình 2.10: Các gói tin đi vào Sniffer*

Khi Suricata đã nhận các gói tin từ quá trình sniffer nó sẽ đi vào quá trình giải mã. Chính xác thì nơi mà gói tin đi vào bộ giải mã phụ thuộc vào lớp liên kết mà trước đó đọc được. Snort hỗ trợ một số lớp liên kết từ pcap: Ethernet, 802.11, Token ring, FDDI, Cisco HDLC, SLIP, PPP và OpenBSD’s PF. Ở trên lớp liên kết Suricata hỗ trợ giải mã các giao thức khác nhau, bao gồm: IP, ICMP, TCP, UDP (chi tiết trong mã nguồn src/decode.c) Bất kể là lớp liên kết nào đang được sử dụng, tất cả các bộ giải mã sẽ đều làm việc theo một kiểu chung. Đối với trường hợp các lớp cụ thể, con trỏ trong cấu trúc của gói tin sẽ được thiết lập trỏ tới một phần khác của gói tin. Dựa vào các thông tin đã giải mã được, nó sẽ gọi các lớp cao hơn và giải mã cho đến khi không còn bộ giải mã nào nữa.

### **6.2 Preprocessors**

Preprocessors là plug-in cho phép phân tích cú pháp dữ liệu theo những cách khác nhau. Nếu chạy Suricata mà không có bất cứ cấu hình nào về preprocessors trong tập tin cấu hình sẽ chỉ thấy từng gói dữ liệu riêng rẽ trên mạng. Điều này có thể làm IDS bỏ qua một số cuộc tấn công, vì nhiều loại hình tấn công hiện đại cố tình phân mảnh dữ liệu hoặc có tình đặt phần độc hại lên một gói tin và phần còn lại lên gói tin khác (kỹ thuật lẩn trốn).  
Dữ liệu sẽ được đưa vào Preprocessors sau khi đi qua bộ giải mã gói tin (packet decoder). Suricata cung cấp một loạt các Preprocessors ví dụ như: Frag3 (một module chống phân mảnh gói tin IP), sfPortscan (module được thiết kế chống lại các cuộc trinh sát, như scan port, xác định dịch vụ, scan OS), Stream5 (module tái gộp các gói tin ở tầng TCP).

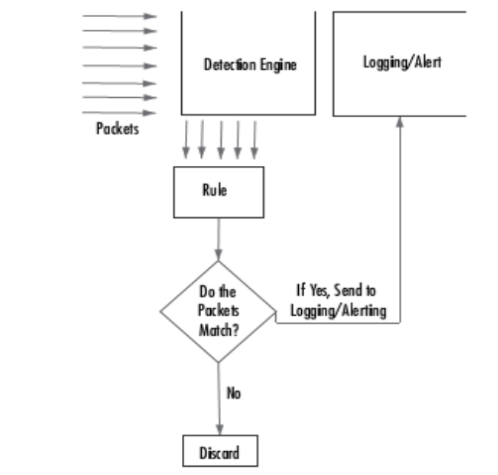


*Hình 2.11: Quá trình xử lý ở Preprocessors*

### **6.3 Detection Engine**

Đầu vào là các gói tin đã được sắp xếp ở quá trình preprocessors. Detection engine là một phần của hệ thống phát hiện xâm nhập dựa trên dấu hiệu. Detection engine sẽ lấy dữ liệu từ preprocessors và kiểm tra chúng thông qua các luật. Nếu các luật đó khớp với dữ liệu trong gói tin, nó sẽ được gửi tới hệ thống cảnh báo, nếu không nó sẽ bị bỏ qua như hình phía dưới .  
Các luật có thể được chia thành 2 phần:

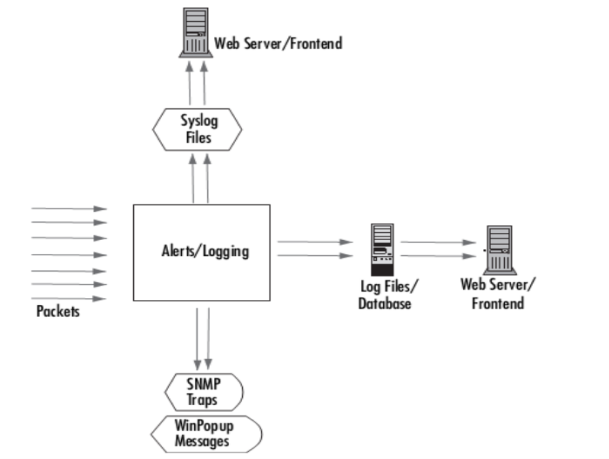
∙ ***Phần Hearder:*** gồm các hành động (log/ alert), loại giao thức (TCP, UDP, ICMP...), địa chỉ IP nguồn, địa chỉ IP đích và port.  
∙ ***Phần Options:*** là phần nội dung của gói tin được tạo ra để phù hợp với luật.  
Luật là phần quan trọng mà bất cứ ai tìm hiểu về Suricata cần phải nắm rõ. Các luật  
trong Suricata có một cú pháp cụ thể. Cú pháp này có thể liên quan đến giao thức, nội dung, chiều dài, header và một vài thông số khác. Một khi hiểu được cấu trúc các luật trong Suricata, người quản trị có thể định nghĩa các luật phù hợp với từng môi trường và hệ thống mạng khác nhau.



*Hình 2.12: Gói tin được xử lý ở Detection Engine bằng các luật*

### **6.4 Thành phần cảnh báo/logging**

Cuối cùng sau khi các luật đã phù hợp với dữ liệu, chúng sẽ được chuyển tới thành phần cảnh báo và ghi lại (alert and logging component). Cơ chế log sẽ lưu trữ các gói tin đã kích hoạt, các luật còn cơ chế cảnh báo sẽ thông báo các phân tích bị thất bại. Giống như Preprocessors, chức năng này được cấu hình trong tập tin ***suricata.yaml***, có thể chỉ định cảnh báo và ghi lại trong tập tin cấu hình nếu muốn kích hoạt.  
Dữ liệu là giá trị cảnh báo, nhưng có thể chọn nhiều cách để gửi các cảnh báo này cũng như chỉ định nơi ghi lại các gói tin. Có thể gửi cảnh báo thông qua SMB (Server Message Block) pop- up tới máy trạm Windows, ghi chúng dưới dạng logfile, gửi qua mạng thông qua UNIX socket hoặc thông qua giao thức SNMP.  
Cảnh báo cũng có thể lưu trữ dưới dạng cơ sở dữ liệu SQL như MySQL hoặc PostgreSQL. Thậm chí một vài hệ thống của các hãng thứ ba có thể gửi cảnh báo thông qua SMS tới điện thoại di động. Có rất nhiều các add-on giúp người quản trị nhận các cảnh báo cũng như phân tích các dữ liệu một cách trực quan.  
∙ **Oinkmaster:** là một Perl Script giúp cập nhật các luật của Suricata và comment nếu không muốn sau mỗi lần cập nhật.



*Hình 2.13: Thành phần cảnh báo và logging*

# Chương 3: CƠ SỞ LÝ THUYẾT VỀ ELK

1. **Một số khái niệm**

Một số khái niệm liên quan đến hệ thống phần mềm

1. ***Sự kiện (Event)***

Một sự kiện là một thay đổi quan sát được đối với hành vi của hệ thống phần mềm. Sự kiện xảy ra có thể là hành vi thông thường hoặc bất thường của hệ thống. Sự kiện bất thường của hệ thống có thể dẫn đến sự cố cho hệ thống.

1. ***Dữ liệu log sự kiện (Event log)***

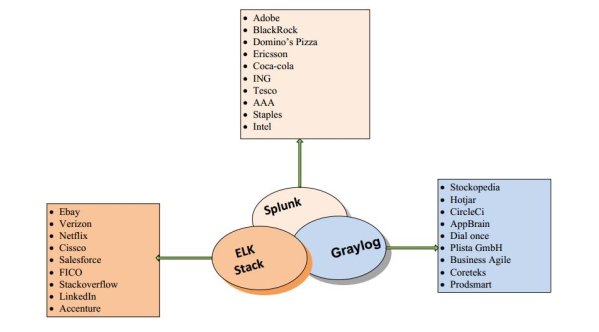
Log sự kiện là một tệp dữ liệu lưu trữ lại các sự kiện xảy ra trong hệ điều hành hoặc phần mềm. Dữ liệu log lưu trữ mọi dấu vết hoạt động hiện tại và quá khứ của hệ thống phần mềm, cho biết tình trạng hoạt động, các lỗi xảy ra và vấn đề về an ninh, bảo mật đối với hệ thống.

1. ***Sự cố (Incident)***

Một sự kiện không nằm trong các hành vi thông thường của hệ thống phần mềm, gây gián đoạn hoạt động của dịch vụ hoặc giảm chất lượng dịch vụ được gọi là sự cố.  
Dữ liệu log bản thân nó bao hàm rất rộng như dữ liệu log sự kiện (event log), dữ liệu log giao dịch (transaction log) hay dữ liệu log thông điệp (message log). Sauđây, trong phạm vi báo cáo, nhóm em quy ước rằng khi đề cập đến dữ liệu log đồng nghĩa với dữ liệu log sự kiện của hệ thống phần mềm.

1. **Lựa chọn công nghệ**

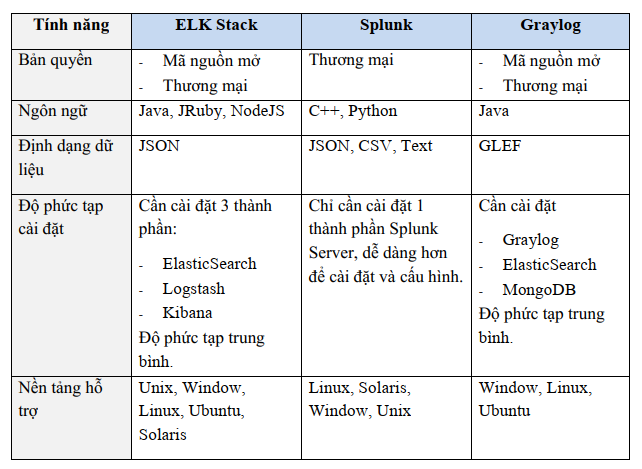
Bài toán quản lý dữ liệu log tập trung và cảnh báo lỗi tự động có thể được thực hiện bằng nhiều giải pháp công nghệ khác nhau bao gồm cả các giải pháp phần mềm thương mại và phần mềm mã nguồn mở, điển hình trong số đó phải kể đến như Splunk, Graylog và ELK. Các tính năng nổi trội của các giải pháp giám sát và phân tích dữ liệu log hệ thống ngày nay phải kể đến như: Khả năng tìm kiếm mạnh mẽ, xây dựng được màn hình giám sát thời gian thực, báo cáo, cảnh bảo ngưỡng, phân tích dữ liệu lịch sử, truy tìm vết, …  
Có thể thống kê một số tổ chức nổi bật đang sử dụng các giải pháp trên vào quản lý và phân tích dữ liệu log:

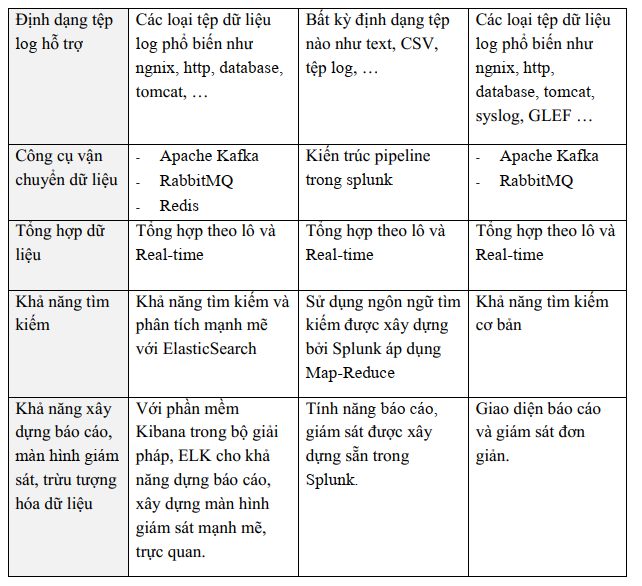


*Hình 1.1 : Một số nền tảng công nghệ được sử dụng để quản lý log*

Mỗi một giải pháp đều có những ưu, nhược điểm và mức độ phù hợp riêng.  
Dưới đây là bảng so sánh về một số tính năng của ba giải pháp trên:

*Bảng 1.1 : Bảng so sánh các tính năng của 3 pháp quản lý dữ liệu log thông dụng trên  
thị trường*

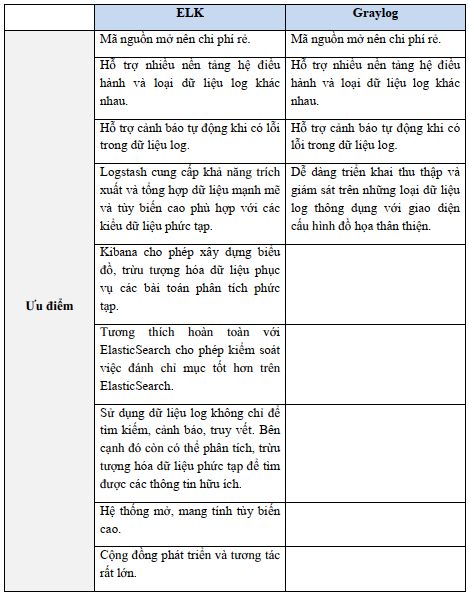


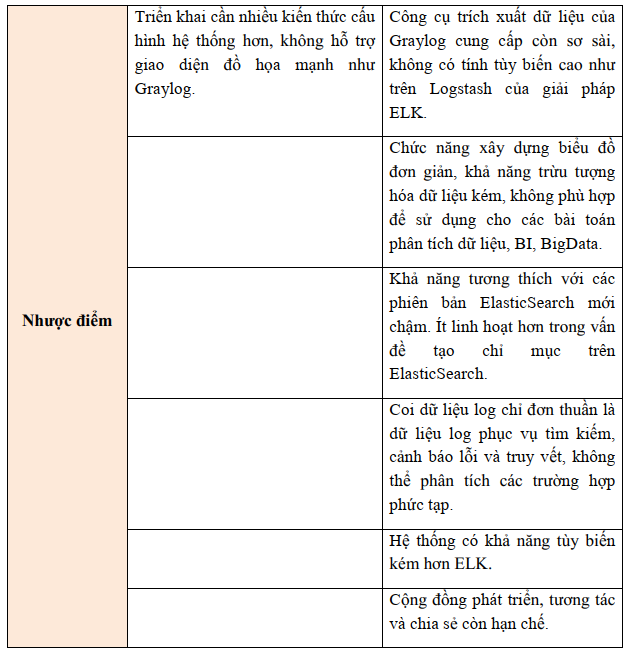
**

ELK gồm 3 thành phần:  
- ElasticSearch: Một hệ truy hồi thông tin mạnh mẽ, được sử dụng để  
lưu trữ và đánh chỉ mục cho dữ liệu log.  
- Logstash: Phần mềm mã nguồn mở thực hiện tiến trình đồng bộ dữ  
liệu ETL, thu thập dữ liệu log.  
- Kibana: Phần mềm mã nguồn mở được sử dụng để trừu tượng hóa dữ  
liệu, xây dựng biểu đồ, màn hình giám sát và phân tích dữ liệu.

ELK sẽ phát huy khả năng của mình nổi trội hơn so với các giải pháp khác khi cần trích xuất và tổng hợp dữ liệu đặc thù do Logstash cung cấp kỹ thuật trích xuất và tổng hợp dữ liệu rất mở và linh hoạt hay khi cần xây dựng các biểu đồ, trừu tượng hóa dữ liệu phức tạp để phân tích vấn đề - nơi mà Kibana sẽ thực hiện hoàn hảo nhiệm vụ của mình.

*Bảng 1.3: Bảng ưu và nhược điểm của 2 giải pháp quản lý dữ liệu log mã nguồn mở  
ELK và Graylog*

**

**

ELK hội tụ một số ưu điểm như:  
- Mã nguồn mở nên chi phí rẻ, cộng đồng phát triển lớn mạnh.  
- Nền tảng tìm kiếm ElasticSearch rất mạnh mẽ, dễ dàng mở rộng theo chiều ngang nếu hệ thống tăng trưởng lớn, tích hợp được sâu rộng với các loại công cụ, ngôn ngữ phân tích dữ liệu lớn như R, Python, Machine learning.

- Khả năng xây dựng báo cáo, phân tích và trừu tượng hóa dữ liệu mạnh mẽ với Kibana phù hợp với các bài toán phân tích dữ liệu lớn, phân tích hành vi người dùng, Business Intelligence.  
- Khả năng trích xuất và tổng hợp dữ liệu log mang tính tùy biến cao, phù hợp sử dụng đối với những trường hợp dữ liệu log phức tạp.  
- Giải pháp có tính mở và hỗ trợ tùy biến tốt.

## **3. Tìm hiểu nền tảng công nghệ ELK**

### **3.1. Giới thiệu ELK**

ELK là một bộ giải pháp công nghệ mã nguồn mở được sử dụng để thu thập, quản lý, phân tích dữ liệu log tập trung.  
ELK gồm 3 thành phần:  
- ElasticSearch: Một hệ truy hồi thông tin mạnh mẽ, được sử dụng để lưu trữ và đánh chỉ mục cho dữ liệu log.  
- Logstash: Phần mềm mã nguồn mở thực hiện tiến trình đồng bộ dữ liệu ETL, thu thập dữ liệu log, chuyển đổi, làm sạch và đưa vào lưu trữ trong ElasticSearch để đánh chỉ mục.  
- Kibana: Phần mềm mã nguồn mở được sử dụng để trừu tượng hóa dữ liệu, xây dựng biểu đồ, màn hình giám sát và phân tích dữ liệu.

### **3.2. ElasticSearch**

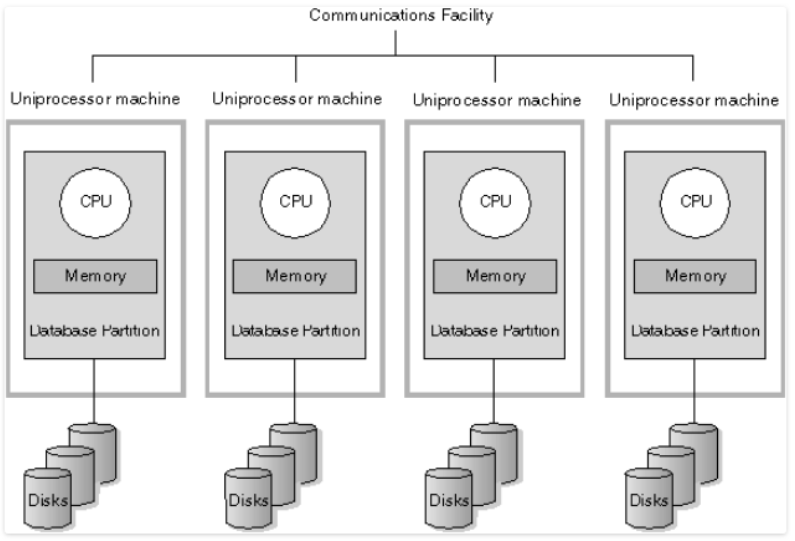
**3.2.1. Giới thiệu ElasticSearch**

ElasticSearch là một giải pháp truy hồi thông tin và phân tích dữ liệu phân tán mã nguồn mở mạnh mẽ và có tính mở rộng cao. ElasticSearch được phát triển trên nền tảng thư viện search-engine mã nguồn mở nổi tiếng “*Apache Lucene*”.  
Apache Lucene sử dụng ngôn ngữ Java và khá phức tạp để sử dụng, ElasticSearch kế thừa Apache Lucene và che dấu sự phức tạp của Lucene đằng sau các RESTful API. ElasticSearch cho phép lưu trữ, tìm kiếm, và phân tích lượng lớn dữ liệu thời gian thực. Nó thường được sử dụng để hỗ trợ cho các ứng dụng có nhu cầu tìm kiếm phức tạp, cần tốc độ nhanh và các ứng dụng phân tích dữ liệu lớn.  
Một số bài toán có thể sử dụng của ElasticSearch:  
- Tìm kiếm sản phẩm trên trang web bán hàng.  
- Thu thập log hệ thống, dữ liệu giao dịch phục vụ phân tích, tìm kiếm. Với bài toán này thì bên cạnh ElasticSearch cần sử dụng thêm 2 công cụ là Logstash và Kibana. Ba phần mềm mã nguồn mở này hợp thành bộ giải pháp ELK (ElasticSearch, Logstash và Kibana).

Một số tổ chức lớn sử dụng ElasticSearch:  
- Trang Wikipedia sử dụng ElasticSearch để cung cấp máy tìm kiếm toàn văn (full-text search) với kết quả tìm kiếm được tô sáng.  
- Trang The Guardian sử dụng ElasticSearch để kết hợp dữ liệu của người đọc với dữ liệu mạng xã hội để cung cấp các hồi đáp thời gian thực giúp tăng trải nghiệm người dùng.  
- Trang cộng đồng cho các nhà phát triển phần mềm nổi tiếng Stack Overflow sử dụng ElasticSearch làm máy tìm kiếm toàn văn kết hợp với vị trí địa lý của người dùng để đưa ra các kết quả tìm kiếm chính xác nhất cho câu truy vấn của người dùng.  
- Trang quản lý mã nguồn mở nổi tiếng GitHub sử dụng ElasticSearch để quản lý hơn 130 triệu dòng codes.

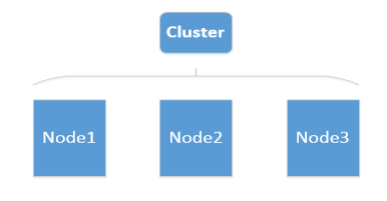
**3.2.2. Kiến trúc ElasticSearch**

**a. ElasticSearch Cluster**ElasticSearch Cluster được xây dựng theo ý tưởng kiến trúc MPP (Massive Parallel Processing). Massive Parallel Processing là một hệ thống gồm nhiều nút (node), hoạt động cùng nhau để cùng thực hiện một chương trình, trong đó mỗi node sẽ xử lý một phần riêng của chương trình trên chính tài nguyên của node đó (memory, CPU, …). Do đó mà một hệ thống MPP còn được gọi là hệ thống “Shared nothing” (vì bản chất các node trong cụm không chia sẻ tài nguyên gì để tính toán, chúng xử lý dữ liệu riêng của chúng trên sức mạnh tài nguyên riêng của chúng). Để có thể xử lý lượng dữ liệu khổng lồ, dữ liệu trong giải pháp MPP thường được phân chia giữa các node thành các phân đoạn (shard), mỗi nút sẽ xử lý dữ liệu cục bộ của nó. Điều này càng tăng tốc độ xử lý dữ liệu, bởi vì sử dụng lưu trữ chia sẻ cho giải pháp MPP sẽ là một khoản đầu tư lớn hơn, phức tạp hơn, tốn kém hơn, ít khả năng mở rộng hơn, sử dụng lưu lượng mạng cao hơn và ít tính toán song song hơn.  
Với cách thiết kế này, một hệ thống MPP sẽ rất dễ dàng để mở rộng, ta chỉ cần thêm node vào cụm cluster theo chiều ngang là có thể mở rộng năng lực tính toán cho toàn cụm. Mô hình MPP như sau:

**

*Hình 1.2: Mô hình Massive Parallel Processing*

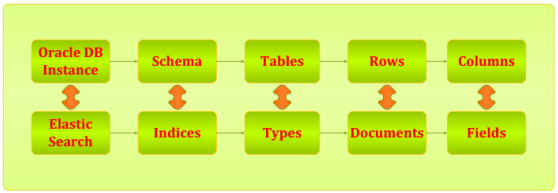
Một Cụm ElasticSearch Cluster bao gồm một hoặc nhiều nút (nodes) có cùng tên Cluster mà nó tham gia vào. Các nodes trong cụm Cluster làm việc cùng nhau và chia sẻ dữ liệu và tải (workload) với nhau. Khi một node được thêm vào hoặc rời khỏi cụm thì Cluster tự động tổ chức và tính toán lại dữ liệu và năng lực tính toán.

**

*Hình 1.3: Mô hình cụm Cluster của ElasticSearch*

Một node là một máy chủ riêng lẻ, là một phần của cụm Cluster, tham gia vào quá trình đánh chỉ mục và tìm kiếm của cụm Cluster. Cũng giống như Cluster, mỗi node được định danh bởi một tên duy nhất và được sinh ngẫu nhiên tại thời điểm khởi động hệ thống. Tất nhiên chúng ta có thể chỉ định tên cho các node này cho mục đích quản lý. Mỗi node có thể tham gia (join) vào một cluster mặc định là “elasticsearch” Cluster nếu không được chỉ định. Mỗi cụm ElasticSearch Cluster có một Master node chịu trách nhiệm quản lý các thay đổi trong toàn cụm như tạo, xóa index hoặc thêm hay loại bỏ node vào/ra cụm Cluster. Master node sẽ không tham gia vào quá trình xử lý tìm kiếm. Bất kỳ node nào trong cụm cũng có thể trở thành master node. Với cụm Cluster chỉ có 1 node thì node đó sẽ thực hiện cả vai trò là master node và node xử lý tìm kiếm. Tất cả các nodes trong cụm Cluster được kết nối, chia sẻ dữ liệu và tải với nhau. Tất cả các nodes đều biết chính xác dữ liệu được lưu trữ ở đâu, do đó khi có một yêu cầu cần xử lý được gửi đến chúng có thể trực tiếp xử lý hoặc chuyển tiếp yêu cầu đến node mà đang thực sự chứa dữ liệu cần xử lý và trả ra kết quả.

**b. Các khái niệm cơ bản trong ElasticSearch**ElasticSearch được cộng đồng mã nguồn mở phát triển đã trải qua rất nhiều phiên bản, phiên bản chính thức mới nhất tại thời điểm chúng tôi nghiên cứu là phiên bản 6.6. Trong phiên bản này có sự thay đổi so với các phiên bản 5.x trước đó, đó là chỉ có một kiểu mapping type duy nhất trong một ElasticSearch Index. Để hiểu rõ hơn chúng ta sẽ nghiên cứu các thành phần trong ElasticSearch. Các thành phần trong ElasticSearch khi so sánh với một cơ sở dữ liệu quan hệ (Ví dụ cơ sở dữ liệu Oracle) như sau:

**

*Hình 1.4 : So sánh các thành phần của ElasticSearch với Cơ sở dữ liệu quan hệ*

Trong đó: Chỉ mục (Index) trong ElasticSearch được coi tương đương với Lược đồ (Schema), Kiểu (Type) tương đương với khái niệm bảng (Table), Tài liệu (Document) tương đương với bản ghi (Row) và Trường (Field) tương đương với Cột (Column) trong cơ sở dữ liệu quan hệ.

⮚ **Index**ElasticSearch sử dụng chỉ mục ngược để đánh chỉ mục cho các tài liệu. Một chỉ mục trong ElasticSearch là một khái niệm logic, nó bao gồm tập hợp các tài liệu có một số đặc điểm tương tự nhau. Ví dụ: một chỉ mục cho dữ liệu khách hàng, một chỉ mục khác cho danh mục sản phẩm và một chỉ mục khác cho dữ liệu đơn hàng.  
Một chỉ mục được xác định bằng một tên duy nhất (phải là chữ thường) và tên này được sử dụng khi thực hiện các hoạt động như lập chỉ mục, tìm kiếm, cập nhật và xóa đối với các tài liệu trong chỉ mục đó.  
⮚ **Type**Type đại diện cho kiểu của tài liệu hay thực thể được đánh chỉ mục. Một loại (Type) là một danh mục / phân vùng logic của chỉ mục để cho phép lưu trữ các loại tài liệu khác nhau trong cùng một chỉ mục, ví dụ: Index có tên là twitter có một loại cho người dùng (user tyle), một loại khác cho bài viết trên blog (tweet type).  
⮚ **Document**Tài liệu (document) là một đơn vị thông tin cơ bản có thể đánh chỉ mục. Document giống như row của table trong cơ sở dữ liệu quan hệ. Thuật ngữ “document” trong ElasticSearch chỉ đến các tài liệu được thể hiện dưới dạng JSON. Hầu hết các objects và documents đều có thể được thể hiện dưới dạng JSON document với key và value. Một key là tên của một field (hay property), và một value có thể là một kiểu String, Boolean, Integer, … Ví dụ một document được thể hiện dưới dạng JSON document:

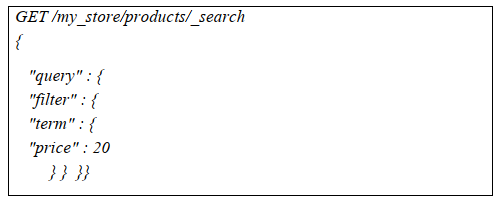
**

Một tài liệu không chỉ chứa dữ liệu, nó còn có siêu dữ liệu (metadata) – thông tin về tài liệu đó, bao gồm:

- Index: Nơi mà tài liệu được đánh chỉ mục và lưu trữ (Tên index)  
- Type: Lớp/Kiểu mà document thể hiện. (VD: kiểu “doc”, kiểu “blog” hay “comment”, …)  
- Id: Định danh duy nhất cho tài liệu để phân biệt giữa tài liệu này với tài liệu khác.  
⮚ **Field**Khái niệm Field để chỉ một trường của tài liệu JSON. Field trong tài liệu JSON được biểu diễn dưới dạng <Key, Value>, với Key là tên của Field và Value là giá trị của Field đó, Value có thể là kiểu String, Boolean, Integer, …  
⮚ **Shard**Khi 1 chỉ mục quá lớn, không thể lưu trữ trên 1 node thì ElasticSearch cho phép chia chỉ mục đó ra thành các phân đoạn (shards). Index chỉ có thể được chia thành các shards khi cụm Cluster có 2 nodes trở lên. Việc chia Index thành các shards có các lợi ích sau:  
- Hệ thống có thể mở rộng theo chiều ngang.  
- Cho phép tìm kiếm song song (parallel) trên các shards.  
ElasticSearch khuyến nghị phải tạo 1 hoặc nhiều bản sao cho mỗi shard của index, bản sao này gọi là replica shard. Bản shard gốc gọi là primary/original shard. Việc tạo nhiều replica shard cho phép tìm kiếm song song, tăng hiệu năng tìm kiếm. Số Primary Shards và số Replica Shards có thể thiết đặt khi tạo index. Sau khi Index được tạo thì số Replica Shards có thể thay đổi được, nhưng số Primary Shards là không thể thay đổi. Mặc định, nếu cụm Cluster có từ 2 nodes trở lên mà khi tạo index không có chỉ định rõ số Primary Shards và số Replica Shards thì ElasticSearch mặc định số Primary Shards là 5 và mỗi Primary Shard có 1 Replica Shard. Mỗi Shard trong ElasticSearch là một Lucene Index. Trong 1 Lucene Index thì số lượng documents tối đa có thể chứa được là 2^31. Khi cụm cluster được mở rộng (thêm node) hoặc co lại (loại bỏ node) thì cluster sẽ tự động tính toán và di chuyển các shards qua lại các nodes để cụm Cluster đạt trạng thái cân bằng (balanced).

**3.2.4. Tìm kiếm trong ElasticSearch**Có 2 dạng tìm kiếm trong ElasticSearch là lọc (**Filter**) và truy vấn (**Query**). Sự khác nhau giữa 2 dạng tìm kiếm này là Query sẽ tính toán độ liên quan và xếp hạng kết quả tìm kiếm; trong khi đó filter sẽ trả ra kết quả chính xác như điều kiện tìm kiếm và không tính toán, xếp hạng kết quả.

**a. Filter**Khi muốn tìm kiếm chính xác các tài liệu chứa một giá trị nào đó ta sẽ sử dụng câu lệnh lọc (filter). Bởi vì Filter không tính toán độ liên quan và xếp hạng kết quả tìm kiếm nên tốc độ của filter là rất nhanh. Và để tăng tốc hơn nữa cho các câu lệnh filter, ElasticSearch hỗ trợ lưu vào bộ nhớ đệm (Cache) các kết quả tìm kiếm của câu lệnh Filter phục vụ cho các lần tìm kiếm sau. Ví dụ câu lệnh Filter trong ElasticSearch:



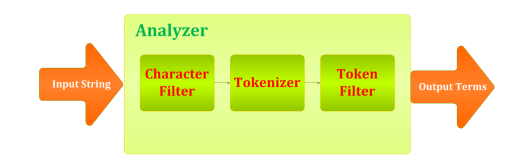
Lệnh filter trên tương đương với câu lệnh truy vấn SQL trong cơ sở dữ liệu quan hệ sau:

Kết quả trả ra cho câu tìm kiếm chính xác (filter) tùy thuộc vào kiểu dữ liệu của từ tố và kiểu của trường trong chỉ mục. Đối với câu filter cho kiểu dữ liệu number, bool hoặc date sẽ luôn cho ra kết quả chính xác với điều kiện tìm kiếm. Tuy nhiên với kiểu dữ liệu String (chuỗi ký tự) thì kết quả tìm kiếm phục thuộc vào cách đánh chỉ mục cho trường dữ liệu này. Trong ElasticSearch có 2 kiểu đánh chỉ mục cho trường dữ liệu String là kiểu Keyword và kiểu Text. Trong đó với kiểu Keyword ElasticSearch sẽ không sử dụng tiến trình phân tích từ tố (analysis) trong quá trình đánh chỉ mục, ngược lại đối với kiểu Text, ElasticSearch sẽ sử dụng tiến trình analysis khi đánh chỉ mục. Ví dụ tìm kiếm lọc (filter) giá trị chính xác cho dữ liệu kiểu chuỗi ký tự:

Câu truy vấn trên tương đương với câu SQL sau trong cơ sở dữ liệu quan hệ:

⮚ **Tiến trình phân tích (Analysis)**Tiến trình phân tích từ tố có nhiệm vụ phân tách trường dữ liệu chuỗi thành các từ tố được chuẩn hóa để phục vụ việc đánh chỉ mục và tìm kiếm toàn văn. Tiến trình phân tích được thực hiện bởi Analyzer, có nhiều loại analyzer trong ElasticSearch, mặc định là “English Analyzer”.

Analyzer gồm 3 module: Character filters, tokenizers, và token filters



*Hình 1.6: Tiến trình phân tích từ tố (Analysis) trong ElasticSearch*

Trong đó:  
- **Character filter**: nhận chuỗi ký tự gốc đầu vào, sau đó có thể thêm, xóa, thay đổi các ký tự cho phù hợp trước khi đưa vào module “tokenizers”, một analyzer có thể không có hoặc có nhiều Character filters.  
- **Tokenizer**: nhận chuỗi các ký tự đầu vào, chia chuỗi ký tự thành các từ tố riêng lẻ (thường là các từ “word” riêng lẻ), và cho đầu ra là một chuỗi các từ tố. Ví dụ: “Whitespace tokenizer” sẽ thực hiện phân tách các từ tố dựa trên khoảng trắng. Tokenizer cũng làm nhiệm vụ đánh dấu thứ tự, vị trí của từng từ tố (được sử dụng cho các truy vấn từ và cụm từ). Một Analyzer bắt buộc phải có một Tokenizer.  
- **Token filter**: nhận một luồng các từ tố (đầu ra từ Tokenizer), và nó có thể thêm, xóa, sửa các từ tố cho phù hợp để được đầu ra là các từ tố được chuẩn hóa phục vụ quá trình đánh chỉ mục ngược. Ví dụ: “lowercase Token filter” sẽ chuyển toàn bộ các từ tố sang dạng viết thường (lowercase), hoặc một “stop token filter” sẽ loại bỏ toàn bộ các từ dừng (stop words). Một Analyzer có thể không có hoặc có nhiều token filter, nếu có nhiều token filter, chúng sẽ được thực hiện theo thứ tự.

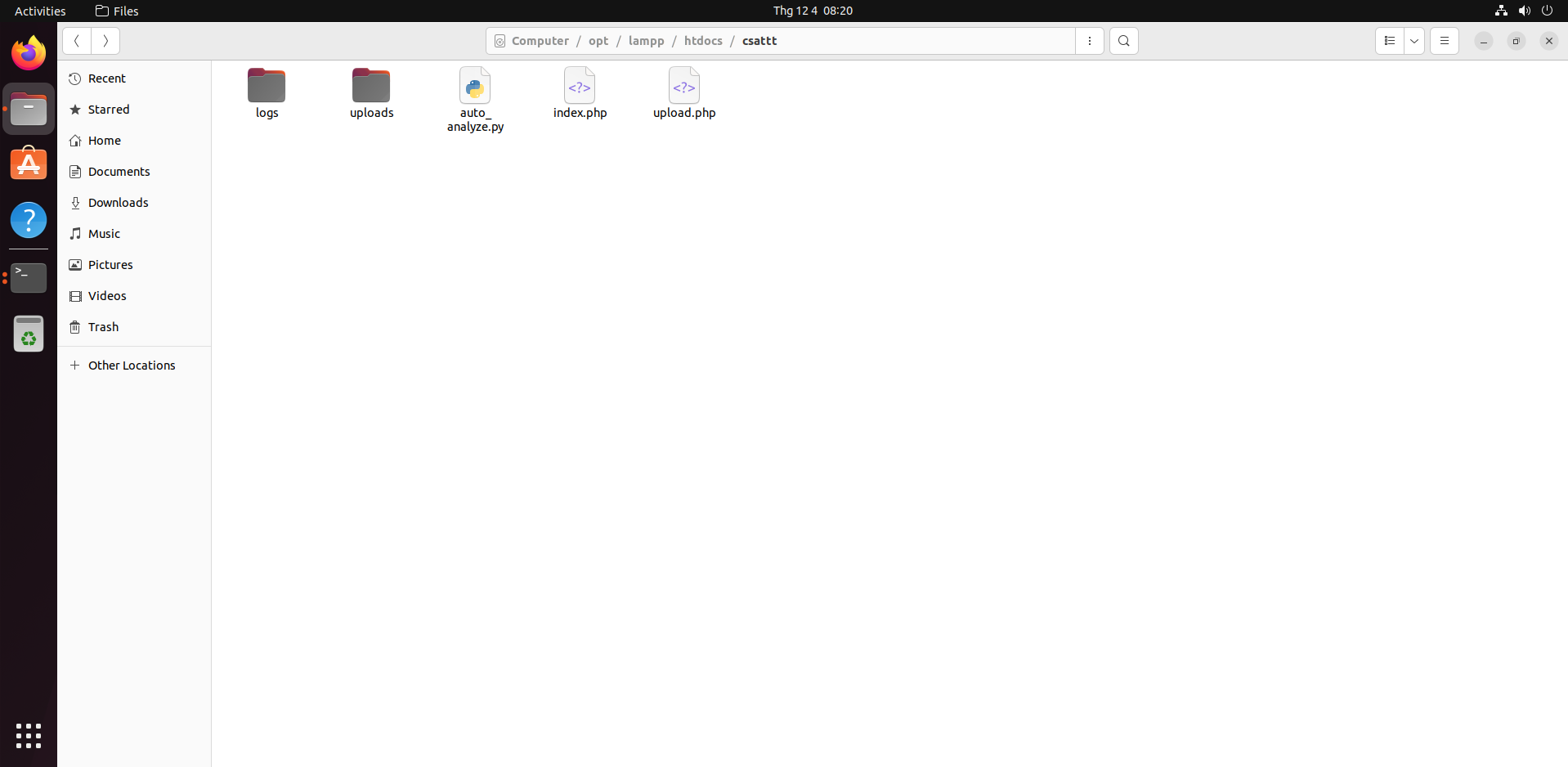
# 

# Chương 4: Thực nghiệm

## **Chủ đề 1: Xây dựng một website để phân tích các gói pcap và xem log.**

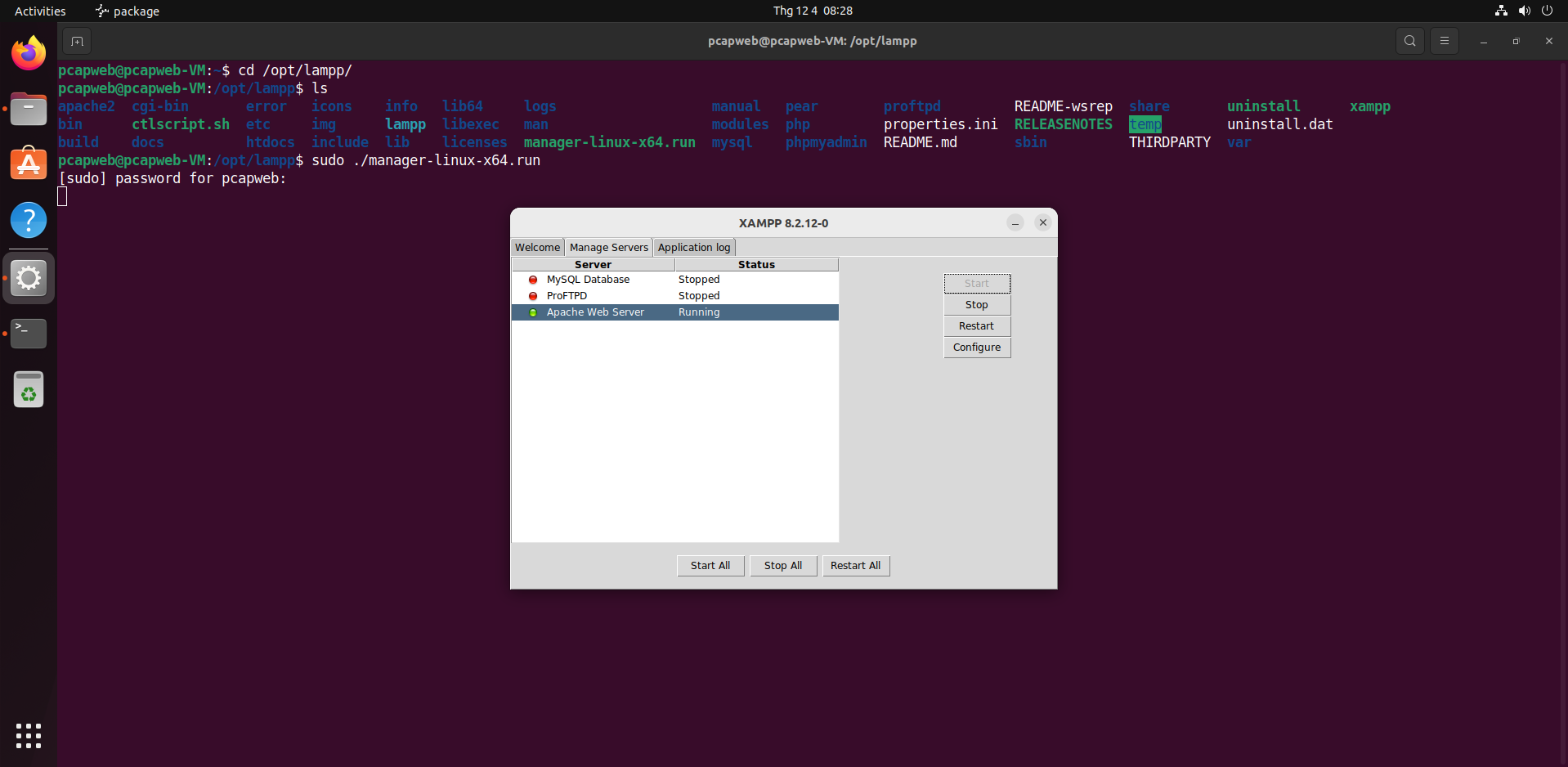
1. **Chuẩn bị**

* Phần mềm Suricata
* XAMPP Apache để chạy Web
* Máy chủ FTP với VSFTPD để gửi file Pcap
* Script python tự động phân tích file Pcap được gửi đến
* 1 trang web cho phép user tải file Pcap lên

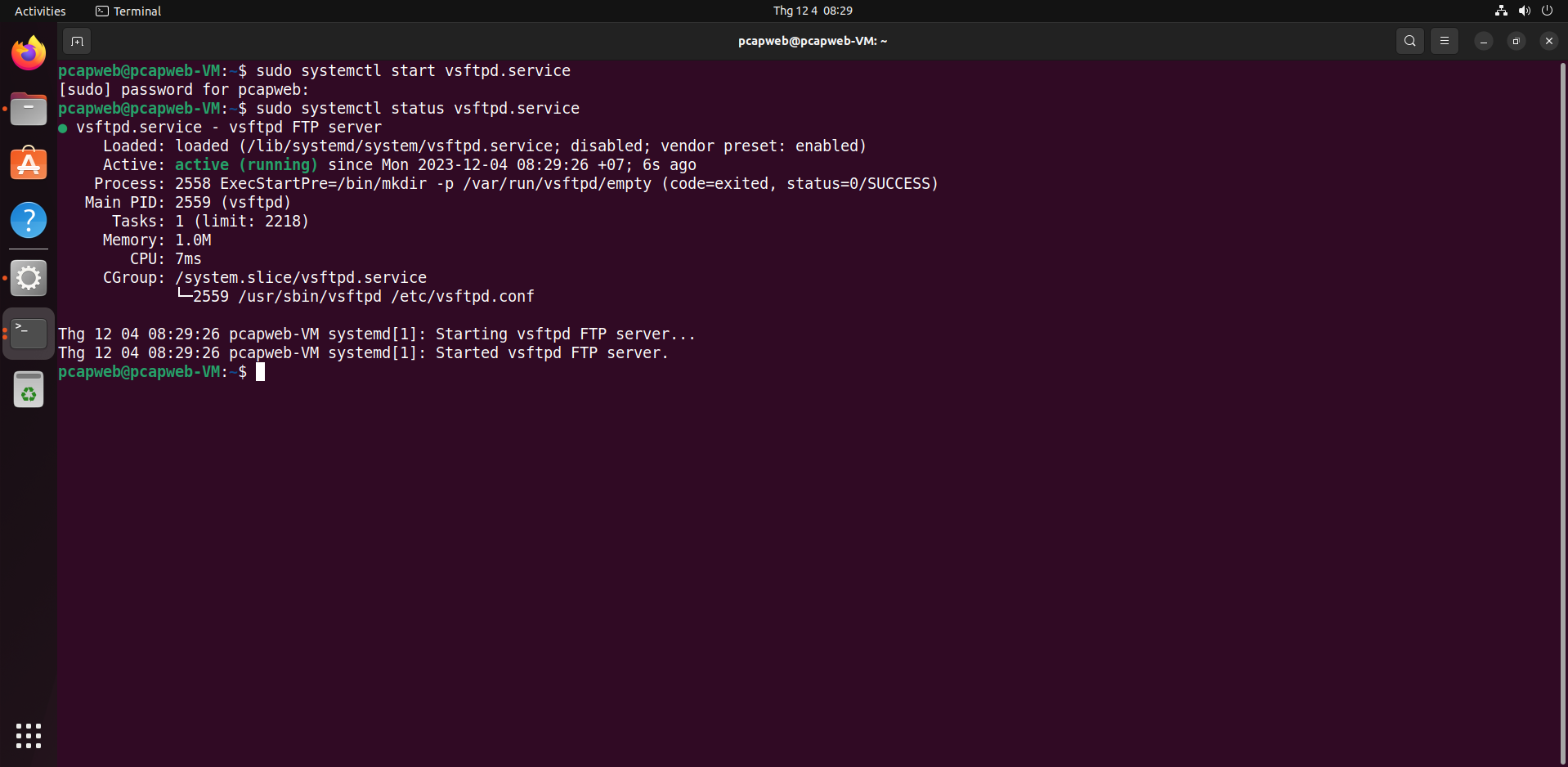


1. **Triển khai**

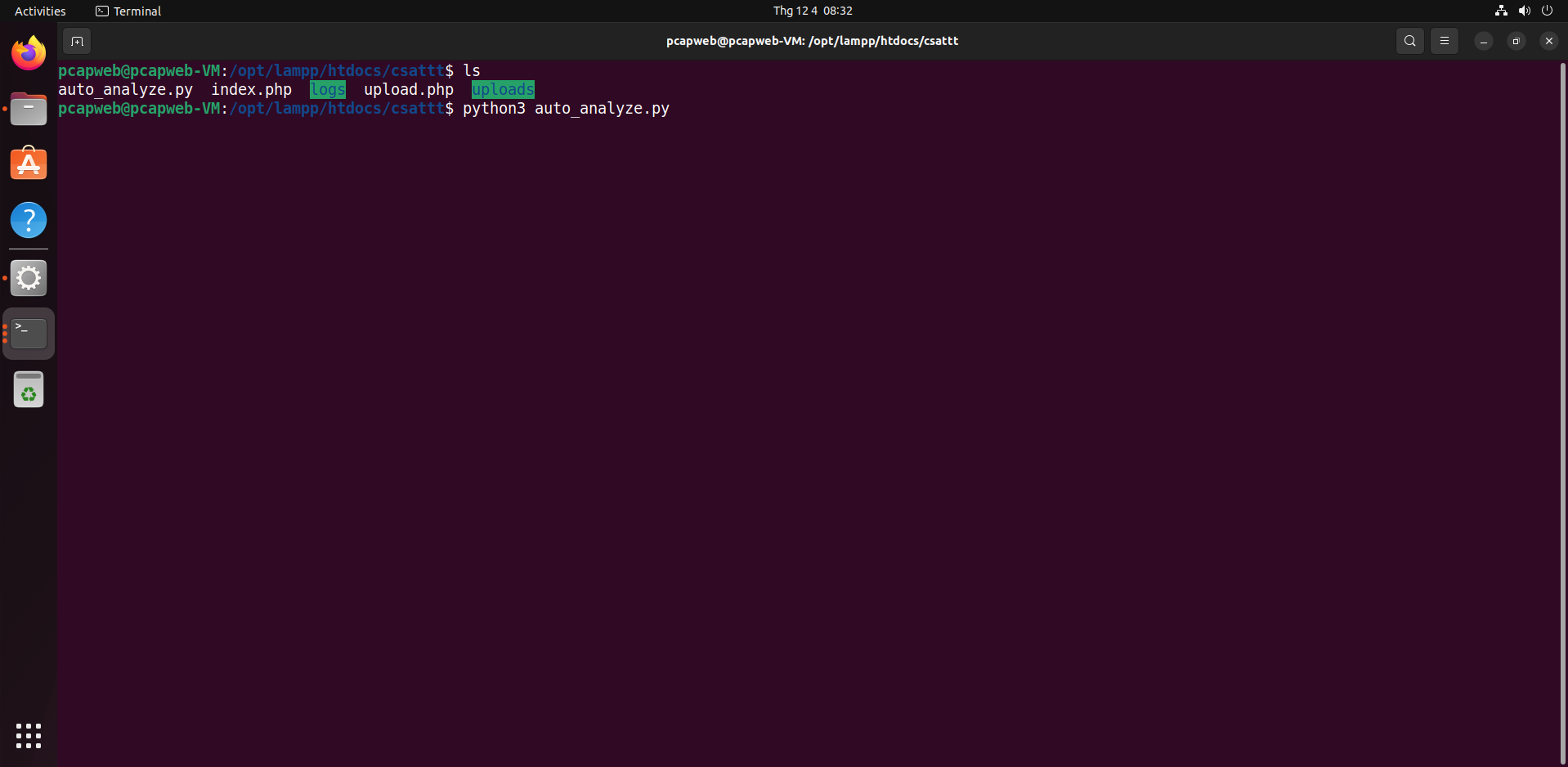
* Khởi chạy Apache



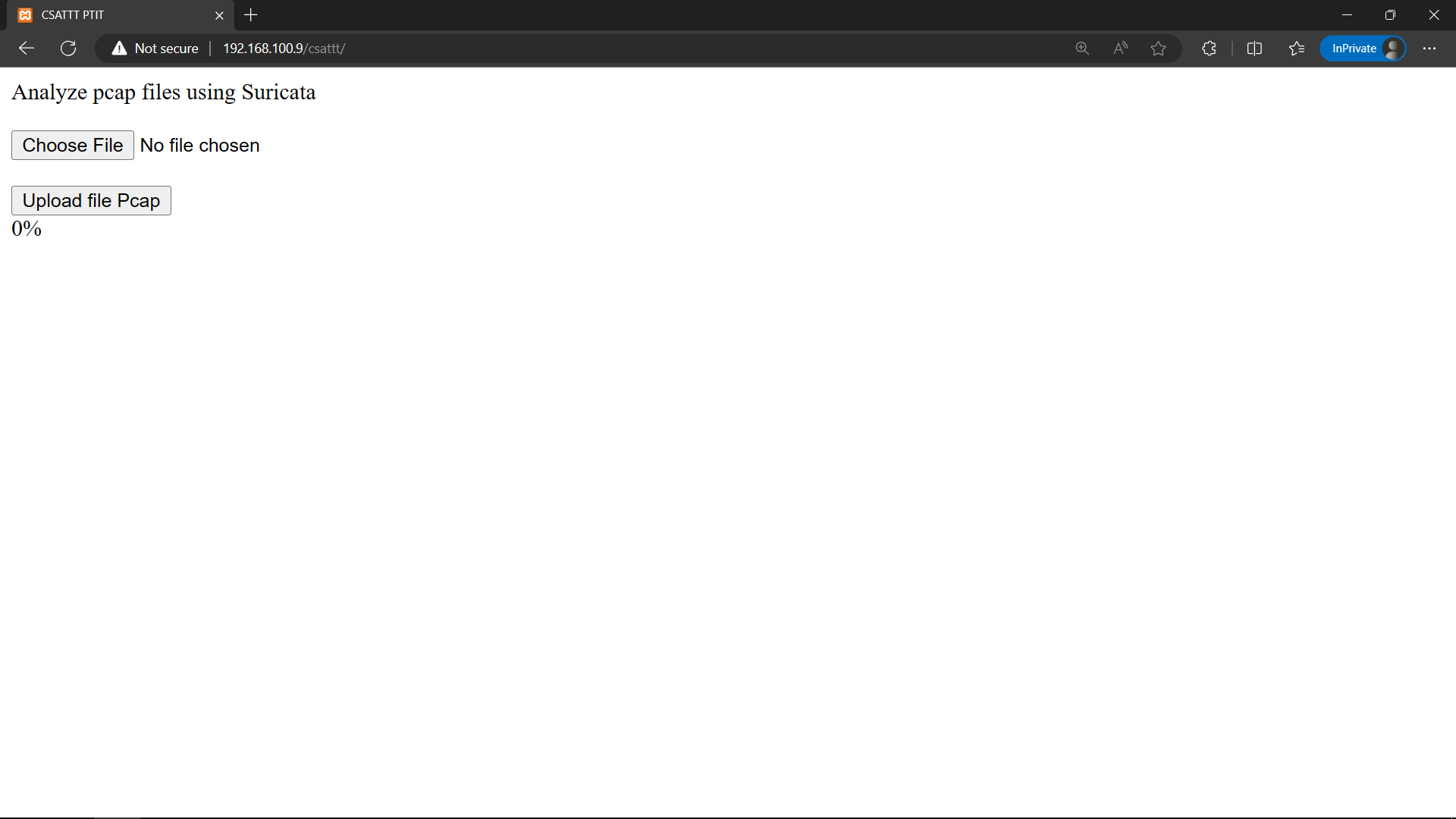
* Chạy dịch vụ vsftpd



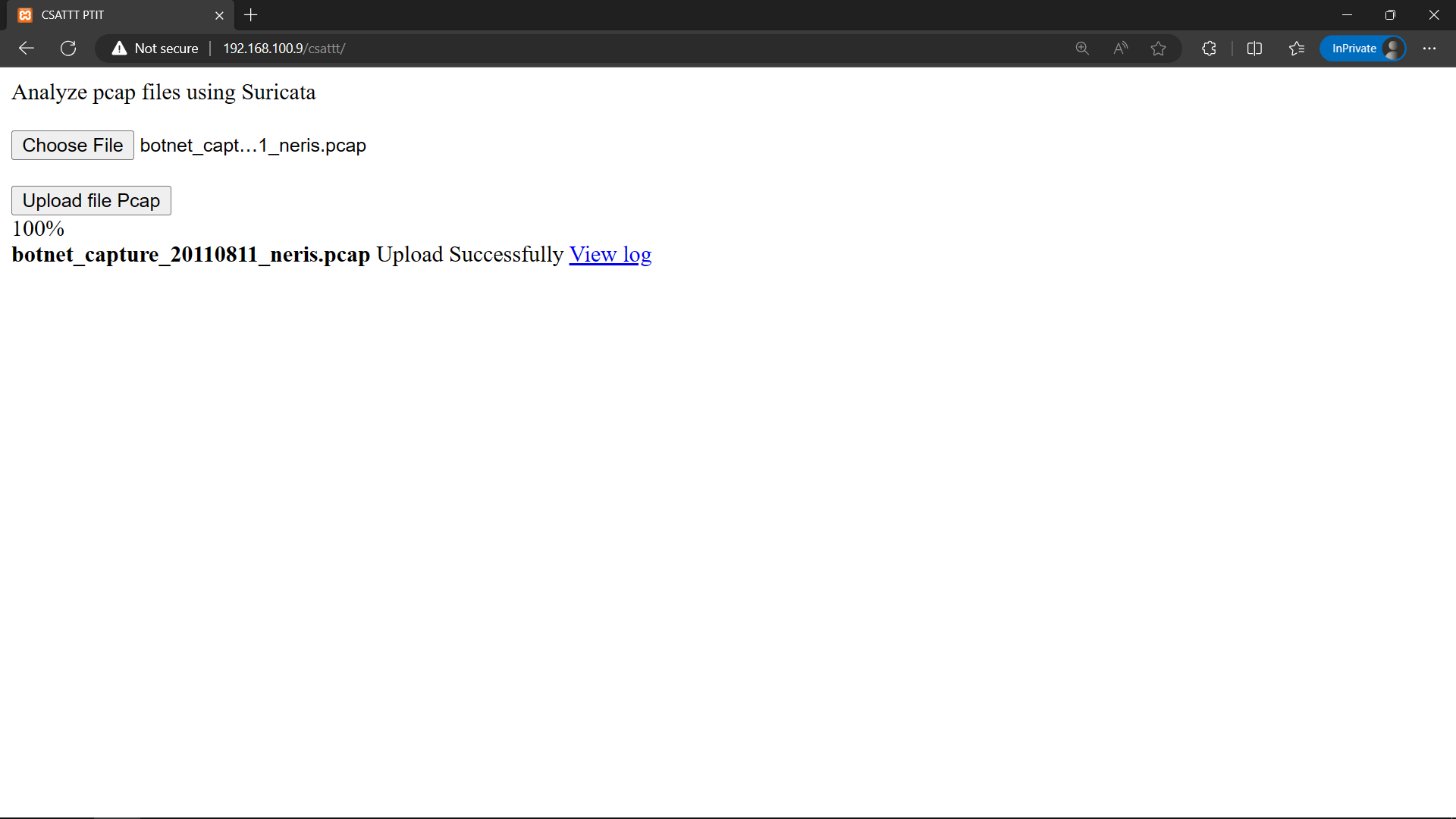
* Chạy file python tự động phân tích



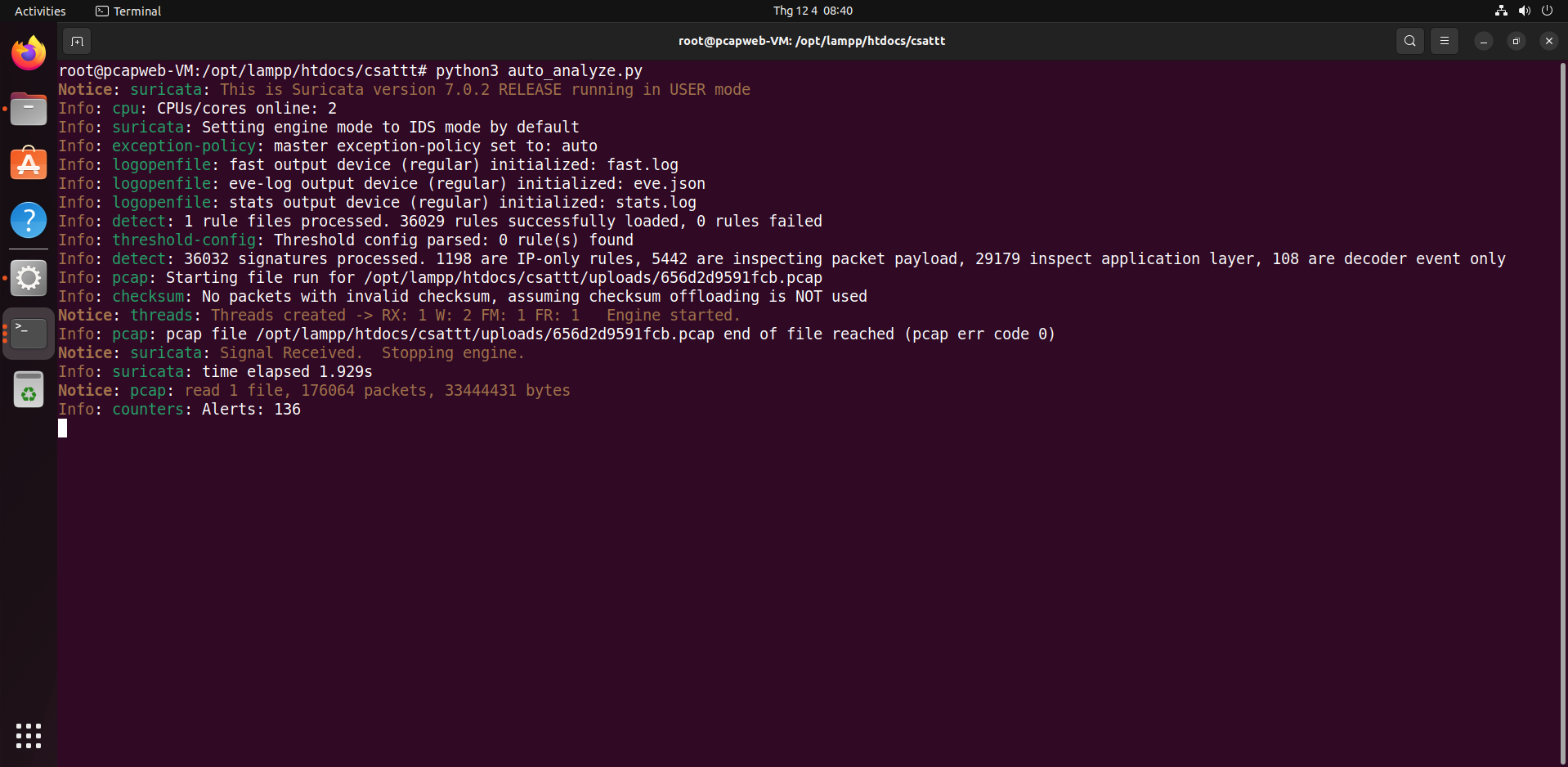
* Vào trang web



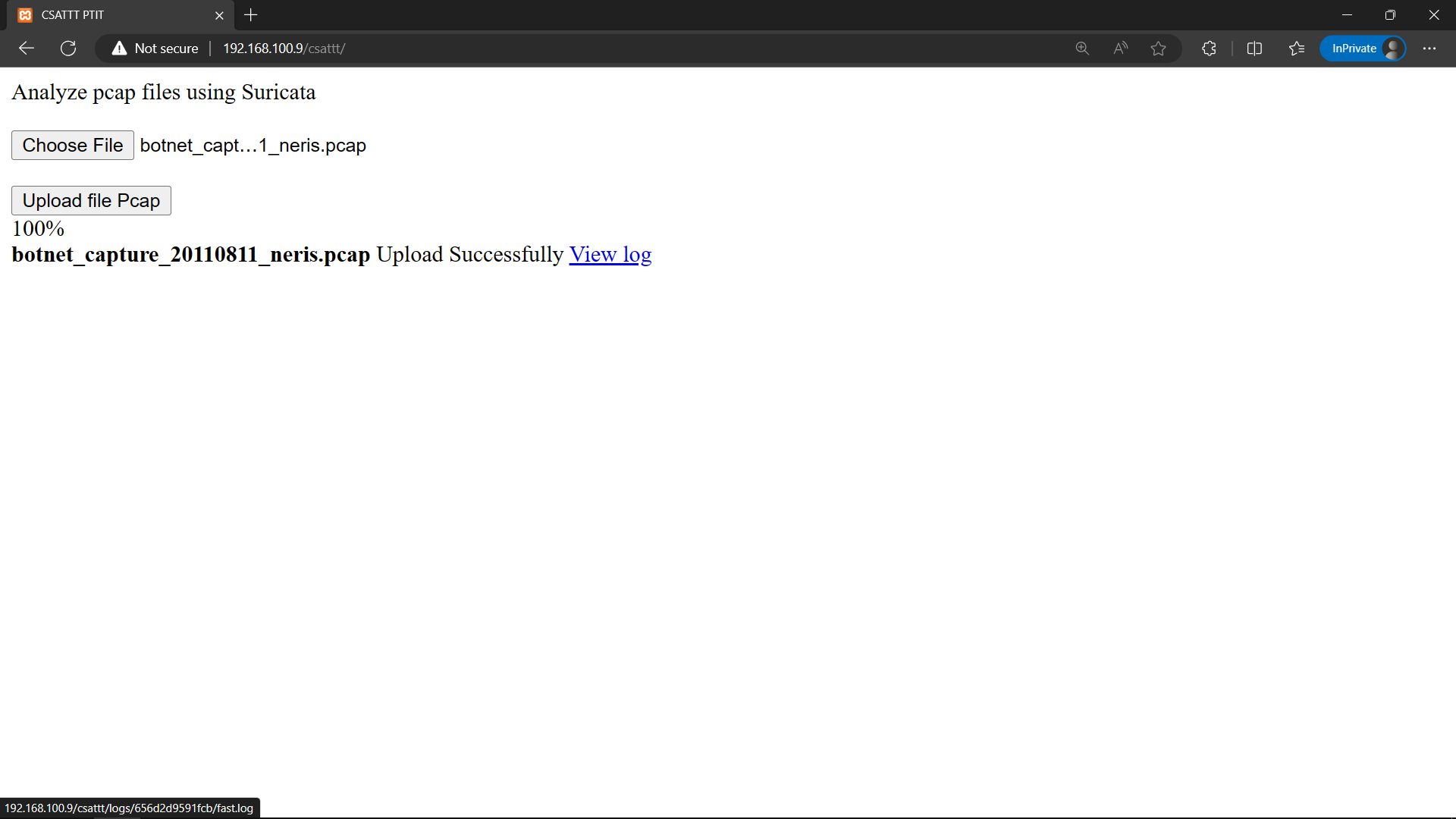
* Tiến hành upload file Pcap

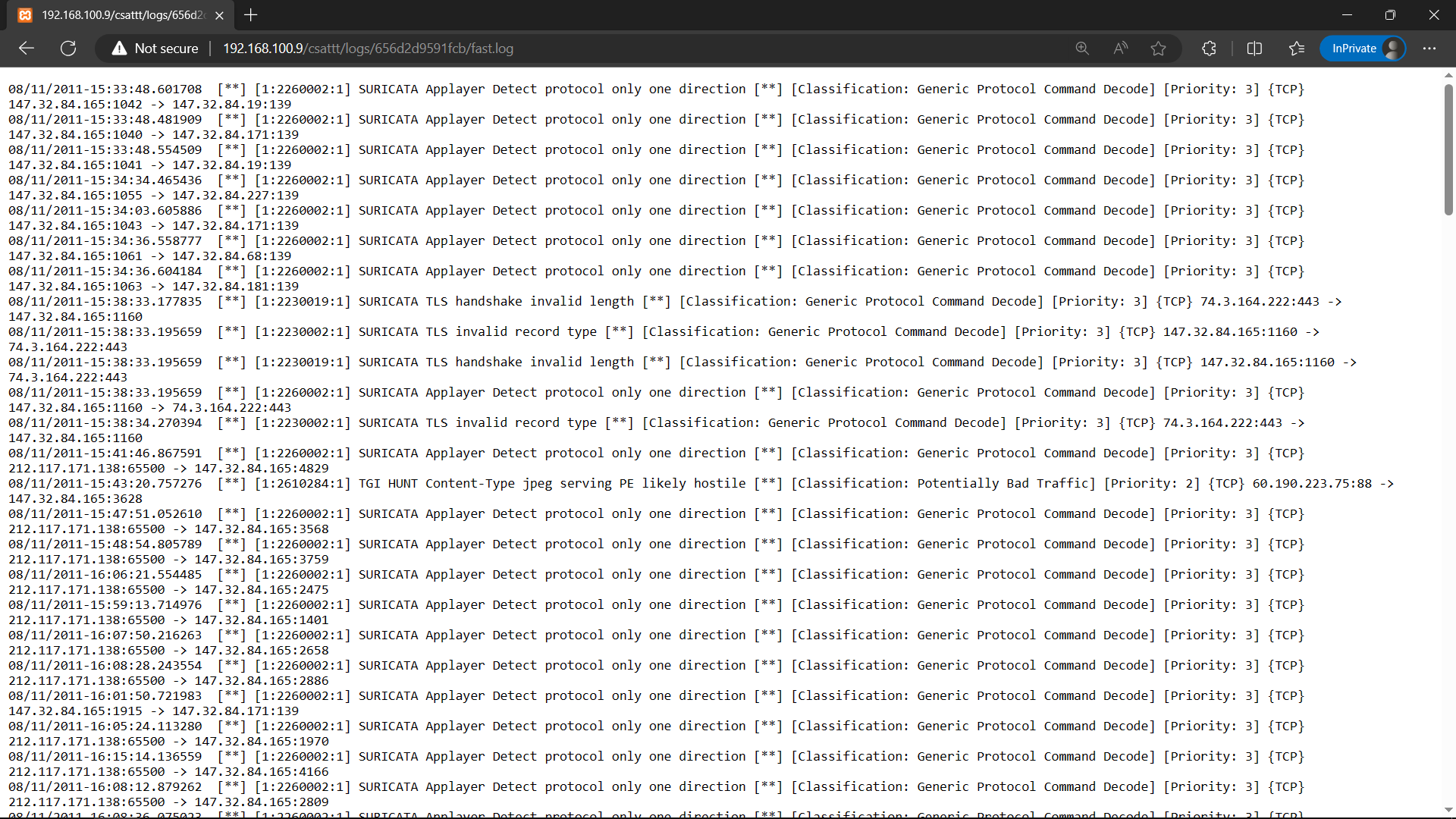


* Ngay khi file Pcap được tải lên thành công, Suricata sẽ tự động phân tích và đưa ra file log



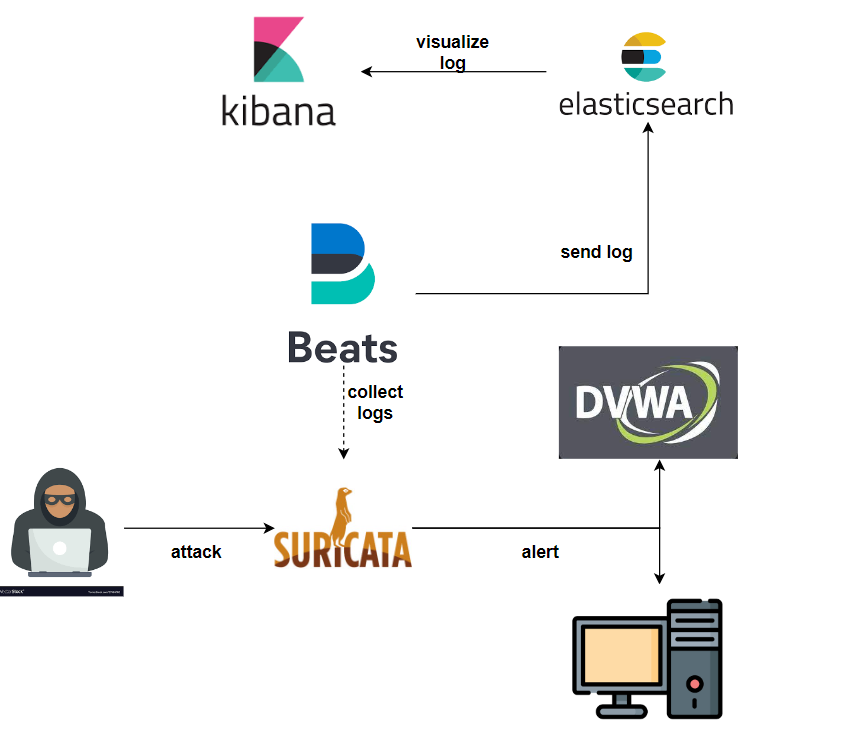
* Click vào View log trên trang web để xem log cảnh báo





## **Chủ đề 2: Thực hiện quản lý, phát hiện tấn công các máy trong mạng LAN**

1. **Mô hình mạng**

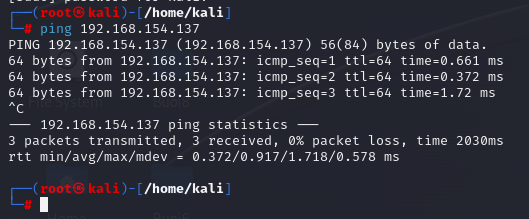


1. **Thực hiện tấn công, phát hiện trên suricata xem log và xem log trên elasticsearch**

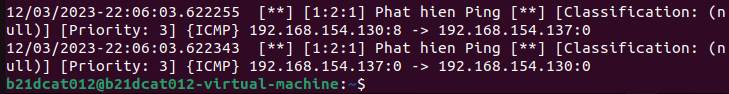
* Phát động tấn công trên máy attack và xem suricata có phát hiện không:

Máy attack (ip: 192.168.154.130), Máy victim (ip: 192.168.154.137)

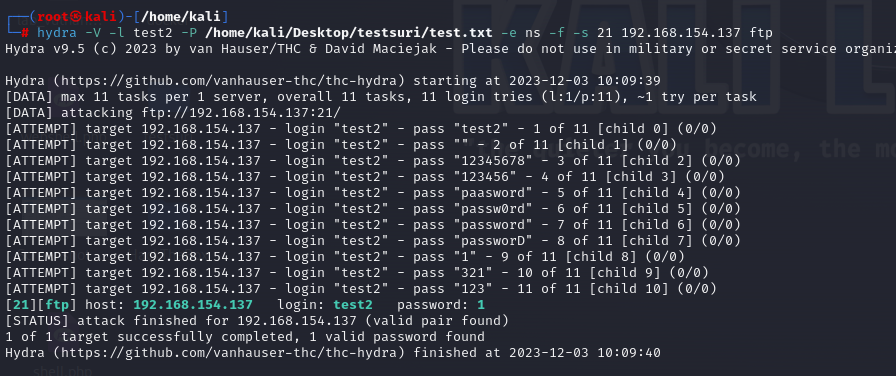
* Thực hiện ping từ máy attack đến victim:



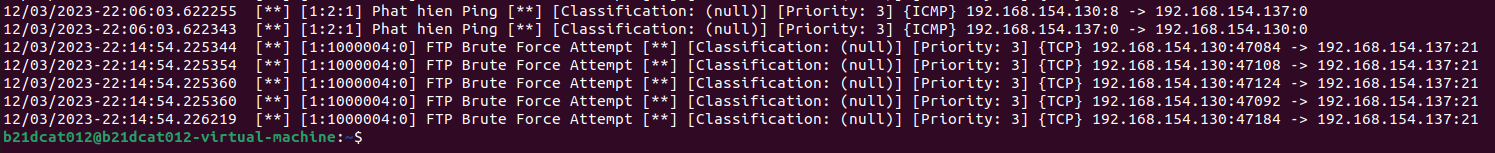
Ta thấy phía suricata đã phát hiện:



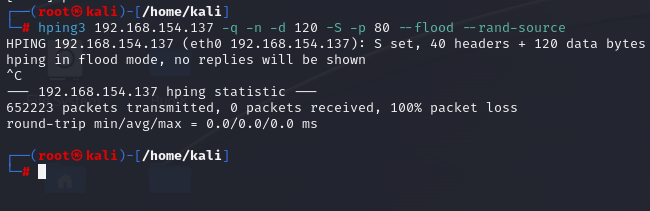
* Thực hiện Brute force từ máy attack đến victim với test.txt là một danh sách các password đã chuẩn bị sẵn và phần user là test2:



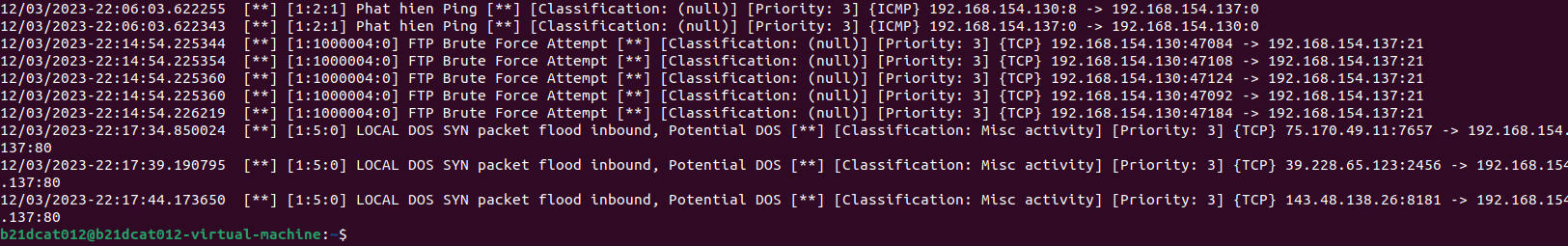
Ta thấy phía Suricata cũng đã phát hiện:



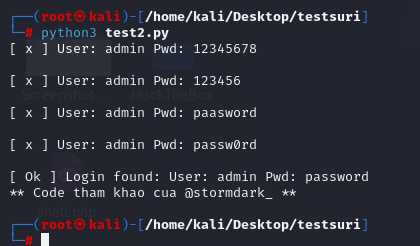
* Thực hiện DOS từ máy attack đến victim:



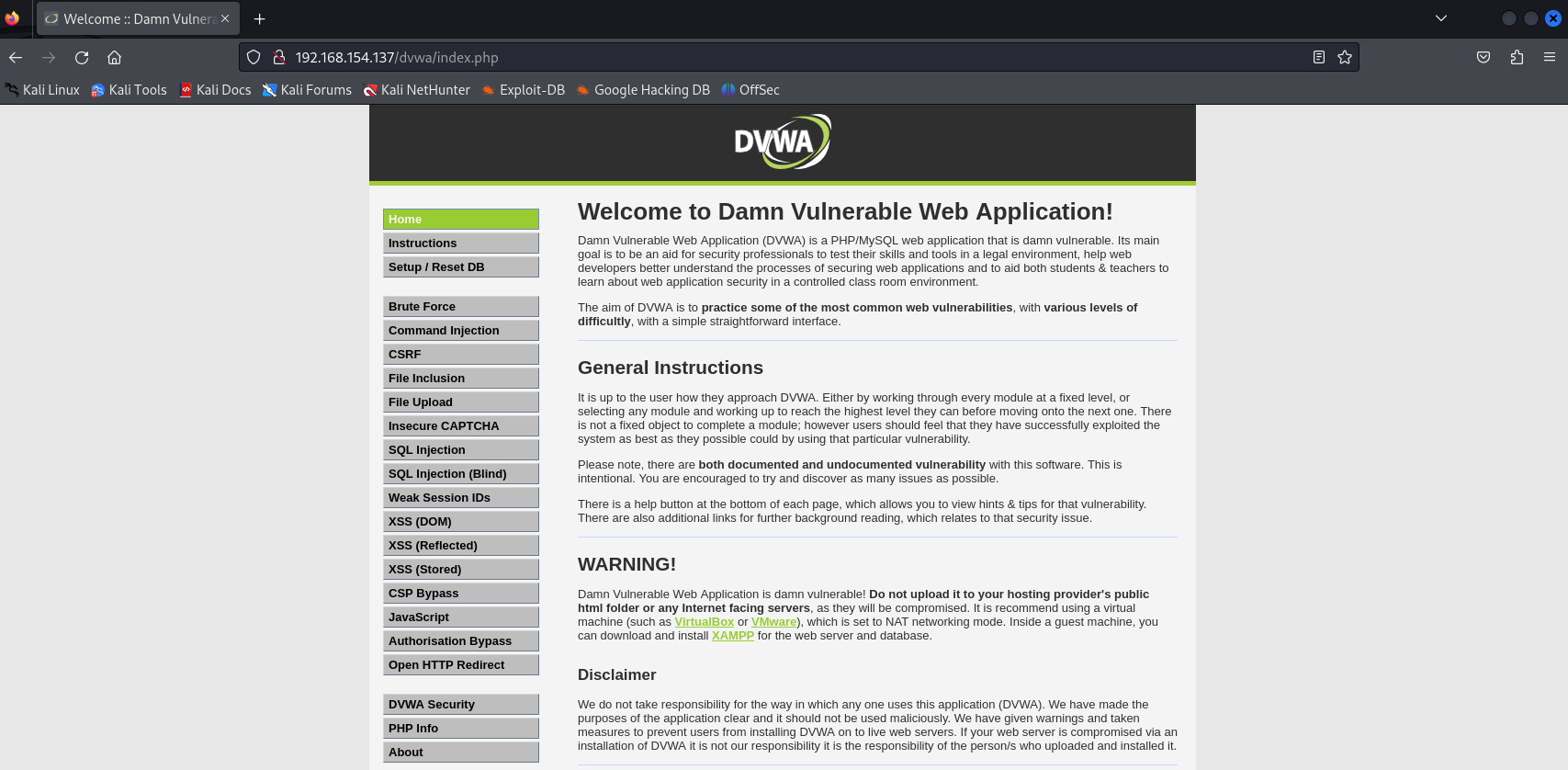
Ta thấy Suricata cũng đã phát hiện.



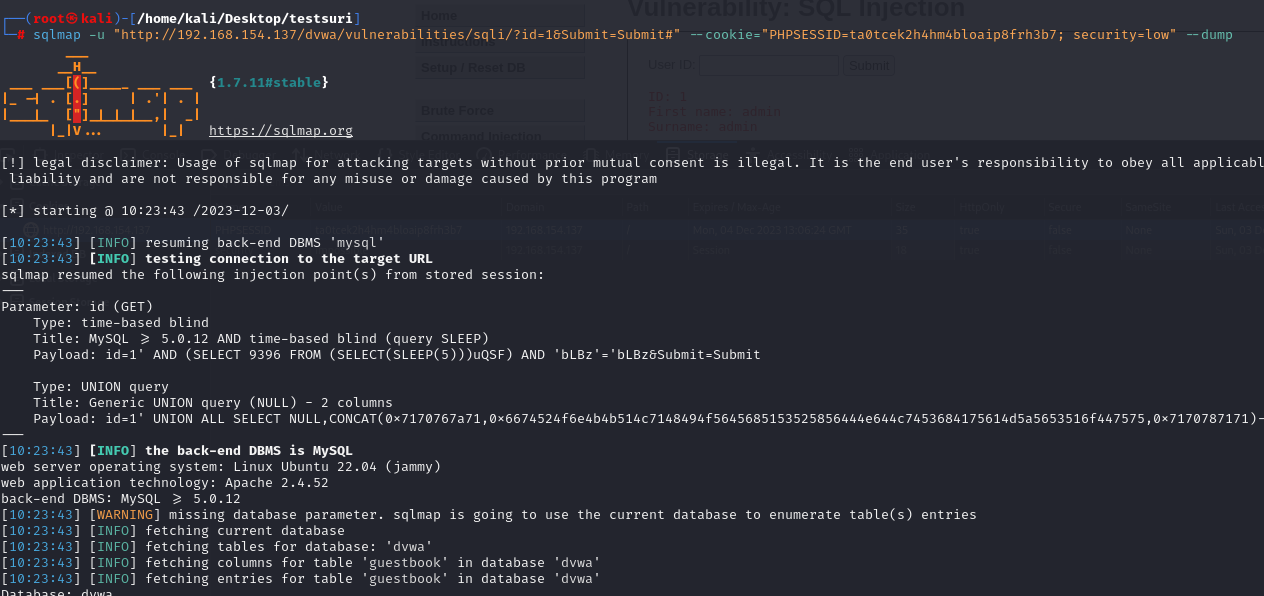
* Trên máy victim có chạy một web dvwa từ phía attack thực hiện Brute force lấy account sau đó vào web thực hiện sql injection:

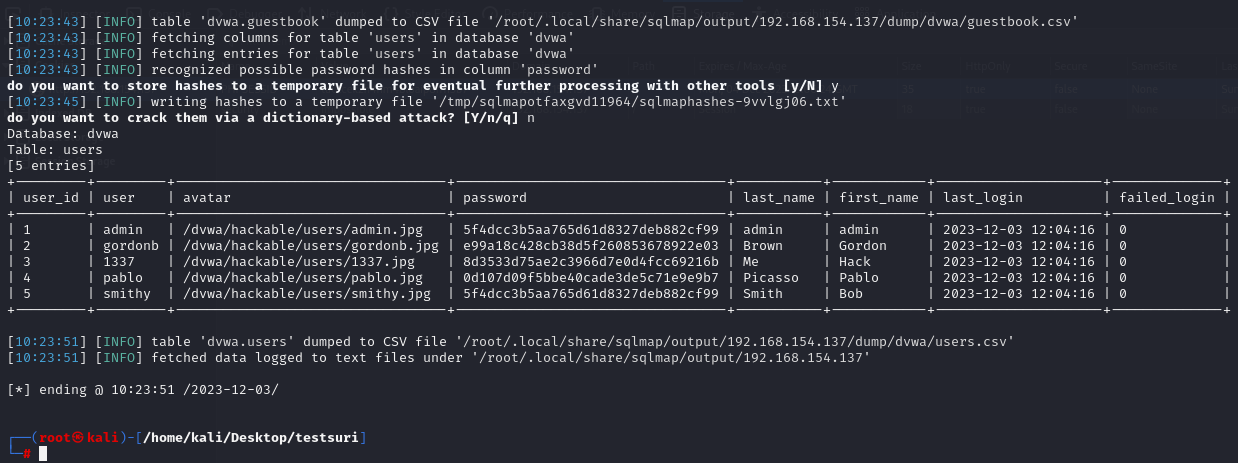


Giao diện khi đăng nhập được vào web:

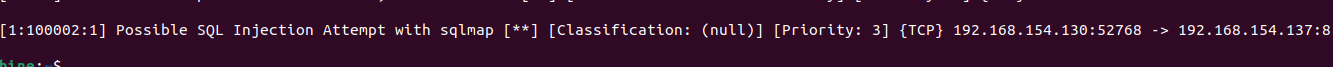


Tiếp tục phía attack ta vào SQL injection, thực hiện tấn công ta lấy cookie thực hiện lệnh như trong ảnh và xâm nhập được vào database:

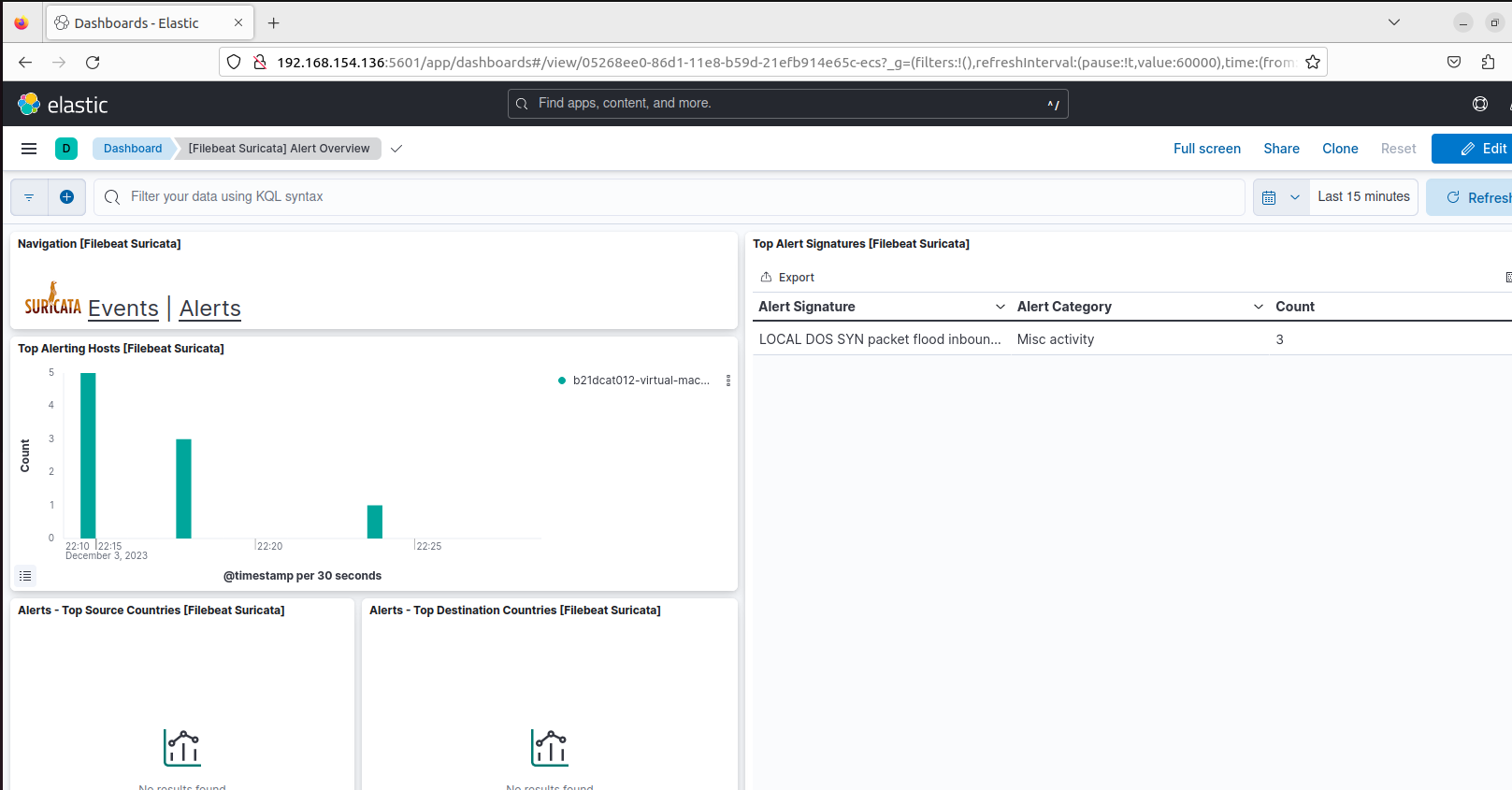




Ta vào phía Suricata thì thấy đã phát hiện được:



* Sau một vài cuộc tấn công ta vào Elastic xem log trong phần Alert thì thấy các sự kiện đều đã được ghi lại:



# TÀI LIỆU THAM KHẢO

[1] http://www.openinfosecfoundation.org/index.php/download-suricata  
[2] http://taosecurity.blogspot.com/2014/01/suricata-20beta2-as-ips-on-ubuntu-  
1204.html  
[3] http://www.linux.org/threads/suricata-the-snort-replacer-part-3-rules.4363/  
[4] http://suricata.readthedocs.io/en/latest/index.html  
[5] https://github.com/OISF/suricata  
[6] *"New Open Source Intrusion Detector Suricata Released"*. Slashdot. 2009-  
12-31. Retrieved 2011-11-08.  
[7] *"Suricata Downloads"*. Open Security Information Foundation.  
Retrieved 2011-11-08