ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Văn Dũng

PHÁT TRIỂN HỆ THỐNG QUẢN LÝ VÀ ĐIỀU KHIỂN CHO CÔNG CỤ PHÁT HIỆN VÀ XỬ LÝ XÂM NHẬP SURICATA

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY
Ngành: Truyền thông và mạng máy tính

Hà Nội – 2020

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Văn Dũng

PHÁT TRIỂN HỆ THỐNG QUẢN LÝ VÀ ĐIỀU KHIỂN CHO CÔNG CỤ PHÁT HIỆN VÀ XỬ LÝ XÂM NHẬP SURICATA

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY
Ngành: Truyền thông và mạng máy tính

Cán bộ hướng dẫn: TS. Phạm Mạnh Linh

Cán bộ đồng hướng dẫn: ThS. Đặng Văn Đô

Hà Nội – 2020

TÓM TẮT

Với sự phát triển nhanh chóng của internet như hiện nay, rất nhiều các doanh nghiệp, cơ quan tổ chức đã và đang xây dựng những hệ thống mạng nhằm phục cho các hoạt động sản xuất kinh doanh, trao đổi thông tin của mình. Sự phát triển của mạng internet đi kèm với sự ra đời của các công nghệ và kỹ thuật giúp đảm bảo an toàn thông tin trên không gian mạng.

Một trong những công nghệ nổi bật thường được các tổ chức sử dụng đó là IPS/IDS điển hình là công cụ Suricata. Việc sử dụng công cụ này đã trở nên phổ biến dẫn đến những vấn đề trong quá trình quản trị khi một quản trị viên phải thực hiện việc quản lý nhiều công cụ được tích hợp ở các thiết bị khác nhau.

Vì vậy em quyết định lựa chọn đề tài "PHÁT TRIỂN HỆ THỐNG QUẢN LÝ VÀ ĐIỀU KHIỂN CHO CÔNG CỤ PHÁT HIỆN VÀ XỬ LÝ XÂM NHẬP SURICATA". Với mục tiêu nghiên cứu và phát triển một hệ thống giúp ích cho việc quản lý công cụ này.

LỜI CAM ĐOAN

Tôi xin cam đoan rằng mọi kết quả trình bày trong khóa luận đều do tôi thực hiện dưới sự hướng dẫn của TS. Phạm Mạnh Linh và đồng hướng dẫn là ThS. Đặng Văn Đô. Tất cả các tham khảo nghiên cứu liên quan đều được nêu rõ nguồn gốc một cách rõ ràng từ danh mục tài liệu tham khảo trong khóa luận. Khoá luận không sao chép lại từ tổ chức hoặc cá nhân nào khác mà không chỉ rõ về mặt tài liệu tham khảo.

Các thống kê, các kết quả trình bày trong khóa luận đều được lấy từ thực nghiệm khi chạy chương trình. Nếu sai tôi xin hoàn toàn chịu trách nhiệm theo quy định của trường Đại học Công Nghệ - Đại học Quốc gia Hà Nội.

.

Hà Nội, ngày tháng năm

Sinh viên

Nguyễn Văn Dũng

LỜI CẢM ƠN

Đầu tiên, em xin gửi lời cảm ơn chân thành tới TS. Phạm Mạnh Linh và ThS. Đặng Văn Đô đã cho em cơ hội được học tập và nghiên cứu, đã nhiệt tình giúp đỡ em trực tiếp giải quyết các vướng mắc kỹ thuật khi thực hiện khóa luận.

Sự hướng dẫn, hỗ trợ và động viên của các Thầy đã giúp em trong việc nghiên cứu và hoàn thành đồ án này. Em xin cảm ơn các thầy, cô trong bộ môn Mạng và Truyền thông máy tính, các thầy cô giảng dạy tại trường Đại học Công nghệ đã giúp đỡ em trong suốt quá trình học tập và nghiên cứu.

Bên cạnh đó, em xin cảm ơn gia đình và bạn bè tại trường Đại học Công nghệ đã đồng hành cùng em suốt hơn bốn năm qua.

Em xin chân thành cảm ơn!

Hà Nội, ngày tháng năm

Sinh viên

Nguyễn Văn Dũng

MỤC LỤC

Bảng các ký hiệu và từ viết tắt

IDS	Intrusion Detection Systems
IPS	Intrusion Prevention Systems
NIDS	Network – Based Intrusion Detection Systems
HIDS	Host- Based Intrusion Detection Systems
NIPS	Network-Based Intrusion Prevention System
HIPS	Host-Based Intrusion Prevention System
NSM	Network Security Monitoring
UTM	Unified Threat Management
AMQP	Advanced Message Queue Protocol
DLQ	Dead letter Queue

CHƯƠNG 1. GIỚI THIỆU BÀI TOÁN

Theo Wearesocial [25] tính đến năm 2019 trên thế giới hiện có khoảng 4.4 tỷ người dùng internet chiếm hơn 57% dân số thế giới, trong đó ngoài người dùng cá nhân ra thì có không ít là các tổ chức doanh nghiệp. Đây là những đối tượng thường sử dụng Internet phục vụ cho các hoạt động kinh doanh và nghiệp vụ của mình với lượng thông tin lớn và quan trọng cần được bảo mật. Phần lớn các tổ chức doanh nghiệp này đều có các hệ thống mạng riêng của mình kết nối với Internet, đây chính là những mục tiêu cho những kẻ tấn công mạng nhằm đánh cắp các thông tin cũng như gây ra các thiệt hại cho các đối tượng bị tấn công.

Theo thống kê vào năm 2018 của Purplesec [26] có 80000 cuộc tấn công an ninh mạng mỗi ngày hay 30 triệu cuộc tấn công mỗi năm, trong đó phần lớn là các cuộc tấn công nhằm vào các cơ quan tổ chức đặc biệt là các công ty nhỏ (Khoảng 61% cuộc tấn công nhằm vào các công ty có quy mô dưới 1000 nhân viên theo Verizon [27]) nơi việc đảm bảo an toàn an ninh mạng vẫn chưa được thực sự quan tâm. Rất nhiều các công ty không xây dựng biện pháp bảo vệ an toàn thông tin một cách bài bản.

Với thực trạng như vậy việc áp dụng các giải pháp an ninh mạng đang dần trở thành nhu cầu cấp thiết, các giải pháp có thể được sử dụng như : firewall, mạng VLAN ảo, sao lưu và mã hóa quyền truy cập dữ liệu v.v.v, một trong số những giải pháp được sử dụng phổ biến đó là tích hợp các công cụ IPS/IDS vào hệ thống mạng, đây là một giải pháp với chi phí thấp dễ dàng cho việc tích hợp và quản trị phù hợp với các tổ chức vừa và nhỏ.

IDS/IPS sử dụng các công nghệ để phân tích các luồng lưu lượng đến các tài nguyên cần được bảo vệ như hệ thống mạng nội bộ hay miền DMZ từ đó phát hiện ra các hoạt động xâm nhập và khai thác lỗ hổng bảo mật của kẻ tấn công đồng thời tiến hành các biện pháp ngăn chặn cũng như thông báo tới đến quản trị viên.

Việc tích hợp các công cụ IDS/IPS đã trở thành một biện pháp đảm bảo an ninh mạng quan trọng, có rất nhiều các công cụ IDS/IPS có mặt trên thị trường với những ưu nhược điểm khác nhau có thể kể đến như Zeek, Snort, Sagan, Suricata… Trong đó Suricata là một công cụ được sử dụng khá phổ biến.

Suricata có thể được triển khai như một Host based IDS/IPS để theo dõi lưu lượng trên một máy tính các nhân hoặc như một Network based IDS /IPS giám sát tất cả các lưu lượng qua mạng.

Việc có thể được triển khai với nhiều vai trò cho phép Suricata có thể được cài đặt ở nhiều vị trí trong một hệ thống mạng để phục vụ các mục đích khác nhau.

Tuy nhiên giống như các IDS/IPS thường thấy khác khi các quản trị viên muốn thực hiện các thao tác điều khiển với các công cụ này thì cần phải làm việc trực tiếp trên thiết bị được cài đặt. Việc này sẽ rất đơn giản nếu hệ thống mạng chỉ có một thiết bị cài đặt công cụ Suricata, trong trường hợp có nhiều hơn một thiết bị sử dụng công cụ này thì việc quản lý các công cụ này dần trở nên khó khăn nhất là khi các thiết bị này bị phân tán về mặt địa lý.

Một bài toán được đặt ra đó là làm cách nào để việc quản lý các công cụ Suricata được triển khai trên các thiết bị khác nhau trở nên tập trung hơn hay không.

Trên thực tế đã có một số giải pháp được đưa nhằm giải quyết vấn đề quản lý các công IDS/IPS như suricata, trong số đó có thể kể Owlh một dự án mã nguồn mở cung cấp hê thống hỗ trợ quản các IDS/IPS. Tuy nhiên không phải mọi trường hợp sử dụng Owlh đều là phương pháp tối ưu nhất, đặc biệt đối với các hệ thống nhỏ và yêu cầu các thao tác điều khiển đơn giản với Suricata được cài đặt như các Host- Based IDS

Trong luận văn này sẽ tập trung vào việc nghiên cứu và phát triển một hệ thống đơn giản giúp giải quyết bài toán trên để việc quản lý và điều khiển các công cụ Suricata đã được tích hợp với số lượng lớn trở nên dễ dàng hơn. Hệ thống sẽ hướng tới việc cho phép các quản trị viên có thể cùng lúc quản lý và thực hiện việc điều khiển đối với các công cụ Suricata đã được cài đặt, có thể trên cùng một hệ thống mạng hoặc nhiều hệ thống mạng khác nhau.

Hệ thống sẽ gồm 3 phần chính:

Suricata: một IDS được cài đặt trên các thiết bị trong hệ thống mạng và thực hiện nhiệm vụ phát hiện và ngăn chặn xâm nhập.

Node : Đây sẽ là một chương trình được tích hợp trên cùng một thiết bị với công cụ Suricata và thực hiện các thao tác điều khiển đối với công cụ đó. Các node sẽ kết nối và được quản lý bởi một web server

Website: Nơi các quản trị viên có thể thực hiện các thao tác điều khiển tới các node một cách trực quan.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Hệ thống IDS

IDS (Intrusion Detection System) [1] – là hệ thống giám sát lưu lượng mạng nhằm phát hiện các hiện tượng bất thường, các hoạt động xâm nhập trái phép vào hệ thống mạng.IDS có thể phân biệt được những tấn công từ bên trong (nội bộ) và từ bên ngoài (do tin tặc tấn công).

IDS phát hiện tấn công dựa trên các dấu hiệu khác thường ứng với các nguy cơ đã biết điều này tương tự như các các phần mềm diệt Virus phát hiện và tiêu diệt virus, ngoài IDS còn phân tích lưu thông mạng hiện tại và so sánh nó với thông số đo đạt chuẩn của hệ thống có thể chấp nhận được tại thời điểm đo để tìm ra các dấu hiệu khác thường.

Các tính năng quan trọng nhất của hệ thống phát hiện xâm nhập IDS là [1]:

- Giám sát lưu lượng mạng và các hoạt động khả nghi

- Cảnh báo về tình trạng mạng cho hệ thống nhà quản trị.

Kết hợp với các hệ thống giám sát, tường lửa ,diệt virus tạo thành một hệ thống bảo mật hoàn chỉnh

2.1.1. Phân loại IDS

Hệ thống IDS được chia làm hai loại theo phạm vi giám sát là là NIDS (Network-Based IDS) và HIDS (Host – Based IDS) :

NIDS[1] : Hệ thống phát hiện xâm nhập mạng. Hệ thống sẽ tập hợp gói tin để phân tích sâu bên trong mà không làm thay đổi cấu trúc gói tin như quét header kiểm tra nội dung gói để phát hiện các đoạn mã nguy hiểm hay các dạng tấn công. NIDS có thể là phần mềm triển khai trên server hoặc dạng thiết bị tích hợp appliance.

- Ưu điểm của NIDS [3] :

+ Quản lý được một network segment (gồm nhiều host).

+ Cài đặt và bảo trì đơn giản, không ảnh hưởng đến mạng.

+ Có khả năng xác định lỗi ở tần network

- Nhược điểm NIDS [3]:

+ Có thể xảy ra trường hợp báo động giả

+ Không thể phân tích dữ liệu đã được mã hóa như SHH hay SSL..

+ Đòi hỏi cần cập nhập các signture mới nhất để thực sự an toàn.

+ Có độ trễ giữa thời điểm bị tấn công với thời điểm phát hiện báo động.

Các NIDS phổ biến ngày nay có thể kể đến như Snort , Cisco Secure IDS, Dragon Enterasys…

HIDS[1] : Hệ thống phát hiện xâm nhập host. Theo dõi các hoạt động bất thường như các tiến trình, các entry, mức độ sử dụng CPU, tình trạng ram trên các host riêng biệt. HIDS được cài đặt trực tiếp trên các máy host cần theo dõi.

- Ưu điểm của HIDS:

+ Có khả năng xác định các user liên quan đến sự kiện.

+ Có thể phát hiện tấn công xảy ra trên một máy cụ thể trong khi NIDS thì không.

+ Có thể phân tích dữ liệu được mã hóa, cung cấp thông tin về host trong lúc cuộc tấn công diễn ra trên chính host đó.

- Nhược điểm HIDS:

+ Thông tin từ HIDS không đáng tin cây ngay khi sự tấn công vào host này thành công

+ Khi OS bị sập do tấn công, đồng thời HIDS cũng sập

+ HIDS phải được thiết lập trên từng host cần giám sát

+ Không có khả năng phát hiện các cuộc dò quét mạng (Nmap, Netcat ..)

+ Cần tài nguyên host để hoạt động và có thể trở lên không hiệu quả khi bị tấn Dos.

Các HIDS phổ biến : Intrust event admin Aelita, ELM 3.0 TNT Software, GFI LANguard S.E.L.

Ngoài được phân phân loại theo phạm vi giám sát thì IDS còn được phần loại theo kĩ thuật thực hiện, gồm có 2 loại là signature-based IDS và Anomaly-based IDS.

● Signature-based IDS

- Signature-based [14] IDS phát hiện xâm nhập dựa trên dấu hiệu của hành vi xâm nhập, thông qua phân tích lưu lượng mạng và nhật ký hệ thống. Kỹ thuật này đòi hỏi phải duy trì một cơ sở dữ liệu về các dấu hiệu xâm nhập gọi là signature database, và cơ sở dữ liệu này phải được cập nhật thường xuyên mỗi khi có một hình thức hoặc kỹ thuật xâm nhập mới.

● Anomaly-based IDS:

- Anomaly-based IDS [14] phát hiện xâm nhập bằng cách so sánh thống kê các hành vị hiện tại với hoạt động bình thường của hệ thống để phát hiện các bất thường có thể là dấu hiệu xâm nhập. Để có thể hoạt động một cách chính xác thì ids cần phải thực hiện một quá trình học, tức là giám sát hoạt động của hệ thống trong điều kiện bình thường để ghi nhận các thông số hoạt động, đây là cơ sở để nó có thể phát hiện được các hoạt động bất thường về sau.

2.1.2. Hệ thống luật của IDS

Tập luật là thành phần quan trọng nhất của một hệ thống IDS [1]. Đây là thành phần định ra các dấu hiệu để có thể so sánh với dữ liệu đầu vào. Thông thường tập luật sẽ gồm có nhiều luật khác nhau, mỗi luật cơ bản sẽ có 2 thành phần chính là Header và Options.

● Header của một luật sẽ bao gồm các thông tin:

- Action : Xác định các hành động được thực thi khi luật được kích hoạt như alert,log,pass,active,dynamic,drop..

- Protocol : Xác định phương thức sẽ được khớp luật (TCP,UDP,ICMP..)

- IP address : Cho biết thông tin về địa chỉ ip

- Port number : Cho biết thông tin về cổng

- Direction : Cho biết hướng dữ liệu mà được so khớp

● Options có 4 phần:

- General : cung cấp thông tin chung về luật (msg, reference, rev ,classtype..)

- Payload: Tìm kiếm nội dung payload của gói tin

- Non-payload: Tìm kiếm nội dung non-payload của gói tin (ttl, ack, tos, id…)

- Post-detection: Cung cấp các phương pháp thực thi kế tiếp (logto, session, tag…)

- Ví dụ về một luật của IDS snort nhằm phát hiện quét SYN FIN với hệ thống đích:

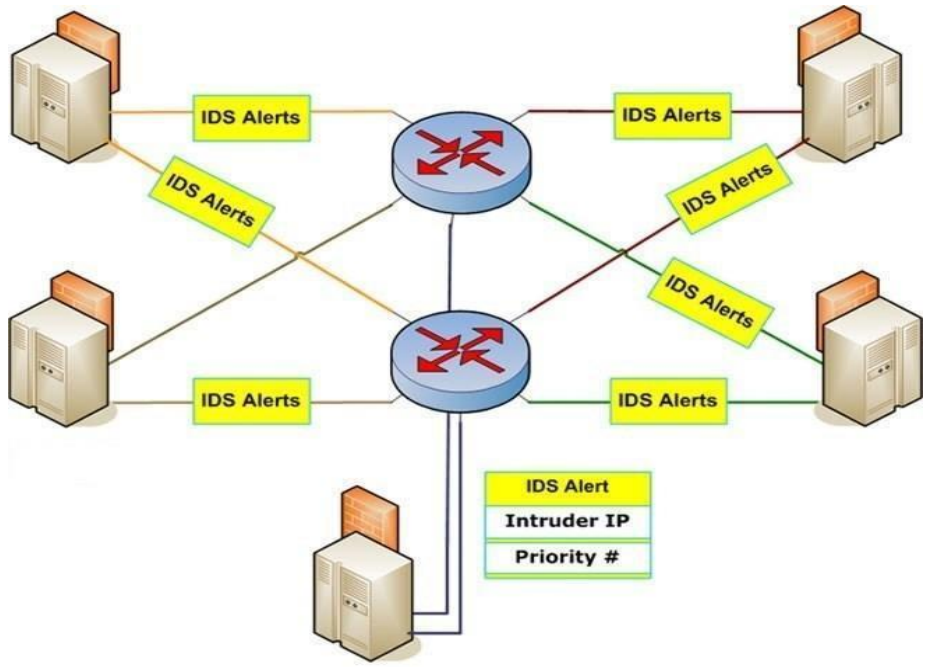
```
alert TCP $EXTERNAL any -> $INTERNAL any (msg: "IDS198/scan_SYN FIN
  Scan"; flags: SF; classtype: info-attempt; reference: arachnids,198;)
```

2.1.3. Các mô hình mạng tích hợp IDS

Tùy thuộc vào các mục đích khác nhau cũng như cấu trúc mạng của hệ thống trong thực tế mà việc tích hợp IDS cũng khác nhau về vị trí để đạt hiệu quả tối đa [1].

- Đặt giữa router và firewall

- Khi đặt trong trường hợp này [1], IDS sẽ theo dõi tất cả các lưu lượng trên cả 2 chiều. Khi triển khai theo cấu trúc này thì IDS phải chịu áp lực lớn về lượng, nhưng lại có khả năng giám sát toàn bộ lưu lượng hệ thống mạng. Vì vậy, trong trường hợp này nên lựa chọn các thiết bị IDS có khả năng chịu tải cao về hiệu năng.

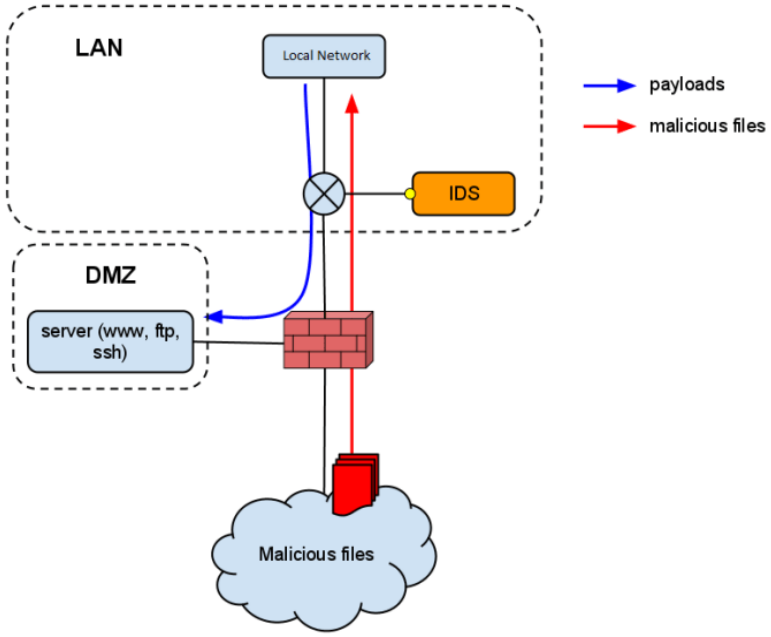


SecurityDaily.Net

Hình 2.1 IDS đặt giữa router và firewall (SecurityDaily.Net)

- Đặt trong miền DMZ

- Trong trường hợp này, IDS sẽ theo dõi tất cả các lưu lượng vào và ra trong miền DMZ [1]

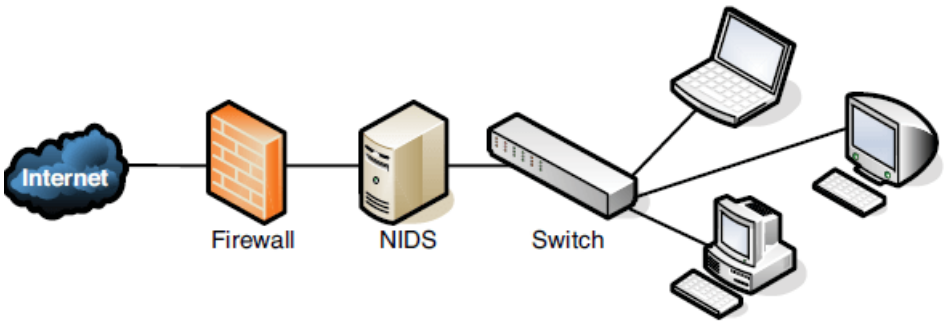


Nguồn: SecurityDaily.Net

Hình 2.2 Mô hình IDS trong miền DMZ

- Đặt sau firewall

- Khi đặt trong trường hợp này, IDS sẽ theo dõi tất cả lưu lượng trao đổi phía sau firewall [1] như: dữ liệu trao đổi trong LAN, hay dữ liệu từ LAN vào ra miền DMZ và ngược lại.



Nguồn: SecurityDaily.Net

Hình 2.3 IDS đặt sau Firewall

2.1.4 Các tiêu chí khi triển khai IDS

- Xác định công nghệ IDS đã đang hoặc dự định triển khai
- Xác định các thành phần của IDS
- Thiết lập cài đặt cấu hình an toàn cho IDS

- Xác định vị trí phù hợp để đặt IDS
- Có cơ chế xây dựng, tổ chức, quản lý hệ thống luật
- Hạn chế thấp nhất các tình huống có thể gây cảnh báo nhầm hoặc bỏ qua xâm nhập mà không có cảnh báo

2.2 Hệ thống IPS

Hệ thống IPS (Intrusion Prevention Systems – Hệ thống ngăn ngừa xâm nhập) [2] là hệ thống theo dõi ngăn ngừa kịp thời các hoạt động xâm nhập không mong muốn. Chức năng chính của IPS là xác định các hoạt động nguy hại, lưu giữ các thông tin này. Sau đó kết hợp với firewall để dừng ngay các hoạt động này, và cuối cùng đưa ra các báo chi tiết về các hoạt động xâm nhập trái phép trên. IPS có thể được xem là trường hợp mở rộng của IDS các thực thức hoạt động tương tự nhau, điểm khác nhau duy nhất đó là IPS ngoài khả năng theo dõi, giám sát thì còn có chức năng ngăn chặn kịp thời các hoạt động nguy hại đối với hệ thống. Hệ thống IPS cũng dùng các tập luật tương tự như IDS

2.2.1 Phân loại và ưu nhược điểm của IPS

Tương tự như IDS thì hệ thống IPS cũng được phân làm 2 loại là Hệ thống ngăn ngừa xâm nhập mạng (NIPS – Network-based Intrusion Prevention) và Hệ thống ngăn ngừa xâm nhập Host (HIPS – Host – Based Intrusion Prevention)

- Hệ thống ngăn ngừa xâm nhập mạng (NIPS)

Thường được triển khai trước hoặc sau firewall. Khi triển khai IPS trước firewall là có thể bảo vệ được hệ thống bên trong kể cả firewall, vùng DMZ. Có thể giảm thiểu nguy cơ bị tấn công từ chối dịch vụ đối với Firewall. Khi triển khai IPS sau firewall có thể tránh được một số kiểu tấn công khai thác điểm yếu của thiết bị di động và sử dụng VPN để kết nối vào bên trong.

- Hệ thống ngăn ngừa xâm nhập host (HIPS)

Thường được triển khai với mục đích phát hiện và ngăn chặn kịp thời các hoạt động thâm nhập trên các host. Để có thể ngăn chặn chạy các tấn công, HIPS sử dụng công nghệ tương tự như các giải pháp antivirus giống như HIDS. Ngoài khả năng phát hiện ngăn ngừa các hoạt động thâm nhập HIPS còn có khả năng phát hiện sự thay đổi các tập tin cấu hình.

- Ưu điểm:

- Cung cấp giải pháp bảo vệ toàn diện hơn đối với tài nguyên hệ thống

- Ngăn chặn kịp thời các tấn công đã biết hoặc chưa biết

● Hạn chế:

- Có thể gây ra tình trạng phát hiện nhầm không đáng có ảnh hưởng đến quá trình quản trị hệ thống

- Có thể không cho phép các truy cập hợp lệ tới hệ thống, gây ảnh hưởng đến hoạt động của hệ thống mạng.

2.2.2 Các mô hình mạng tích hợp IPS

● Đặt trước firewall:

- Trong mô hình này IPS được đặt trước tường lửa. Đây là mô hình được đề xuất đầu tiên khi IPS xuất hiện trên thị trường, tuy nhiên ngày nay mô hình này không còn quá phổ biến. IPS đặt trước firewal sẽ đóng vai trò như bộ lọc lưu lượng trước khi nó đên firewall

- Ưu điểm:

+ Phát hiện sớm các hoạt động nguy hiểm

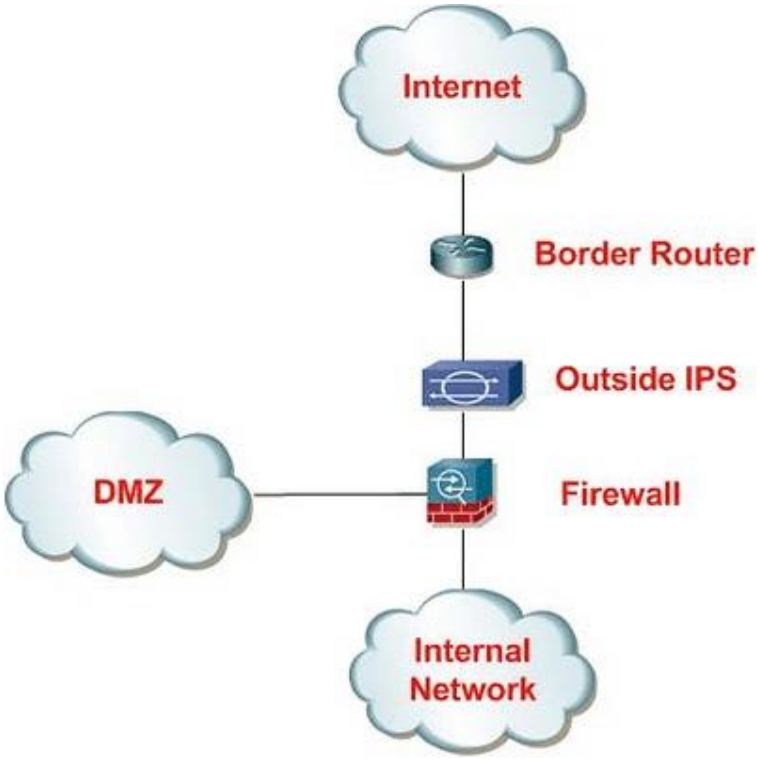
+ Giảm bớt gánh nặng và tiết kiệm tài nguyên cho firewall

- Nhược điểm:

+ Khi một thiết bị bên trong mạng nội bộ bị tấn công sẽ rất khó để xác định vi địa chỉ đích cần được NAT

+ Có thể tạo các cảnh báo dư thừa

+ Không thể kiểm tra lưu lượng truy cập nội bộ đến dmz



Nguồn: SecurityDaily.Net

Hình 2.4 Ips đặt trước Firewall

- Đặt bên trong tường lửa

  - Kiến trúc này đặt IPS bên trong tường lửa internet giúp bảo vệ cả mạng nội bộ cũng như miền DMZ

  - Ưu điểm :

  + Kiểm tra lưu lượng mà tường lửa cho phép vào mạng làm giảm thiểu các cảnh báo sai

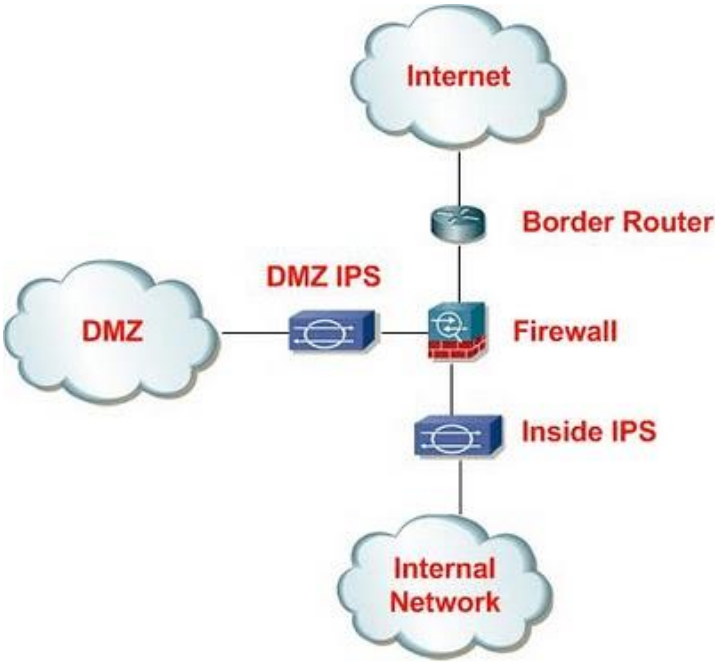
  + Các sự kiện sẽ bao gồm cả địa chỉ IP thực và địa chỉ IP NAT

  + Phân biệt dược lưu lượng truy cập giữa mạng nội bộ và miền DMZ

  + Nhược điểm :

  + Yêu cầu có 2 IPS hoặc IPS có đủ Interface để bảo vệ cả 2 vùng

  + Lưu lương truy cập giữa mạng nội bộ và DMZ sẽ bị kiểm tra 2 lần k cần thiết.

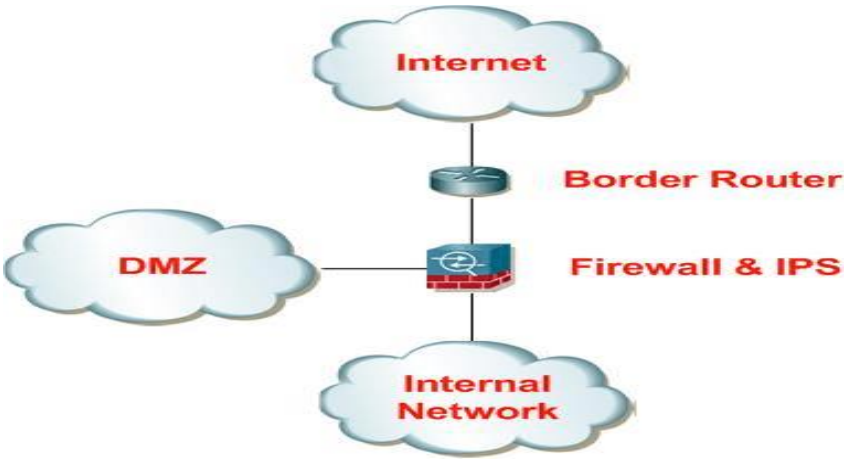


Nguồn: SecurityDaily.Net

Hình 2.5 Ips đặt giữa firewall và miền DMZ

●Là một module trong giải pháp UTM (Unified Threat Management)

- UTM là giải pháp hợp nhiều chức năng bảo mật [5] trong đó có bao gồm cả ips thành một thiết bị duy nhất, vì vậy một trong những cách triển khai IPS đó chính là tích hợp nó vào một thiết bị UTM cói như là một module của thiết bị. Đây là một cách hiệu quả và đơn giản để đối phó với các mối đe dọa an ninh mạng phức tạp và đang ngày càng phát triển. Giải pháp này phù hợp nhất với các tổ chức vừa và nhỏ để quản lý các mối đe dọa mới nhất đồng thời làm giảm thiểu đáng cả các chi phí.



Nguồn: SecurityDaily.Net

Hình 2.6 IPS triển khai như module của giải pháp UTM

2.3 Tìm hiểu Suricata

2.3.1 Giới thiệu chung

Suricata là một công cụ Giám sát an ninh mạng IPS và IDS với hiệu năng cao. Nó là một công cụ mã nguồn mở được phát triển và sở hữu bởi OISF (Open Information Security Foundation), đây là một tổ chức phi lợi nhuận.

Suricata có khả năng phát hiện xâm nhập theo thời gian thực (IDS), ngăn chặn xâm nhập (IPS), giám sát an ninh mạng (NSM) và xử lý pcap ngoại tuyến.

Công cụ này sử dụng các quy tắc, ngôn ngữ chữ ký và các tập lệnh Lua để phát hiện các mối đe dọa. Suricata hỗ trợ các chuẩn đầu vào và đầu ra dưới dạng YAML hay Json tích hợp cùng một số công cụ có sẵn như SIEMs, Splunk, Logstash/Elasticsearch, kibana, và các cơ sở dữ liệu khác trở lên dễ dàng hơn.

Đây là một công cụ đa luồng giúp tăng tốc độ hiệu quả cho việc phân tích lưu lượng mạng, tăng hiệu năng của phần cứng. [6]Công cụ này được tạo ra để tận dụng được khả năng xử lý cao được cung cấp bởi chip CPU đa lõi mới nhất.

2.3.2 Các chức năng chính của Suricata

Suricata được giới thiệu là có các chức năng phát hiện và ngăn chặn xâm nhập (IDS/IPS), giám sát an ninh mạng (Network Security Monitoring – NSM) và xử lý gói PCAP.

Về hệ thống IDS/IPS chúng ta đã tìm hiểu ở phần 2.1 và 2.2 nên phần này chúng ta sẽ chỉ tìm hiểu xem công nghệ NSM và xử lý PCAP là gì

- Giám sát an ninh mạng – Network Security Monitoring :

- Là việc thu thập các thông tin [6] 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 giám sát an ninh mạng là tất cả các thành phần, thiết bị trong hệ thống mạng như :

+ Các máy trạm

+ Các cơ sở dữ liệu

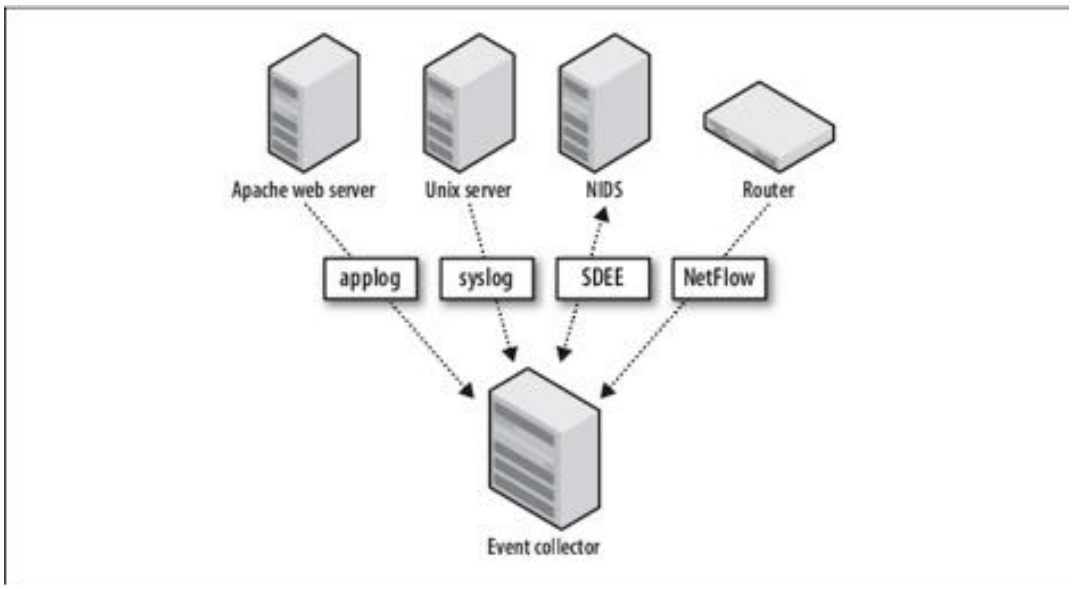
+ Các ứng dụng

+ Các server

+ Các thiết bị mạng

- Các bước mà một hệ thống NSM thực hiện để giám sát gồm có:

+ Thu thập dữ liệu : lấy các thông tin liên quan đến tình trạng hoạt động của các thiết bị trong hệ thống mạng. Để dễ dàng cho việc thu thập thông tin trên nhiều loại thiết bị với các nền tảng và địa điểm khác nhau thì mô hình Log tập trung được đưa ra để giải quyết vấn đề này với 2 cách thức thu thập dữ liệu chính theo mô hình này là phương pháp đẩy (The Push Method) và phương pháp kéo (The Pull Method)



Nguồn: SecurityDaily.net

Hình 2.7 Phương pháp đẩy

+ Phân tích dữ liệu : Khi dữ liệu được thu thập những thông tin [24] về hệ thống thì công việc tiếp theo của hệ thống NSM là phân tích thông tin. Cụ thể việc thực hiện chỉ mục hóa dữ liệu, phát hiện các bất thường các mối đe dọa đến hệ thống mạng. Dựa trên những thông tin về lưu lượng truy cập, trạng thái truy cập, định dạng request

+ Cảnh báo : Sau khi thực hiện việc phân tích dữ liệu hoàn thành đánh giá sẽ được đưa ra và đưa thông tin cảnh báo tới người quản trị để thực hiện những tiến trình nhằm chống lại những mối đe dọa và khắc phục sự cố có thể xảy ra.

- Xử lý PCAP (Capture Packet):

- PCAP là giao diện lập trình ứng dụng (PCAP) dùng để thu thập dữ liệu gói mạng trực tiếp từ mô hình OSI các tầng từ 2-7. PCAP thực hiện các chức năng lọc gói [6] dữ

liệu theo những tập luật của người dùng khi chúng được truyền tới ứng dụng, truyền những gói dữ liệu thô tới mạng, thu thập và thống kê lưu lượng mạng. PCAP có một số thư viện như Libpcap, WinPcap, và PCAPng

- Suricata cũng cung cấp các tùy chọn để chạy công cụ ở chế độ PCAP với các cấu hình được lấy trong phần pcap ở file cấu hình.

2.3.3 Luật (Rules) trong Suricata

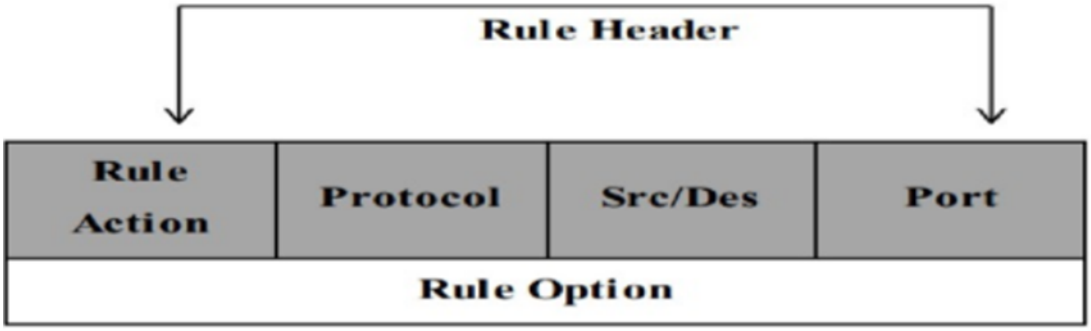
Suricata là một công cụ IDS/IPS vì vậy nó cũng sử dụng các tập luật để xác định các dấu hiệu bất thường trong lưu lượng mạng. Về cơ bản luật trong Suricata là nơi định ra các ngữ cảnh có thể xảy ra thực tế trong hệ thống mạng cho thấy rằng đang có dấu hiệu bất thường, có nguy cơ bị xâm nhập, công cụ sẽ so sánh các thông tin thu thập được sau khi phân tích lưu lượng mạng và so khớp với các tập luật từ đó đưa ra các hành động cụ thể.

Suricata cung cấp sẵn cho người dùng một lượng lớn các tập luật để có thể download và sử dụng cho hệ thống của mình, các tập luật cũng thường xuyên được cập nhật nhằm đáp ứng những thay đổi trong thực tế khi các hình thức xâm nhập có sự thay đổi.

Ngoài cung cấp sẵn các tập luật thì Suricata cũng hỗ trợ người quản trị viên có thể tự viết và áp dụng các tập luật của riêng mình. Điều này giúp cho việc tích hợp Suricata vào hệ thống mạng trở lên linh hoạt và mềm dẻo hơn, phù hợp các đặc điểm của từng mô hình mạng khác nhau trong thực tế.

Để có thể tự mình viết được các luật riêng có thể hoạt động được các quản trị viên cần tìm hiểu rõ về luật và áp dụng đúng các quy tắc viết luật trong Suricata. Một luật trong Suricata tuân thủ các nguyên tắc chung về luật IDS, mỗi luật sẽ bao gồm 2 phần chính là Rule Header và Rule Option.

- Rule Header



Nguồn: suricata.readthedocs.io

Hình 2.8 Các thành phần Header rule

Phần header trong một rule bao gồm các thông tin như hành động được thực thi (action) , giao thức mạng khớp với luật (protocol), các thông tin về địa chỉ như địa chỉ ip nguồn/đích số port , subnet mask.

- Action rule

Đây là thành phần quan trọng của một luật nằm ở phần header, nó là nơi quy định xem hành động gì sẽ được thực thi khi có gói tin phù hợp với các thông tin được định sẵn trong luật, action trong Suricata mặc định có kiểu : Pass , Drop , reject, alert.

+ Pass : Nếu signature trùng khớp và action được chỉ định là pass thì Suricata sẽ ngừng quét gói tin đồng thời bỏ qua tất cả các luật khác đối với gói tin này, thường được áp dụng cho các gói tin đặc biệt.

+ Drop : là action được sử dụng ở chế độ IPS/inline. Nếu công cụ nhận thấy một signature trùng khớp với chỉ định action là drop nó sẽ dừng ngay lập tức. Gói tin sẽ không được gửi đi nữa. Phía người nhận sẽ không nhận được thông báo về những gì đang xảy ra dẫn đến hết thời gian chờ (giao thức TCP). Suricata sẽ tạo ra một cảnh báo cho gói tin này.

+ Reject : Đây là hành động từ chối gói tin. Khác với drop cả người nhận và người gửi đều sẽ nhận được một gói tin từ chối. Có hai loại gói tin từ chối sẽ được chọn tự động.Nếu gói vi phạm liên quan đến TCP, nó sẽ là gói Reset. Còn đối với tất cả các giao thức khác nó sẽ là gói lỗi ICMP. Suricata cũng tạo ra một cảnh báo.Khi Suricata được chạy ở chế độ IPS/Inline thì gói tin vi phạm cũng sẽ bị loại bỏ giống như action drop (Suricata vẫn sẽ gửi thông báo cho người nhận và gửi).

+ Alert : Nếu một signature trùng khớp với action chỉ định là alert, gói tin sẽ được coi như là một gói tin không hợp lệ và một cảnh báo sẽ được Suricata tạo ra và chỉ quản trị viên có thể nhìn thấy cảnh báo này.

+ Các luật sẽ được tải theo thứ tự xuất hiện trong tệp, tuy nhiên chúng sẽ được xử lý theo một thứ tự khác. Các signature có các thứ tự ưu tiên khác nhau, các signature quan trọng sẽ được quét trước. Các quản trị viên có thể thay đổi các thứ tự ưu tiên này trong tệp cấu hình của Suricata. Theo mặc định thứ tự sẽ là : Pass, drop, reject, alert

```
action-order:
 - pass
 - drop
 - reject
 - alert
```

Nguồn: suricata.readthedocs.io

Hình 2.9. Action rule

- Protocol

Đây là một keyword nằm trong phần header của luật, nó cho biết giao thức nào được khớp với luật đó. Chúng thường sử dụng 4 giao thức cơ bản là : TCP, UDP, ICMP,IP

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Likely Bot Nick in IRC (USA +..)";
flow:established,to_server; flowbits:isset,is_proto_irc; content:"NICK "; pcre:"/NICK .*USA.*[0-9]
{3,}/i"; reference:url,doc.emergingthreats.net/2008124; classtype:trojan-activity; sid:2008124;
rev:2;)
```

Nguồn: suricata.readthedocs.io

Hình 2.10 Suricata Rule với giao thức TCP

-Khi một luật được tạo ra và được tích hợp vào trong Suricata thì luật đó chỉ được so khớp khi có lưu lượng mạng liên quan đến giao thức đã được chỉ định trong luật.

Ví dụ : Nếu có một signature với giao thức http thì Suricata sẽ chỉ so khớp signature đó với các lưu lượng truy cập http.

- Nguồn và đích

- Đây là nới chỉ định nguồn của lưu lượng cũng như đích đến tương ứng của lưu lượng đó. Chúng ta có thể gán địa chỉ IP cụ thể (Suricata hỗ trợ cả Ipv4 và Ipv6) hoặc có thể gán một giải IP .Suricata cũng cấp một số toán tử hỗ trợ việc gán địa chỉ :

Operator	Description
../..	IP ranges (CIDR notation)
!	exception/negation
[.., ..]	grouping

Nguồn: suricata.readthedocs.io

Hình 2.11 Các toán tử xác nhận địa chỉ IP

- Thông thường quản trị viên có thể sử dụng các biến như $HOME_NET và $EXTERNAL_NET thay vì việc phải ghi rõ một địa chỉ IP hay một giải địa chỉ IP cụ thể trong mỗi luật. Các biến này được gán giá trị trong tệp cấu hình.

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Likely Bot Nick in IRC
(USA +..)"; flow:established,to_server; flowbits:isset,is_proto_irc; content:"NICK ";
pcre:"/NICK .*USA.*[0-9]{3,}/i"; reference:url,doc.emergingthreats.net/2008124;
classtype:trojan-activity; sid:2008124; rev:2;)
```

Nguồn: suricata.readthedocs.io

Hình 2.12 Sử dụng biến môi trường thay thế địa chỉ Ip

- Trong ví dụ trên $HOME_NET được cho là nguồn và $EXTERNAL_NET là đích (có thể căn cứ vào hướng mũi tên).

- Cổng (Nguồn và đích)

- Lưu lượng ra vào qua các cổng khác nhau với số hiệu khác nhau. Ví dụ cổng mặc định của HTTP là 80 và HTTPS là 443. Tuy vậy từ số hiệu của cổng không thể chỉ định ra một giao thức cụ thể nào được sử dụng trong giao tiếp. Thay vào đó nó xác định ứng dụng nào đang nhận dữ liệu.

- Khi nhắc đến cổng trong các luật thường ta chú ý đến các cổng đích, trong khi các cổng nguồn (nơi gói tin được gửi đi) thường được hệ điều hành gán cho cổng ngẫu nhiên. Ví dụ : khi viết một luật cho dịch vụ HTTP ta thường viết any -> 80 có nghĩa là bất kì gói tin nào đến từ bất kỳ cổng nào được gửi đến ứng dụng HTTP (chạy trên cổng 80) đều sẽ được khớp.

- Ví dụ:

Example	Meaning
[80, 81, 82]	port 80, 81 and 82
[80: 82]	Range from 80 till 82
[1024:]	From 1024 till the highest port-number
!80	Every port but 80
[80:100,!99]	Range from 80 till 100 but 99 excluded
[1:80,![2,4]]	Range from 1-80, except ports 2 and 4
[.., [..,..]]	

Nguồn: suricata.readthedocs.io

Hình 2.13. Ví dụ cách định nghĩa port trong Rule

- Thêm luật vào Suricata

Sau khi tìm hiểu về luật trong Suricata các quản trị viên có thể tùy theo tình hình thực tế mà tự mình tạo ra các luật phù hợp để sử dụng cho hệ thống của mình và để các luật này hoạt động thì cần tích hợp chúng vào công cụ suricata.

- B1: Chúng ta cần tạo một file .rules trong đường dẫn /var/lib/suricata/rules

- B2: viết luật chúng ta đã tạo ra vào trong file .rules đó và lưu lại

-B3: cần chỉnh sửa file config suricata.yaml với đường dẫn /etc/suricata/suricata.yaml, chúng ta thêm tên file .rules vừa tạo vào phần rule – files

- B4: khởi động lại Suricata

2.3.4 Suricata Command Line

Suricata cung cấp cho người các command line để thực hiện các điều khiển công cụ này. Các command này định dạng bắt bắt đầu bằng "suricata" kèm theo là các option khác nhau

Một vài option thường dùng khi sử dụng command line của Suricata:

+ **-c <path>** với path là đường dẫn [22] đến file cấu hình, thường là **"/etc/suricata/suricata.yaml",** option được sử dụng khi khởi động Suricata với các thông số được cấu hình trong file cấu hình.

+ **-i <interface>** dùng để chỉ định interface mà người cài đặt muốn Suricata giám sát. Có thể sử dụng nhiều option này trong một câu lệnh để có thể giám sát nhiều interface

+ **-r <path>** thường được sử dụng trong chế độ ngoại tuyến, để đọc và phân tích các tệp pcap. Nếu đường dẫn chỉ tới một thư mục thì tất cả các tệp có trong thư mục sẽ được đọc.

+ **-s <filename.rules>** tải tệp rules cùng với các tệp có tên trong file cấu hình

+ **-l <directory>** chỉ định thư mục để ghi log.

+ **--runmodes<runmodes>** tùy chọn để chỉ định chế độ chạy mong muốn, tùy chọn này có thể ghi đè chế độ chạy được mặc định trong file cấu hình

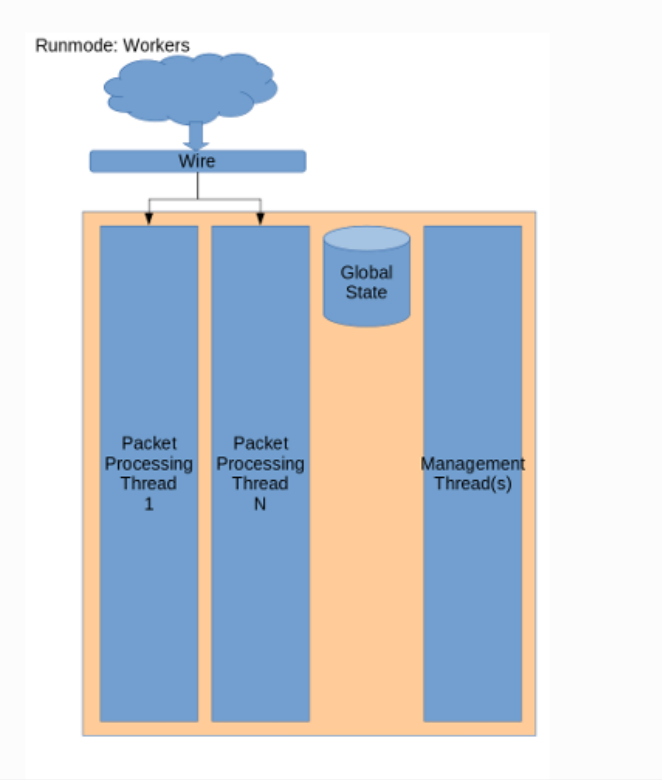
+ **-D** tùy chọn chỉ thị suricata chạy dưới chế độ nền không chiếm dụng bảng điều khiển.

2.3.4 Các chế độ chạy của Suricata

Suricata là một chương trình được tạo nên từ nhiều khối được gọi là luồng [23], module luồng, và hàng đợi.Luồng giống như tiến trình chạy trên máy tính và Suricata chương trình đa luồng với nhiều luồng được chạy cùng một lúc.Module luồng là tiến trình đang chạy của các chức năng. Ví dụ một module dùng để giải mã các gói tin,trong khi module khác thực hiện phát hiện. Một gói tin có thể được xử lý bởi nhiều luồng,việc chuyển các gói tin giữa các luồng được thông qua các hàng đợi. Tại một thời điểm một gói tin được xử lý bởi một luồng nhưng sẽ có nhiều gói tin được xử lý cùng lúc. Các luồng có thể có nhiều hơn một module luồng. Cách các luồng,module luồng và hàng đợi sắp xếp với nhau được gọi là Runmode hay chế độ chạy. Suricata có 3 chế độ chạy

- Worker

- Đây là luồng hoạt động tốt nhất, ở chế độ này [23] trình điều khiển đảm bảo các gói tin được phân bổ cân bằng cho các luồng, các luồng lúc này sẽ có đầy đủ các module luồng

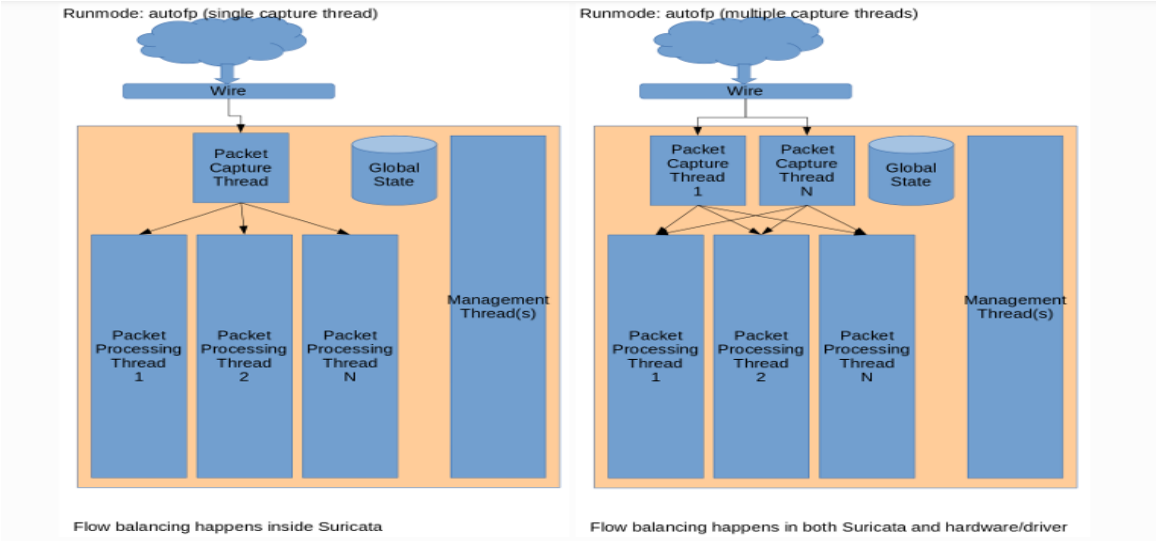


Nguồn: suricata.readthedocs.io

Hình 2.14 Worker Run mode

- Autofp

- Thường được sử dụng ở chế độ xử lý tệp PCAP [23] hoặc ở chế độ IPS. Ở chế độ này sẽ có một hoặc chiều luồng đảm nhập việc bắt và giải mã các gói tin trước khi được chuyển cho các luồng worker



Nguồn: suricata.readthedocs.io

Hình 2.15 Autofp Run Mode

- Single

Hoạt động giống như chế độ worker, tuy nhiên chỉ có một luồng duy nhất chạy thường được dùng trong quá trình phát triển.

2.4 Tìm Hiểu về RabbitMQ

RabbitMQ là một message broker (message-oriented middleware) [10] là một phần mềm quản lý hàng đợi message, hay có thể nói là nơi định nghĩa ra các hàng đợi giúp các ứng dụng có thể vận chuyển message. RabbitMQ sử dụng giao thức AMQP Advanced Message Queue Protocol, trên thực tế RabbitMQ còn hỗ trợ nhiều giao thức khác tuy nhiên trong đồ án này sẽ chỉ tập trung vào giao thức AMQP.RabbitMQ được phát triển bằng ngôn ngữ Erlang một ngôn ngữ thường được sử dụng cho các ứng dụng thời gian thực được sử dụng trong các lĩnh vực như viên thông, thương mại điện tử hay ngân hàng.

RabbitMQ cung cấp cho các nhà phát triển một phương tiện trung gian để giao tiếp giữa các thành phần trong một hệ thống lớn. Nhiệm vụ của RabbitMQ đơn giản là nhận các message từ các thành phần khác nhau trong hệ thống, lưu trữ chúng an toàn trước khi được đẩy đến đích một các chính xác điều này rất phù hợp để ứng dụng giải quyết bài toán trong đồ án này.

2.4.1 Giao thức AMQP

Trước khi tìm hiểu kĩ hơn về RabbitMQ chúng ta cần tìm hiểu về một trong những giao thức phổ biến được hỗ trợ bởi RabbitMQ là giao thức AMQP trong đó sẽ tập trung vào giao thức AMQP 0-9-1 đây là giao thức cốt lõi được hỗ trợ bởi RabbitMQ.

AMQP là một giao thức internet mở và được chuẩn hóa để có thể truyền message tin cậy giữa các ứng dụng (tầng 7) hoặc tổ chức [7]. AMQP giúp các chuyên gia công nghệ thông tin xây dựng một hệ sinh thái tin nhắn đồng nhất, đa dạng , kết nối các hệ thống với độ tương tác cao.

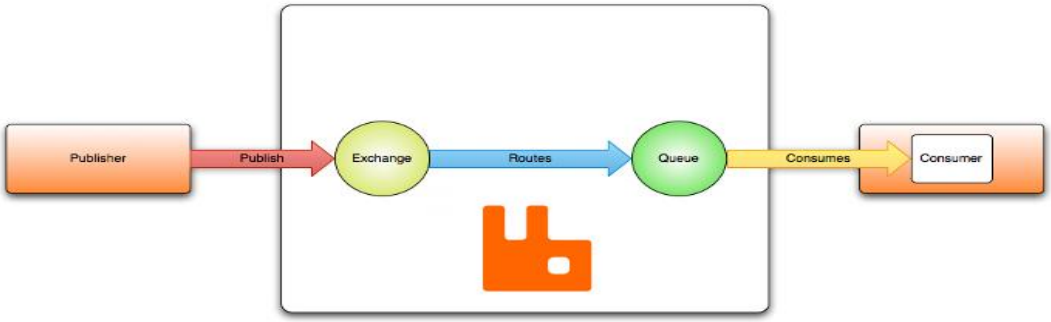
Các tính năng chính của AMQP là định hướng message, hàng đợi, định tuyến bao gồm cả point-to-point và publish-subscribe với tính bảo mật và độ tin cậy cao, các hoạt động sẽ được thông hiện thông qua broker, nó cung cấp khả năng điều khiển luồng (flow control).AMQO là một giao thức có dây (wire-protocol) [8], có khả năng diễn tả các message phù hợp với định dạng dữ liệu và có thể triển khai với rất nhiều loại ngôn ngữ lập trình.

- Lịch sử của AMQP :

  - 2003 AMQP được bắt đầu phát triển bởi JP Morgan
  - 2004 Nhóm phát triển AMQP được thành lập
  - 2006 AMQP 0-8 được phát hành vào tháng 6 và 0-9 vào tháng 12
  - 2011 AMQP 1.0 được phát hành bởi nhóm phát triển

- Mô hình hoạt động của AMQP

  - Mô hình AMQP có cách hoạt động như sau : các message được các Publisher đẩy đến các Exchanges, có thể coi đẩy giống như các bưu điện trong thực tế. Nhiệm vụ của các Exchanges này là phân phối các bản sao message đến với các Queue (hàng đợi) bằng cách sử dụng các quy tắc được gọi là Bindings (ràng buộc). Sau đó broker sẽ gửi các thông điệp này cho các Consumer đã đăng ký các Queue hoặc các Consumer nạp/kéo các message từ các Queue theo yêu cầu.



Nguồn: www.rabbitmq.com

Hình 2.16 Mô hình AMQP

- Khi xuất bản một message các Publisher có thể chỉ định các thuộc tính message khác nhau (message meta-data). Các thuộc tính này có một sẽ được sử dụng bởi các Exchange như routing-key, phần còn lại sẽ được sử dụng bởi các ứng dụng nhận thông báo (Consumer).

- Do một số trường hợp mà mạng sẽ trở lên không đáng tin cậy và các ứng dụng không thể xử lý các thông báo do đó AMQP 0-9-1 đưa khái niệm xác nhận thông báo: Khi một thông báo được gửi đến Consumer thì Consumer đó sẽ thông lại cho broker một cách tự động hoặc tại thời điểm nào đó mà người phát triển ứng dụng lựa chọn. Khi sử dụng tính năng xác nhận thông báo thì Broker sẽ chỉ xóa hoàn toàn tin nhắn khỏi hàng đợi khi nó nhận được thông báo xác nhận cho message đó.

- Trong một số tình huống nhất định, như : không thể định tuyến thư, thư sẽ có thể bị trả lại publisher hoặc bị loại bỏ, hoặc broker sử dụng tính năng mở rộng gọi là hàng đợi thư chết.

- Ngoài các khái niệm như queue, exchange, binding được gọi là các thực thể AMQP.

- Giao thức AMQP (0-9-1) là giao thức tương đối linh hoạt hay một giao thức có thể lập trình được, theo đó thì các thực thể cũng như lược đồ định tuyến của giao thức này phụ thuộc chủ yếu chính ứng dụng sử dụng chứ không phải broker.Thông các các hoạt động như khai báo Queue ,Exchange cũng như xác định ràng buộc giữ chúng các nhà phát triển có thể tùy chỉnh cách hoạt truyền tin sao cho phù hợp.

- Một vài broker sử dụng giao thức AMQP
  - RabbitMQ
  - Windows Azure Service Bus
  - Apache Qpid
  - Apache ActiveMQ
  - SwiftMQ
  - StormMQ
  - WSO2 message broker
  - OpenAMQ

2.4.2 RabbitMQ và Tại sao nên sử dụng RabbitMQ

- Như đã nói ở trên RabbitMQ là message broker hay còn gọi là integration broker hoặc interface engine, đây là một module trung gian [9] có nhiệm vụ trung chuyển message từ người gửi đến người nhận, nó là một mô hình kiến trúc để kiểm tra trung chuyển và điều hướng message, làm trung gian giữa các ứng dụng với nhau, nhằm làm tối giản hóa giao tiếp giữa các ứng dụng đó và tăng hiệu quả tối đa cho việc tách ra các khối nhỏ hơn. Nhiệm vụ chính của một message broker tiếp nhận những message từ các ứng dụng và thực hiện một vài thao tác nào đó

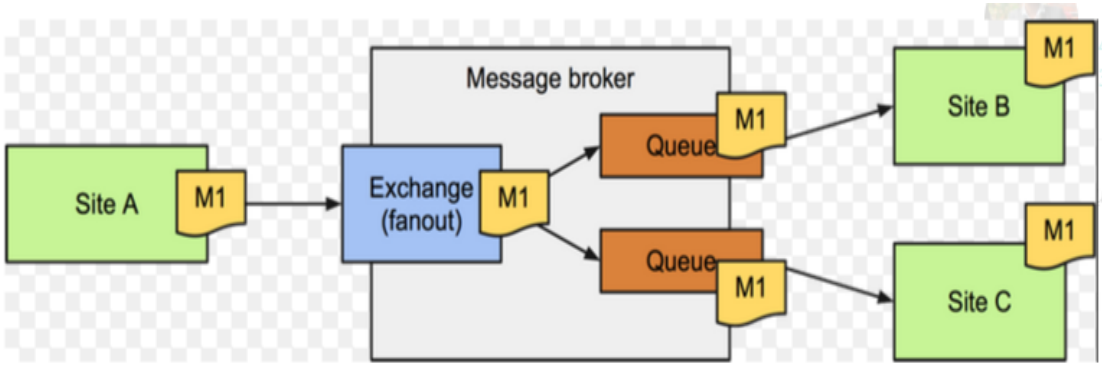
- RabbitMQ được cài đặt trên một máy chủ trong hệ thống, được gọi RabbitMQ service là nơi sẽ nhận và xử lý các message từ các client. Các client là các ứng dụng có thể nằm trên cùng một máy hoặc các máy trong hệ thống. RabbitMQ cung cấp các thư viện lập trình máy khách bằng các ngôn ngữ. khác nhau giúp nhà phát triển của thể tạo ra các module dùng để gửi (publisher) và nhận tin nhắn từ RabbitMQ service (consumer), có rất nhiều ngôn ngữ lập trình được hỗ trợ như C#, Ruby,Python,PHP,.. tuy nhiên trong đồ án này chúng ta sẽ sử dụng thư viện hỗ trợ cho ngôn ngữ lập trình java.

- RabbitMQ cũng cũng cấp một plugin cho phép hoạt động quản trị qua một web interface, điều này giúp cho việc quản trị trở lên thân thiện và trực quan hơn.Ngoài ra

tất cả các hoạt động quản trị qua web cũng có thể thực hiện qua một command line tool có tên là rabbitmqadmin. Tool này đi kèm management plugin.

- Tại sao nên sử dụng RabbitMQ

- Trong một hệ thống phân tán (distributed system) [10] sẽ bao gồm rất nhiều thành phần, để các thành phần có thể giao tiếp với nhau thì chúng sẽ phải nhận biết lẫn nhau,điều gây rắc rối cho việc lập trình, đồng thời cũng rất khó cho việc debug cũng bảo trì chương trình về sau. Phương pháp được ra đó là thay vì các thành phần kết nối trực tiếp với nhau khiến các thành phần phải nhận biết lẫn nhau thì có thể sử dụng một thành trung gian để liên kết đó chính là một message broker. Với sự có mặt của thành phần trung gian như message broker thì các thành phần trong hệ thống lúc này đóng vai trò như các publisher và consumer sẽ không cần phải nhận biết nhau mà vẫn có thể gửi thông tin qua lại. Các publisher chỉ cần gửi message vào các exchange trong broker, các exchange sẽ định tuyến để chuyển message đến đúng các queue và Consumer đăng ký sẽ nhận các message từ Queue.



Nguồn: kipalog.com

Hình 2.17.Mô hình Message broker với Exchange

- Mặc dù có những giải pháp khác cũng có thể giải quyết vấn đề này như sử dụng database với các bảng tạm thời để lưu các message tuy nhiên xét về hiệu năng thì không thể so sánh với message broker, khi nhu cầu trao đổi tin nhắn trở lên tăng mạnh thì kéo theo sẽ làm tăng việc tải dữ liệu của database điều này làm giảm hiệu suất đáng kể của hệ thống. Ngoài ra trong môi trường multithread database sẽ cần cơ chế lock để đảm bảo tính toàn vẹn của dữ liệu, việc này cũng sẽ làm giảm tốc độ xử lý.Trong khi đó nếu sử dụng message broker sẽ không phát sinh những vấn đề tương tự như trên.

- Ngoài ra trong một hệ thống phân tán thì các thành phần có thể được viết bằng các ngôn ngữ lập trình khác nhau nên việc giao tiếp với nhau thông qua message broker với message ở các định dạng kiểu dữ liệu chung sẽ giúp cho các thành phần có thể giao tiếp dễ dàng khi mà ngày nay RabbitMQ đã cung cấp các thư viện máy khách hỗ trợ rất

nhiều các ngôn ngữ khác nhau, đây là điều rất quan trọng giúp cho việc tích hợp message broker vào hệ thống linh hoạt hơn.

- Asynchronous (bất đồng bộ) : [11] Producer không thể biết khi nào message đến được consumer hay khi nào message được consumer xử lý xong.Đối với producer việc nó đẩy message đến broker là xong việc. Consumer sẽ lấy message khi nó muốn, đặc điểm này có thể tận dụng để xây dựng hệ thống lưu trữ và xử lý log.

- Flexible Routing : [11] message sẽ được định tuyến thông qua Exchange để có thể đến được với các Queue. RabbitMQ có hỗ trợ các loại Exchange thường dùng, chúng ta cũng có thể định nghĩ riêng các Exchange cho riêng mình.

- Lightweight: [11] mỗi một single instance của rabbitMQ chỉ chiếm khoảng 40MB ram điều này giúp tiết kiệm tài nguyên của máy.

- RabbitMQ ngoài hỗ trợ AMQP như một giao thức cốt lõi thì còn hỗ trợ các phương thức khác nữa như MQTT (Message Queuing Telemetry Transport), STOMP (Streaming Text Oriented Messaging Protocol)

- Cluster : Hỗ trợ gộp nhiều RabbitMQ instance vào một cluster, một queue được định nghĩa trên một instance có thể được truy xuất message từ các instance còn lại, điều này có thể được sử dụng để hỗ trợ load balancing

- High availability : cho phép failover sử dụng các mirror queue.

- Reliability: đây là cơ chế để đảm bảo message được nhận bởi consumer đã được xử lý, lưu trữ (persistence) message,.high availability, publisher confirm…

- Extensibility : cung cấp hệ thống plugin linh hoạt, dễ dàng tích hợp các plugin của third party. Ví dụ: plugin lưu message vào cơ sở dữ liệu.

- Cloud: dễ dàng triển khai với hạ tầng hiện có hoặc cloud

- Management & Monitoring : như đã giới thiệu ở trên RabbitMQ cung câp các HTTP API, command-line tool và UI để quản lý giám sát

- Tools support: Hoạt động với các công cụ CI/CD và có thể triển khai với Bosh, Chef, Docker,Puppet

2.4.3 Các khái niệm trong RabbitMQ

- Exchanges và các loại Exchanges

- Exchange là các thực thể AMQP nơi các tin nhắn được gửi. Các exchanges nhận một tin nhắn từ các publisher và định tuyến nó vào không, một hoặc nhiều hàng đợi. Thuật toán định tuyến được sử dụng phụ thuộc vào loại exchange và các quy tắc gọi là ràng buộc,RabbitMQ cung cấp 4 loại exchange cơ bản của giao thức AMQP:

Exchange type	Default pre-declared names
Direct exchange	(Empty string) and amq.direct
Fanout exchange	amq.fanout
Topic exchange	amq.topic
Headers exchange	amq.match (and amq.headers in RabbitMQ)

Nguồn: www.rabbitmq.com

Hình 2.18 Các kiểu Exchange

- Bên cạnh các loại Exchange, thì khi một exchange được khai báo thường có thể đi kèm một số thuộc tính giống với các thuộc tính đi kèm khai báo các queue:

+ Name

+ Durability (Thời gian sử dụng của exchange đó)

+ Auto-delete (exchange sẽ tự động được xóa bỏ khi queue cuối cùng xóa bỏ ràng buộc khỏi nó)

+ Argument (tùy chọn, được sử dụng bởi các plugin và các tính năng đặc biệt của broker)

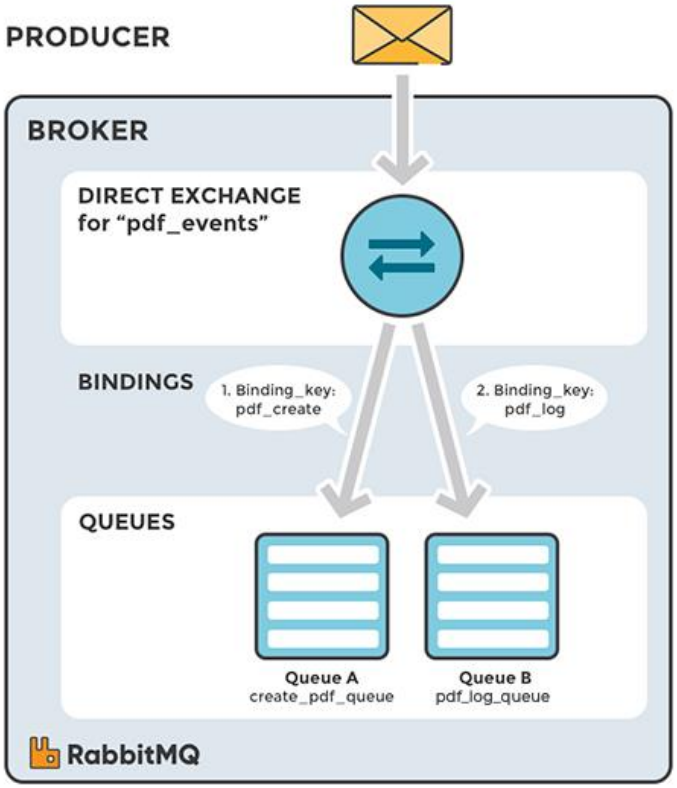
- Các exchange có thời gian tồn tại lâu dài hoặc chỉ trong một thời điểm cần thiết. Các exchange lâu dài được khởi tạo ngay khi broker được bật trong khi các exchange tạm thời thì không. Không phải trong tất cả trường hợp đều yêu cầu sử dụng các exchange lâu dài.

- Direct Exchange

- Trao đổi trực tiếp gửi tin nhắn đến hàng đợi dựa trên khóa định tuyến (Routing key).Khóa định tuyên là một thuộc tính thông báo được thêm vào tiêu đề (Header) của message bởi publisher. Có thể coi khóa định tuyến như là một địa chỉ mà exchange sử dụng để quyết định cách định tuyến message.Thông báo đi đến các hàng đợi với khóa liên kết khớp chính xác với khóa định tuyến của message.

- Direct exchange rất hữu ích để phân biệt các thông message được đẩy cho cùng một exchange bằng cách sử dụng một chuỗi định danh đơn giản [11]

- Trong trường hợp message được gửi tới direct exchange với routing_key không trùng khớp với bất kì một binding_key nào của các queue với exchange đó thì message sẽ bị hủy.



Nguồn: www.cloudamqp.com

Hình 2.19 Direct Exchange

- Default exchange

- Là một direct exchange được khai báo tự động từ trước và không có tên, thường được gọi bằng một chuỗi trống. Khi sử dụng default exchange message sẽ được gửi đến hàng đợi có tên giống với tên của routing_key. Mọi hàng đợi được tự động liên kết với exchange bằng khóa định tuyến giống như tên hàng đợi, hay lúc này routing_key = binding_key = queue_name.

- Topic Exchange

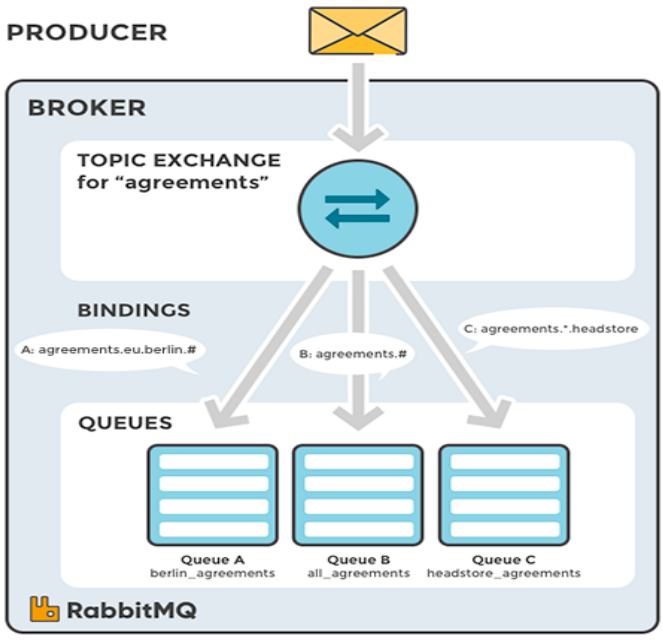
- Topic exchange định tuyến các message đến các hoàng đợi dựa trên việc so sánh các ký tự đại điện (matches wildcard) của routing_key và routing_pattern (mẫu định tuyến) được chỉ định bởi ràng buộc hàng đợi,các routing_pattern về chức năng sẽ giống như các binding_key của direct exchange mặc dù sẽ có chút khác biệt khi đây không phải một chuỗi cụ thể xác định và chỉ trùng khớp với một routing_key như binding_key. Message sẽ được chuyển đến một hoặc nhiều hàng đợi dựa vào sự phù hợp của routing_key và routing_pattern.

- Khóa định tuyến phải là một danh sách các từ được phân cách với nhau bởi một dấu chấm (.). Ví dụ như agreements.us và agreements.ey.stockholm.

- Trong khi đó các routing_patterns có thể chứa một dấu sao (*) để khớp với một vị trí cụ thể của routing_key.Ví dụ routing_pattern agreements.*.*.b.* sẽ được khớp với tất cả các routing key có 5 chuỗi cách nhau bởi dấu (.) và có chuỗi bắt đầu là agreement chuỗi thứ 4 là b. Ngoài ký tự dấu sao (*) ra thì còn có ký tự (#) có thể khớp hoặc không khớp ký tự dữ routing_key và routing_pattern tại vị trí có ký tự này. Ví dụ routing_pattern agreements.eu.berlin.# sẽ khớp với bất kỳ routing key nào bắt đầu bằng agreements.eu.berlin mà không cần quan tâm ký tự phía sau.

- Các consumer sẽ khai báo xem nó quan tâm đến những topic nào.Các consumer này sẽ khai báo các hàng đợi với các ràng buộc là những mẫu định tuyến nhất định (routing_pattern) với các exchange. Khí đó tất cả các message có khóa định tuyến routing_key phù hợp với kiểu định tuyến của những mẫu định tuyến đó sẽ được chuyển đến hàng đợi ở đó cho đến khi consumer lấy thông báo đó

- Một topic exchange mặc định được cung cấp có tên là “amq.topic”.



Nguồn: www.cloudamqp.com

Hình 2.20 Topic Exchange

- Một vài trường hợp sử dụng topic exchange :

+ Phân phối dữ liệu liên quan đến vị trí địa lý cụ thể

+ Xử lý tác vụ nền được thực hiện bởi nhiều thành phần trong hệ thống, mỗi thành phần có khả năng xử lý các tác cụ thể

+ Cập nhập các dữ liệu liên quan đến tài chính như giá cổ phiếu

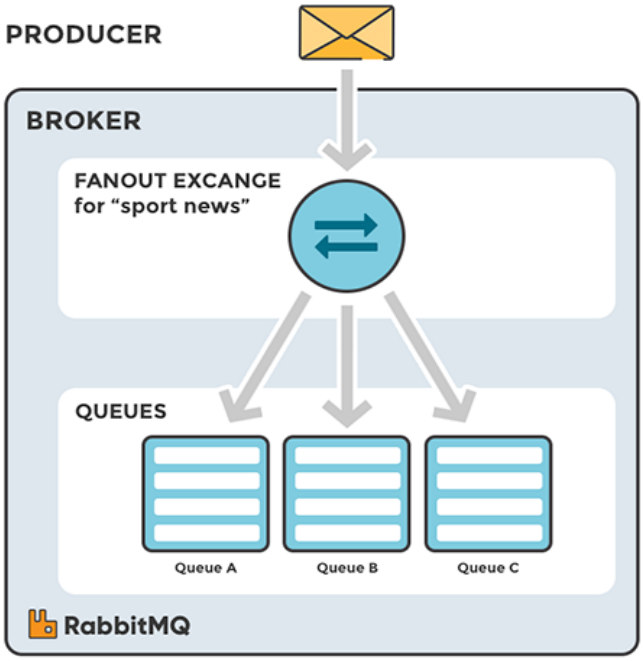
+ Cập nhập các loại tin tức đã được phân loại hoặc gắn thẻ như dữ liệu về một môn thể thao cụ thể trong một kỳ đại hội

+ Điều phối các dịch vụ thuộc các loại khác nhau trên đám mây

- Fanout Exchange

- Một fanout exchange sao chép và định tuyến các message đã nhận được đến tất cả các hàng đợi được liên kết với nó bất kể khóa định tuyến của message hay mẫu định tuyến của các ràng buộc như với topic exchange và direct exchange. Các khóa được cung cấp sẽ đơn giản bị bỏ qua

- Fanout Exchange có thể rất hữu ích khi cùng một thông điệp cần được gửi đến một hoặc nhiều hàng đợi với những consumer có thể xử lý cùng một thông điệp theo những cách khác nhau



www.cloudamqp.com

Hình 2.21 Fanout Exchange

- Như hình 2.21 cho thấy một ví dụ trong đó một message mà exchange nhận được sau đó nó sao chép và chuyển đến ba hàng đợi đã liên kết với nó. Chẳng hạn, đó có thể là các bản cập nhập về thể thao hoặc thời tiết sẽ được gửi đến từng thiết bị di động được kết nối khi điều đó xảy ra

- Các trường hợp có thể sử dụng Fanout Exchange:

+ Các trò chơi trực tuyến nhiều người chơi có thể sử dụng no để cập nhật bảng xếp hạng hoặc các sự kiện toàn cục khác

+ Các trang web tin tức thể thao có thể sử dụng trao đổi chủ đề để phân phối cập nhật điểm số cho khách hàng di động trong thời gian gần.

+ Hệ thống phân tán có thể phát các bản cập nhật trạng thái và cấu hình khác nhau

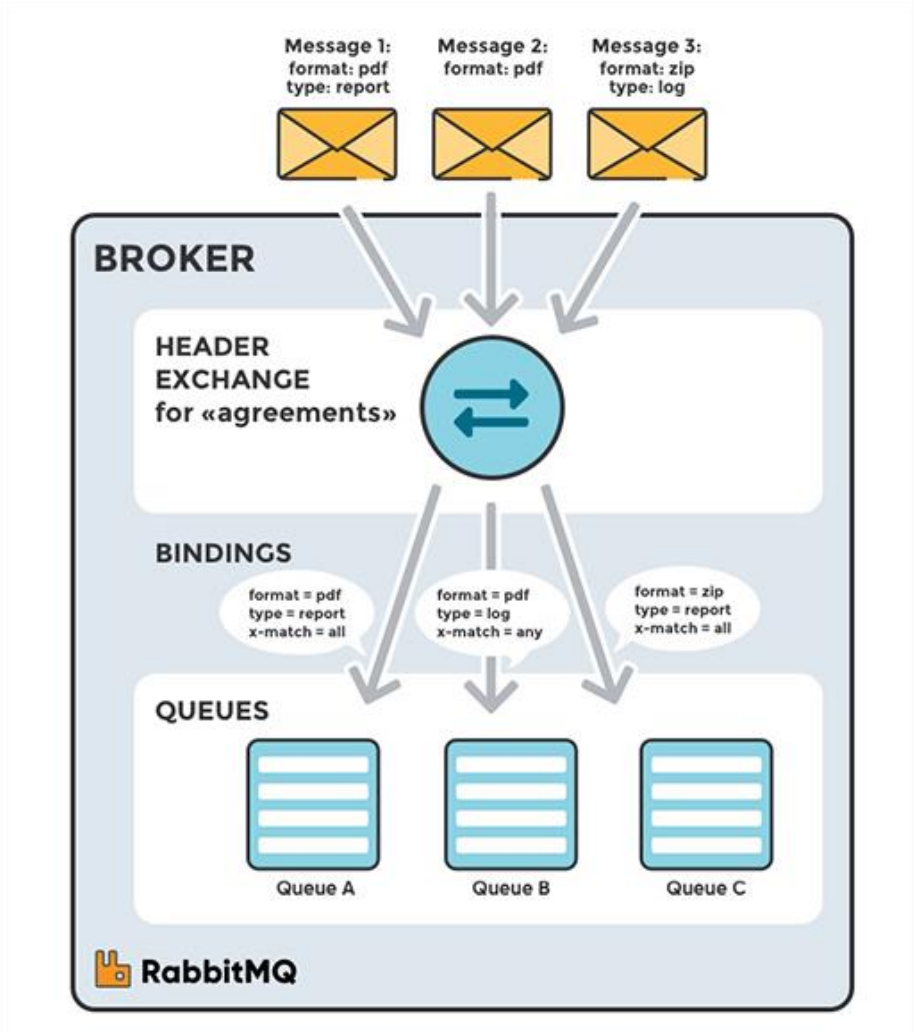
- Header Exchange

- Header Exchange định tuyến các message dựa trên các đối số chứa tiêu đề và các giá trị tùy chọn. header exchange rất giống với topic exchange nhưng không định tuyến message dựa trên giá trị tiêu đề và khóa định tuyến. Một thông báo được khớp nếu giá trị của tiêu đề bằng với giá trị được chỉ định ràng buộc.

- Đối số đặc biệt có tên " x-match " , được thêm vào ràng buộc giữa exchange và queue, chỉ định xem tất cả các tiêu đề phải được khớp hay chỉ cần một. Thuộc tính "x-match" này có thể có hai giá trị khác nhau là "any" hoặc "all" trong đó giá trị "all" là giá trị mặc định. Giá trị "all" có nghĩa là tất cả các cặp tiêu đề (key,value) đều phải khớp, trong khi giá trị "any" có nghĩa là chỉ cần ít nhất một cặp tiêu đều (key,value) khớp. Các tiêu đề có thể được tạo từ các kiểu dữ liệu khác nhau như số nguyên, băm thay vì chỉ có thể sử dụng chuỗi.

- Header Exchange có thể được sử dụng như một Direct Exchange trong đó khóa định tuyến sẽ được thay thế bằng các cặp giá trị tiêu đề, điều này rất hữu ích khi các cặp tiêu đề có thể được tạo nhiều loại dữ liệu khác nhau như đã nói ở trên thay vì chỉ dùng chuỗi như routing key của direct exchange.

- Lưu ý rằng các tiêu đề bắt đầu bằng chuỗi x- sẽ bị bỏ qua không được sử dụng để đánh giá các kết quả phù hợp.



Nguồn: www.cloudamqp.com

Hình 2.22 Header Exchange

- Dead Letter Exchange

- Dead letter là một tin nhắn không thể gửi đến người nhận. Dead letter Queue gọi tắt là DLQ là hàng đợi được sử dụng để chứa các tin nhắn này. Các tin nhắn không đến được với người nhận có thể do một trong các nguyên nhân sau :

+ Tin nhắn bị từ chối (rejected) bởi một Queue Exchange [13]

+ Message hết hạn (expire) do Time to Live [13]

+ Tin nhắn đến khi độ dài của hàng đợi đã vượt quá giới hạn. [13]

- Dead letter Exchange về cơ bản không phải là một loại Exchange riêng biệt mà nó có thể là một trong 4 loại exchange đã nêu như Direct, Fanout,Topic,Headers.

- Các Dead Letter Message sẽ được thêm một số thông tin hoặc không trước khi được gửi đến Dead Letter Exchange.

```
channel.exchangeDeclare("gpcoder.exchange.name", "direct");

Map<String, Object> args = new HashMap<String, Object>();
args.put("x-dead-letter-exchange", "gpcoder.exchange.name");
channel.queueDeclare("GPCoderQueue", false, false, false, args);
```

Nguồn: gpcoder.com

Hình 2.23 Ví dụ và định nghĩ Exchange và Queue

- Trong hình 2.23 là cách tạo ra một Dead Letter Exchange đồng thời tạo một Queue với tham số x-dead-letter-exchange. Điều này có nghĩa là khi queue chứa các tin nhắn chưa được gửi nhận được một message (dead letter) nó sẽ chuyển tin nhắn đó đến Dead Letter Exchange được chỉ định với key x-dead-letter-exchange

- Các message khi được chuyển đến Dead-Letter Message sẽ bị thay đổi Header.

- Một vài phương thức với Exchange
  - Exchange.declare
  - Exchange.declare-ok
  - Exchange.delete
  - Exchange.delete-ok

- Các phương thức trên tạo thành các cặp phương thức có quan hệ logic với nhau : exchange.declare và exchange.declare-ok, exchange.delete và exchange.delete-ok. Các phương thức đại diện cho các hành động "request" do client gửi và "responses" do broker gửi để phản hồi các "request".

- Ví dụ : client yêu cầu broker khai tạo một exchange với phương thức exchange.declare:

Nguồn: www.rabbitmq.com

Hình 2.24 Phương thức exchange.declare định nghĩa Exchange

- Hình 2.24 thể hiện phương thức exchange.declare đã yêu cầu broker tạo một exchange với một số tham số, chúng cho phép khách hàng chỉ định tên

exchange,loại,thời gian tồn tại…. Nếu thao tác trên thành công,broker sẽ sử dụng phương thức exchange.declare-ok.

Nguồn: www.rabbitmq.com

Hình 2.25 Broker dùng phương thức exchange.declare-ok

- Phương thức exchange.declare-ok không mang bất kỳ tham số nào ngoài trừ số kênh

- Với 2 phương thức exchange.delete và exchange.delete-ok cũng hoạt động theo mô hình tương tự.

- Hàng đợi (Queue)

- Hàng đợi trong RabbitMQ hay trong giao thức AMQP nói chung đều rất giống với hàng đợi trong các hệ thống xếp hàng nhiệm vụ hay tin nhắn khác đều dựa trên cấu trúc dữ liệu hàng đợi. Chúng lưu trữ các thông được các ứng dụng sử dụng, hàng đợi chia sẻ một số thuộc tính với các exchange nhưng cũng có một số thuộc tính bổ sung như :

+ Name : ứng dụng có thể chọn tên hàng đợi hoặc yêu cầu broker tạo tên cho cho hàng đợi được khai báo.Tên hàng đợi có thể dài tới 255 byte ký tự UTF8. RabbitMQ sẽ tạo ra một cái tên duy nhất cho một ứng dụng, để sử dụng tính năng này chỉ cần để một chuỗi trống làm tham số cho tên hàng đợi. Ứng dụng không thể khai báo tên hàng đợi bắt đầu bằng “amq”

+ Durable (thời gian tồn tại của queue): Hàng đợi có thể được khai báo với thời gian tồn tại lâu dài hoặc tạm thời.Các metadata của các hàng đợi có thời gian tồn tại lâu dài được lưu trong ổ đĩa trong khi với hàng đợi tạm thời thì chúng được lưu trong bộ nhớ khi có thể

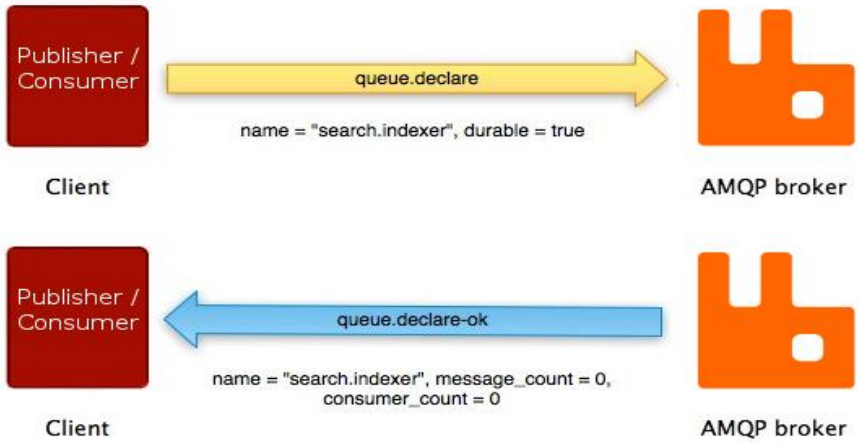
+ Exclusive (chỉ được sử dụng bởi một kết nối và hàng đợi sẽ bị xóa khi kết nối đó đóng)

+ Auto – delete (Hàng đợi sẽ bị xóa khi không còn consumer nào đăng ký)

+ Argument (các tùy chọn được sử dụng bởi các broker)

- Trước khi một hàng đợi có thể được sử dụng nó phải được khai báo. Khai báo một hàng đợi sẽ khiến nó được tạo nếu trước đó nó chưa tồn tại, khi hàng đợi đó đã tồn tại thì khai báo đó không có hiệu lực nếu các thuộc tính giống với hàng đợi đã tồn tại, nếu thuộc tính trong khai báo không giống với các thuộc tính hiện có một ngoại lệ có mã 406 (Precondition_failed) được đưa ra.

- Queue trong RabbitMQ cũng có một căp phương thức cơ bản là queue.declare và queue.declare-ok với mô hình hoạt động tương tự các phương thức của exchange.



Nguồn: www.rabbitmq.com

Hình 2.26 Hai phương thức khai báo Queue

- Bindings (các ràng buộc)

- Bindings là các quy tắc mà exchange sử dụng để định tuyến message đến với đúng hàng đợi. Để một exchange A khi nhận được message có thể định tuyến nó đến hàng đợi B thì B phải khai báo ràng buộc với E. Các ràng buộc có thể là các thuộc tính khóa định tuyến với tùy chọn khác nhau tùy vào loại exchange được sử dụng. Tác dụng của khóa định tuyến là xác định được các message được gửi đến exchange sẽ được chuyển đến hàng đợi nào trong số các hàng đợi đã liên kết với exchange đó. Nói cách khác thì các khóa định tuyến giống như bộ lọc.

- Ví dụ : Các hàng đợi sẽ đóng vai trò như các địa điểm trong một thành phố, các exchange đóng vai trò như bến xe , còn các ràng buộc sẽ như các tuyến đường dẫn từ bến xe đến các địa điểm.

- Việc có lớp chuyển hướng bằng binding này giúp loại bỏ các tình huống định tuyến vốn rất khó thực hiện bằng cách gửi trực tiếp message vào hàng đợi, đồng thời cũng giảm một số việc lặp đi lặp lại mà các nhà phát triển ứng dụng phải làm.

- Consumer

- Việc lưu trữ các message trong hàng đợi là vô ích nếu các ứng dụng không thể lấy và sử dụng chúng,RabbitMQ cung cấp hai cách để ứng dụng có thể nhận được các message:

+ Đăng ký nhận message gửi đến (push API), đây là tùy chọn thường được sử dụng, khi message được gửi đến queue thì ứng dụng cũng sẽ tự động nhận được và bắt đầu tiến trình xử lý.

+ Kéo message về ứng dụng (pull API), đó là ứng dụng có thể kiểm tra xem có message được gửi đến queue hay không, nếu có thì có thể tiến hành kéo message về.

- Với tùy chọn sử dụng push API ứng dụng cần phải thực hiện việc đăng ký một consumer cụ thể hoặc đơn giản là đăng ký một hàng đợi cụ thể. Có thể đăng ký nhiều consumer cho một hàng đợi hoặc đăng ký một consumer độc quyền việc này sẽ ngăn chặn tất cả các consumer khác khỏi hàng đợi.

- Mỗi consumer có một định danh được gọi là consumer tag. Nó có thể được sử dụng để hủy đăng ký nhận tin nhắn. Consumer tag chỉ là một chuỗi ký tự.

- Connection (Kết Nối)

- Như RabbitMQ sử dụng giao thức AMQP đây là một giao thức tầng ứng dụng sử dụng kết nối TCP để truyền tin tin cậy giữ client (publisher và consumer) và broker . Kết nối sử dụng xác thực và có thể được đảm bảo bằng TLS.

Nguồn: www.cloudamqp.com

Hình 2.27 Tạo kết nối TCP giữa App client và broker

- Channel (Kênh)

- Các kết nối có thể tạo các kênh thông qua một kết TCP, hay có nghĩa là các ứng dụng có thể mở ra một " lightweight connections " trên một connection (kết nối). Các " lightweight connections " được gọi là các kênh. Mỗi connection có thể duy trì một tập hợp các kênh

- Nhiều ứng dụng cần có nhiều kết với broker và thay vì mở ra nhiều kết nối thực sự, một ứng dụng có thể sử dụng lại một kết nối, để làm được việc này thì ứng dụng sẽ tạo các kênh dựa trên kết nối đó,khi sử dụng có thể xóa các kênh đó đi. Điều này tránh được vấn đề không mong muốn khi phải mở nhiều kết nối TCP cùng lúc gây ra tiêu tốn tài nguyên hệ thống, quá trình bắt tay cho một kết nối khá phức tạp khi cần 7 gói tin TCP trở nên nếu sử dụng TLS.

- Mỗi kênh hoạt động như một kết nối ảo bên trong kết nối TCP.Một kênh sẽ sử dụng lại kết nối mà không cần ủy quyền lại hay mở một luồng TCP mới,việc này cho phép các ứng dụng sử dụng tài nguyên hiệu quả hơn.

- Mọi hoạt động liên quan đến giao thức AMQP đều xảy ra trên một kênh.

Nguồn: www.cloudamqp.com

Hình 2.28 Tạo channel trên kết TCP

- Kết nối được tạo bằng các mở kết nối TCP đến máy chủ đích. Máy khách phân tách tên máy chủ thành một hoặc nhiều địa chỉ IP trước khi gửi bắt tay, sau đó máy chủ xác thực máy khách

- Để gửi tin nhắn hoặc quản lý hàng đợi, kết nối được tạo với broker trước khi thiết lập kênh thông qua máy khách. Kênh đóng gói các message và xử lý các bước giáo thức. Các máy khách gửi tin nhắn thông quan phương thức basic_publish của kênh. Việc tạo và duy trì hàng đợi cũng thông qua kênh với các lệnh queue.create hay exchange.create. Việc đóng kết nối sẽ đóng tất cả các kênh được tạo dựa trên kết nối đó.

- Các nhà phát triển RabbitMQ khuyến cáo nên sử dụng và duy trì kết nối lâu dài thay vì sử dụng nhiều kết nối vì điều này có thể gây quá tải máy chủ RabbitMQ. Mỗi quy trình chỉ nên sử dụng một kết nối TCP và sử dụng nhiều nhiều kênh trong kết nối cho các luồng hoạt động khác nhau.

- Đối với các ứng dụng bao gồm cả Publisher và Consumer thì có thể tách biệt kết nối của hai thành phần này,vì RabbitMQ có thể gây áp lực ngược lên kết nối TCP khi các Publisher gửi quá nhiều message để xử lý. Nếu sử dụng trên cùng một kết nối

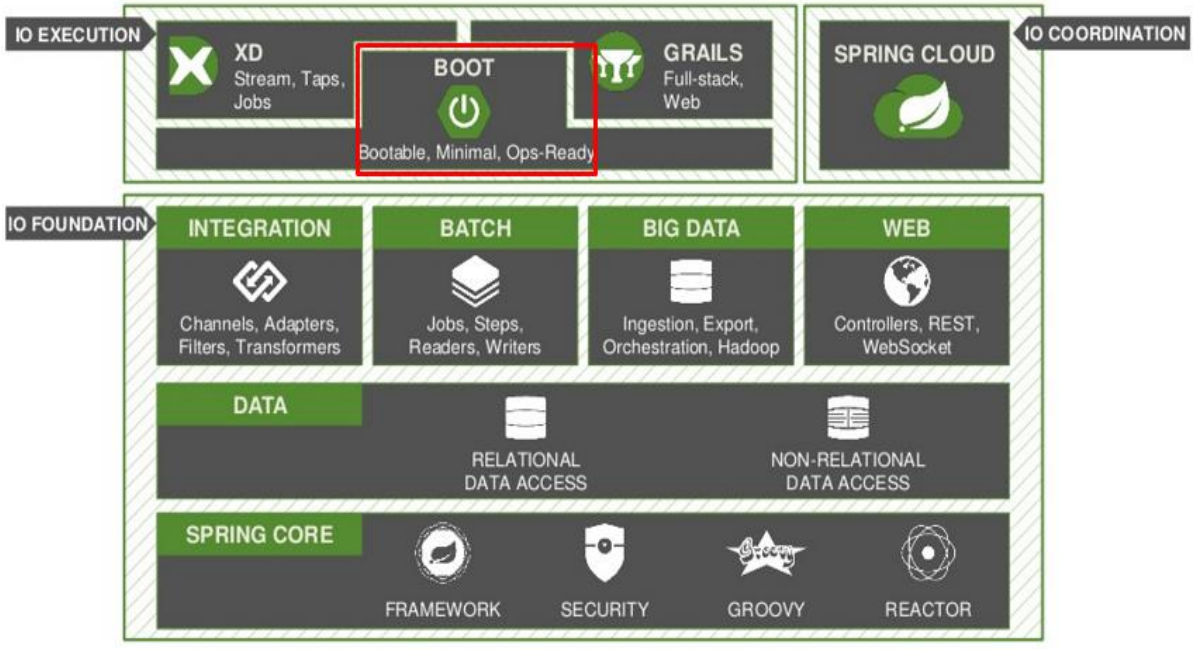
TCP,máy chủ có thể không nhận được thông báo xác nhận từ máy khách do đánh ảnh hưởng đến hiệu suất của hệ thống.

- Virtual Hosts (Máy chủ ảo)

- Để một broker duy nhất có thể lưu trữ nhiều " môi trường " biệt lập là các thực thể amqp như exchange,queue…,thì rabbitMQ hoạt đông trên một máy chủ ảo tương tự như các máy chủ ảo được sử dụng bởi nhiều máy chủ web phổ biến và cung cấp các một trường hoàn toàn biệt lập cho các thực thể AMQP có thể hoạt động. Giao thức máy khách sẽ chỉ định vhosts mà máy khách muốn sử dụng thông qua quá trình tạo kết nối.

2.5 Spring – boot

Spring Boot là một module của hệ sinh thái spring Framework, cung cấp tính năng RAD (Rapid Application Development) một cách thức để phát triển ứng dụng một cách nhanh chóng. Điều giúp các lập trình viên đơn giản hóa quá trình lập trình một ứng dụng với spring và chỉ cần tập trung vào việc phát triển các chức năng nghiệp vụ cho ứng dụng.



Nguồn: spring.io

Hình 2.29 Spring IO platform

- Một số tính năng nổi bật của spring-boot:

- Tạo các ứng dụng spring độc lập

- Nhúng trực tiếp Tomcat, Jetty, hoặc Undertow (Không cần phải deploy ra file war)

- Các starter dependency giúp việc cấu hình Maven đơn giản hơn

- Tự động cấu hình Spring khi cần thiết

- Không sinh code cấu hình và không yêu cầu phải cấu hình bằng XML

- Dễ dàng tương tác với các module cần thiết khác trong hệ sinh thái spring để có thể xây dựng một ứng dụng hoàn chỉnh như Spring JDBC,Spring ORM, Spring DATA, Spring Security..

- Cung cấp sẵn công cụ CLI (Command Line Interface) để phát triển và test các ứng dụng Spring

Có sẵn nhiều plugin để làm việc với các cơ sở dữ liệu nhúng và cả cơ sở dữ liệu lưu trữ trên bộ nhớ.

CHƯƠNG 3. TRIỂN KHAI GIẢI PHÁP

3.1 Đưa ra giải pháp

3.1.1 Ý tưởng ban đầu

Bài toán cần giải quyết của đồ án là có thể thực hiện việc quản lý và điều khiển các công cụ IDS Suricata được cài đặt trong hệ thống một cách tập trung thay vì phải thao tác trực tiếp các công cụ này tại các thiết bị mà chúng được cài đặt.

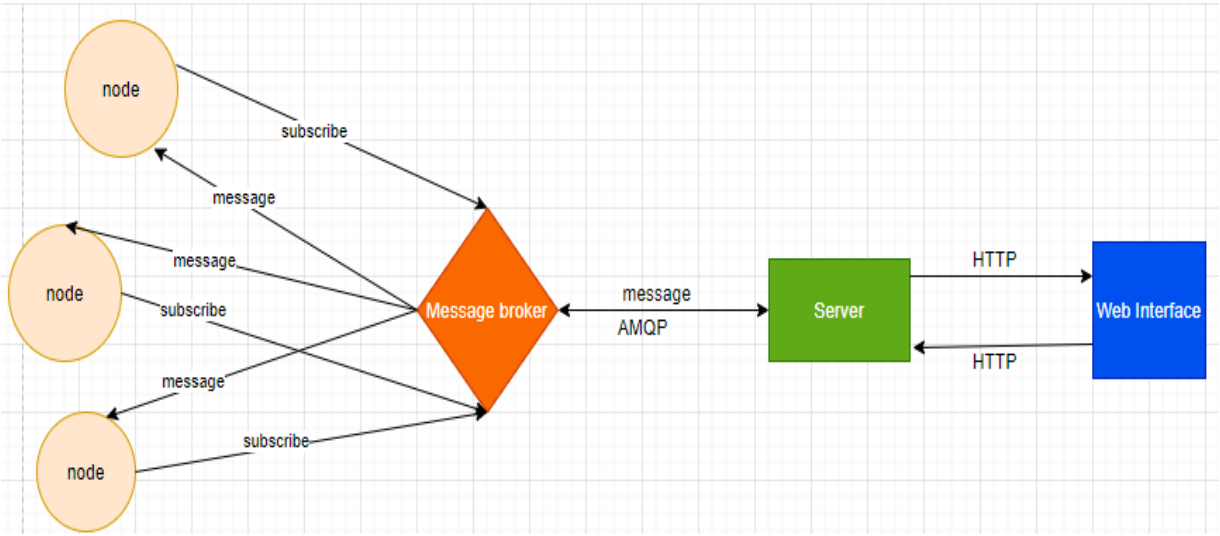
Từ yêu cầu bài toán nêu trên giải pháp được đưa ra đó là xây dựng một hệ thống cho phép các quản trị viên có thể thực hiện các thao tác điều khiển đến các công cụ Suricata được cài đặt từ xa một cách tập trung thông qua một giao diện web.

Hệ thống sẽ gồm 3 bộ phần chính:

+ Một công cụ tự phát triển được cài đặt trên cùng thiết bị với công cụ Suricata và tương tác trực tiếp với công cụ này được gọi là node

+ Một message broker có nhiệm vụ làm trung gian truyền tin giữa server và các node

+ Một web server là nơi các quản trị viên thực hiện các thao tác điều khiển đồng thời nhận và xử lý output từ các node



Hình 3.1 Mô hình hoạt động của giải pháp

- Khi triển khai mô hình hệ thống các node sẽ được cài đặt trên cùng một thiết bị với công cụ Suricata để có thể giao tiếp trực tiếp và thực hiện các lệnh điều khiển với công cụ này

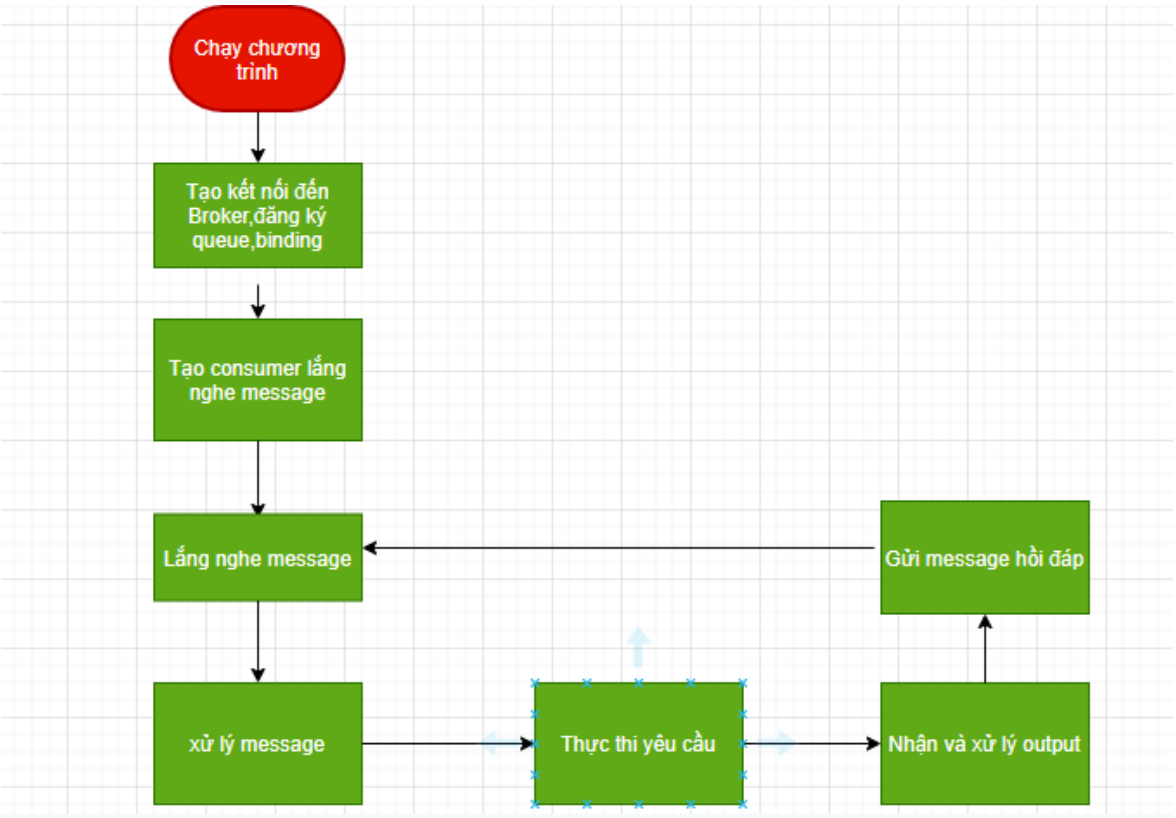
- Message Broker cùng Web server sẽ được cài đặt trên cùng một máy chủ, quản trị viên sẽ tương tác thông qua một giao diện web, các yêu cầu sẽ được gửi về web server để xử lý sau đó được truyền tới các node thông qua message broker để được thực thi

3.1.2 Mô tả các bộ phần trong hệ thống

- **Node**

- Các node sẽ được xây dựng dựa trên ngôn ngữ lập trình java một ngôn ngữ lập trình phổ biến có thể chạy trên nhiều nền tảng khác nhau. Các node sẽ tương tác với công cụ Suricata thông qua việc thực thi các command system với các command của Suricata được cung cấp bởi nhà phát triển.

- Các node sử dụng một bộ thư viện Client do RabbitMQ cung cấp để xây dựng các publisher và consumer sử dụng giao thức AMQP 1-0-9 để có thể giao tiếp với message broker. Thông qua các publish và consumer này các node sẽ nhận các chỉ thị từ quản trị viên, xử lý và thực thi chúng sau đó nhận output từ việc thực thi và gửi lại.

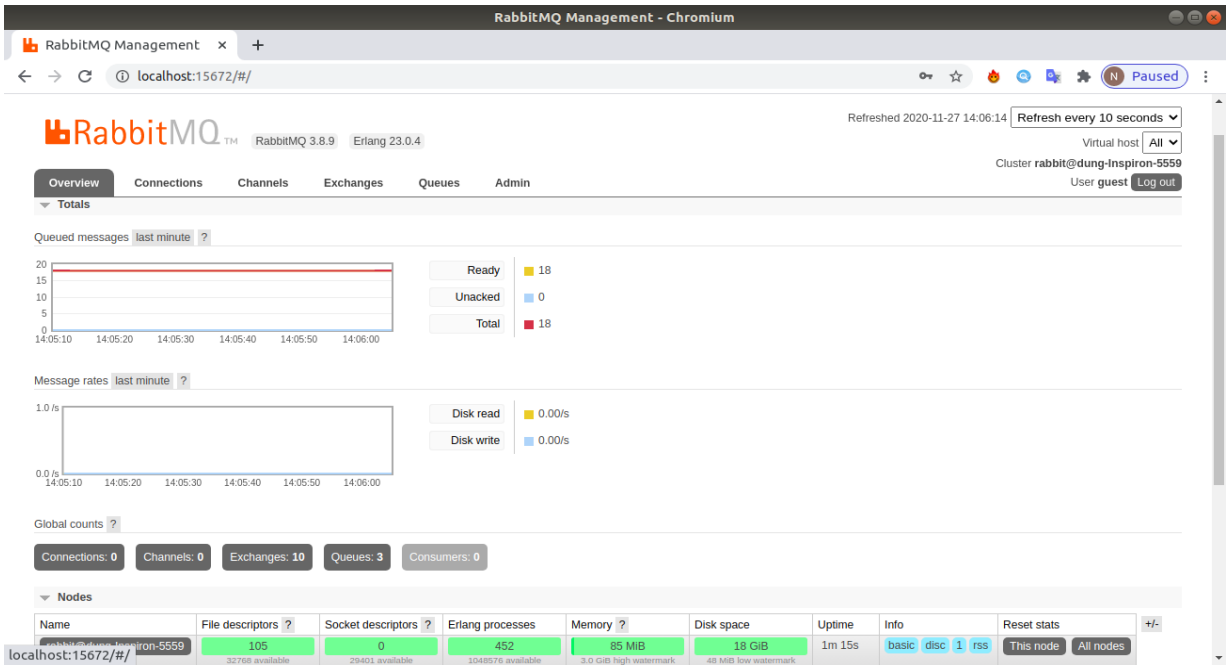


Hình 3.2 Luồng Hoạt động của Node

- Message Broker

- Message broker được sử dụng trong hệ thống là RabbitMQ đây là một message broker phổ biến và dễ sử dụng, hỗ trợ đầy đủ các tính năng các tính năng truyền tin cần thiết, đồng thời cung cấp bộ thư viện để xây dựng các client với rất nhiều các ngôn ngữ lập trình khác nhau.

- RabbitMQ còn có giao diện người dùng có để hỗ trợ việc quản lý giám sát các nút và các cụm RabbitMQ, dưới sự hỗ trợ của Management plugin định kỳ thu thập và tổng hợp dữ liệu của hệ thống hiển thị cho người vận hành giúp cho việc quản lý trở nên dễ dàng hơn.



Hình 3.3 Giao diện quản lý RabbitMQ

- Việc sử dụng message broker thay vì các cách thức truyền tin thông thường khác là để tận dụng ưu điểm của phương pháp truyền tin này, hệ thống được xây dựng nhằm giải quyết vấn đề tương tác và điều khiển các công cụ Suricata được cài đặt một cách phân tán trong hệ thống mạng với các địa chỉ khác nhau. Các Host IPS như Suricata thường không có địa chỉ IP tĩnh vì vậy để có thể truyền tin bằng các phương pháp khác như RestAPI trở nên không khả thi, khi server cần xác định được địa chỉ chính xác.

- Với việc sử dụng một message broker làm trung gian truyền tin cho toàn hệ thống, chúng ta sẽ không cần quan tâm đến việc xác định địa chỉ của các node, mà chỉ cần một địa chỉ xác định cho broker, khi đó các node có thể subscribe broker và nhận tin. Việc định tuyến các message để xác định các message cần tới node nào cũng trở nên dễ dàng hơn với các tính năng có sẵn của một broker như exchange, binding, routing key….

- Web Server

- Web server là nơi quản trị viên trực tiếp thực thiện các thao tác để điều khiển các node. Web server trong hệ thống được xây dựng với framework spring-boot, một framework phổ biến để xây dựng các hệ thống website. Ngoài ra framework cũng hỗ trợ các thư viện để có thể tương tác truyền tin với một AMQP message broker như RabbitMQ

- Các quản trị viên sẽ thực hiện các thao tác thông quan giao diện người dùng của web đưa ra các chỉ thị. Backend web sẽ nhận các thông tin, tiến hành xử lý tạo message để gửi tới message broker. Web server cũng sẽ nhận lại output từ các node thông qua broker đồng thời tiến hành hiển thị chúng tới quản trị viên.

3.1.3 Các chức năng chính của hệ thống

- Nhiệm vụ chính của hệ thống là có thể giúp cho quản trị viên tiến hành các thao tác từ xa để quản lý và điều khiển đối với các công cụ Suricata đã được cài đặt và tích hợp cùng các công cụ hệ thống.

- Các tác vụ chính có thể thực hiện với công cụ Suricata bao gồm:

+ Khởi động Suricata

+ Tắt Suricata

+ Tiến hành thêm rule mới do quản trị viên viết

+ Xem log file.

- Chương trình có thể hoạt động trên các máy linux có cài đặt máy ảo java (java virtual machine).

3.1.4 So sánh với công cụ Owlh

- Owlh là gì

Owlh là một dự án mã nguồn mở hỗ trợ việc triển khai và quản lý các hệ thống phát hiện xâm nhập trong đó bao gồm cả Suricata. Các thành phần của Owlh sẽ bao gồm 3 phần chính :

+Owlh node : Là single box dùng để chạy các hệ thống NIDS mà mặc định là Suricata và zeek, node sẽ thực hiện việc phân tích dữ liệu cùng với Suricata và zeek, đồng thời thực hiện các chức năng điều khiển và cấu hình cho hai hệ thống này. Node cũng cung cấp các RESTful API để hỗ trợ việc trao đổi thông tin.

+Owlh master: Công cụ bao gồm các API để thực hiện các chức năng quản lý đối với các node. Không có giới hạn về số node mà một master có thể quản lý. Sau khi các node được đăng ký người dùng có thể cấu hình như giao diện lắng nghe, các bộ rules được triển khai tại mỗi node. Các kết nối được bắt đầu từ master đến node. Là một điểm quản lý tập trung, nó cho phép thu thập lưu lượng truy cập từ các nút từ xa.

+Owlh UI: là một giao diện đồ họa cung cấp khả năng truy cập và thao tác trực quan. Người dùng sẽ thông qua UI để làm việc với Master thực hiện các chức năng với các node thông qua các API của chúng.

Người dùng có thể triển khai các thành phần của Owlh thông qua một trình cài đặt là Owlh Installer giúp đỡ cài đặt các thành phần. Người dùng cần cài đặt cấu hình Master và UI trước, tiếp theo sẽ cài đặt và cấu hình các node. Việc tạo các node để quản lý ở Master được thực hiện cuối cùng vì cần có các thông tin như IP và port của các Owlh Node được cài đặt trước đó,trong quá trình làm việc Master sẽ kết nối tới các Owlh Node thông qua các thông tin này.

- So sánh giữa giải pháp của đồ án và Owlh

- Mục tiêu chính của đồ án này có sự tương đồng với Owlh khi hướng tới việc có thể giúp người dùng thực hiện các chức năng quản lý các công cụ IDS từ xa đồng thời có thể quản lý được nhiều công cụ một lúc.

- Sự khác nhau chủ yếu nằm ở mô hình,cách hoạt động cũng như triển khai.

+ Về mô hình trong khi Owlh coi các IDS là một phần trong mô hình của mình, các IDS này là một bộ phận của một Owlh Node thì giải pháp của đồ án chỉ coi IDS mà cụ thể là Suricata là một đối tượng để giao tiếp và điều khiển, Suricata hoàn toàn là một chương trình chạy độc lập. Ngoài ra trong mô hình của giải pháp còn sử dụng thêm một thành phần là Message Broker

+ Về cách triển khai thì Owlh tiến hành cài đặt các Owlh Node trong đó có cả các IDS, đồng thời chỉ có thể tiến hành tạo một đối tượng node tại Master để quản lý khi Owlh Node đã được cài đặt trước đó vì cần có các thông tin liên quan đến IP và Port của thiết bị nơi Owlh Node được cài đặt. Trong khi giải pháp của đồ án tiến hành tạo các

node tại web trước và dùng các thông tin nhận được để cấu hình cho các Node thực tế được cài đặt trên thiết bị

+ Cách thức hoạt động cũng có sự khác biệt khi việc thực hiện giao tiếp giữa Owlh Node và Owlh Master là thông qua API, điều này bắt buộc Owlh Master phải có địa chỉ Ip của Owlh Node hay nói cách khác các Node này cần có static IP. Trong khi đó với giải pháp của đồ án thì chỉ cần một static IP cho máy chủ nơi Message Broker và web server được cài đặt, các node sẽ lưu địa chỉ này để có tiến hành đăng ký nhận và gửi message.

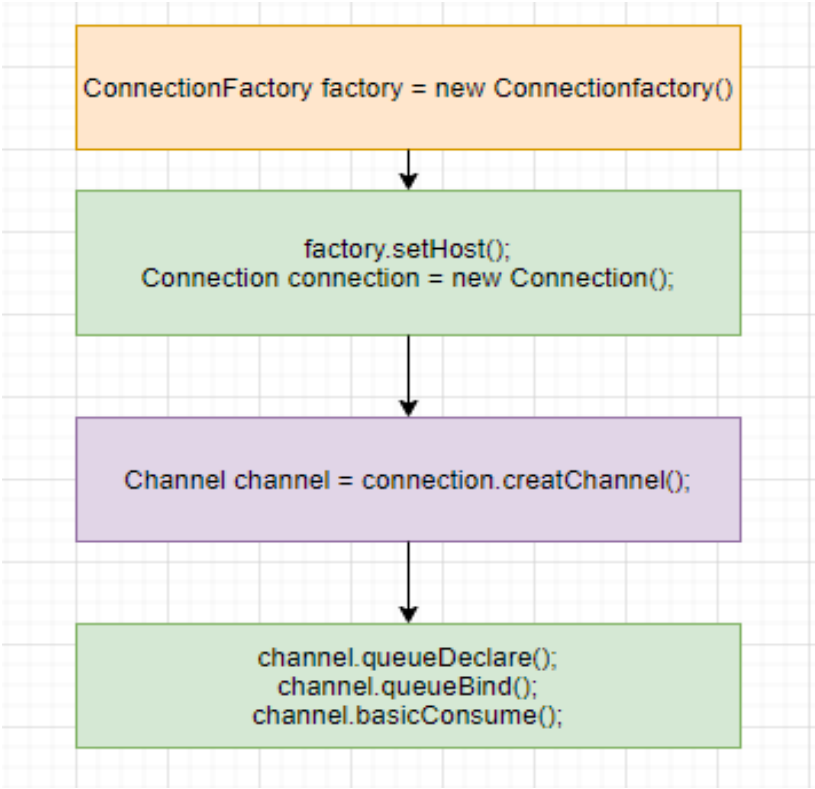
Việc áp dụng broker message vào làm một công cụ truyền tin của giải pháp trong đồ án giúp đơn giản hóa vấn đề kết nối giữa các node và trình quản lý nơi quản trị viên làm việc.

3.2 Triển khai xây dựng chương trình

- Lập trình Node với thư viện amqp-client

- Ngôn ngữ lập trình được sử dụng để phát triển node là java. Nhiệm vụ của các node trước tiên là có thể nhận và gửi được các message đến với broker, để thực hiện việc này chúng ta sẽ sử dụng thư viện client do RabbitMQ cung cấp cho ngôn ngữ java như amqp-client-5-7-1.jar,slf4j-api…

- Để nhận được message từ broker, node cần có một consumer luôn luôn hoạt động để có thể nhận được các message từ broker ngay lập tức. Lần lượt import các Class từ thư viện amqp như com.rabbitmq.client.Channel, com.rabbitmq.client.Connection, com.rabbitmq.client.ConnectionFactory, com.rabbitmq.client.DeliverCallback để có thể khởi tạo các đối tượng và sử dụng các phương thức cần thiết.



Hình 3.4 các bước lập trình một consumer

- Ngoài ra chúng ta cũng cần 1 phương thức triển khai của functional Interface DeliverCallback làm tham số cho phương thức channel.basicConsume. Bản triển khai này chính là nơi message được xử lý sau khi được nhận.

- Tiếp theo chương trình cần thực thi các lệnh điều khiển nhận được từ message, để thực hiện được các lệnh điều khiển này chương trình sử dụng lớp ProcessBuilder nhằm khởi chạy một tiến trình chạy câu lệnh khởi động Suricata.

- Cuối cùng sau khi lệnh điều khiển được thực thi các output sẽ được xử lý, lúc này, chương trình cần có một publisher để có thể tiến hành xử lý output tạo message và gửi trả lại cho broker thông qua phương thức channel.basicPublish() với các tham số là message và các tham số định tuyến cần thiết. Message trả về được định dạng ở dạng json với các giá trị bao gồm, tên node và giá trị output của câu lệnh được thực thi.

- Message sẽ được broker tiếp nhận và chuyển đến server để xử lý trước khi hiển thị kết quả lên giao diện web cho quản trị viên.

- Web server với Spring-boot

- Nhiệm vụ của web server là nhận lệnh của quản trị viên thông qua các HTTP request từ giao diện web, các controller tiến hành xử lý các yêu cầu sau đó gửi các message đến broker, sau đó tiến hành nhận message phản hồi để thông báo kết quả đến cho người dùng.

- Với spring-boot chúng ra có thể sử dụng dependency được cung cấp để có thể giao tiếp với broker RabbitMQ, cũng như tạo các thực thể AMQP.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
```

Hình 3.12 Spring-boot-starter-amqp

- Server cũng cần khởi tạo một số thành phần để giao tiếp với broker và gửi message bao gồm queue để nhận thông tin message từ các Node, exchange để thực hiện việc gửi và định tuyến tới các queue của các node.

- Exchange được sử dụng tại server đó topic exchange để gửi message. Với kiểu exchange này cho phép gửi tin nhắn đến nhiều node có các ràng buộc phù hợp với routing key của message. Điều này đặc biệt hiệu quả trong một số trường hợp, như khi muốn tắt toàn bộ số công cụ Suricata đang hoạt động, update rule v.v.v..

- Xây dựng một trình lắng nghe với class SimpleMessageListenerContainer được cung cấp thư viện amqp để lấy các message từ queue. Để khởi tạo một trình lắng nghe message bằng SimpleMessageListenerContainer cần có các tham số là một đối tượng MessageListenerAdapter có nhiệm vụ chỉ định class và function sẽ xử lý message và một đối tượng ConnectionFactory để quản lý kết nối.

- Framework spring hỗ trợ việc khởi tạo các đối tượng thực thể amqp này một cách nhanh chóng ngay sau khi chương trình của server bắt đầu được chạy thông qua việc tạo các Bean bằng annotation @bean trả về đối tượng của các thực thể nói trên. Việc khởi tạo bằng @bean cho phép các đối tượng được khởi tạo có thể sử dụng nhiều lần ở bất kì đâu trong chương trình.

- Để gửi message đến broker cũng cần khởi tạo một trình message, dependency spring-boot-starter-amqp cung cấp sẵn một bean của class AmqpTemplate với phương thức convertAndSend() để gửi message. Sử dụng bean này xây dựng một Service để sử dụng trong các controller.

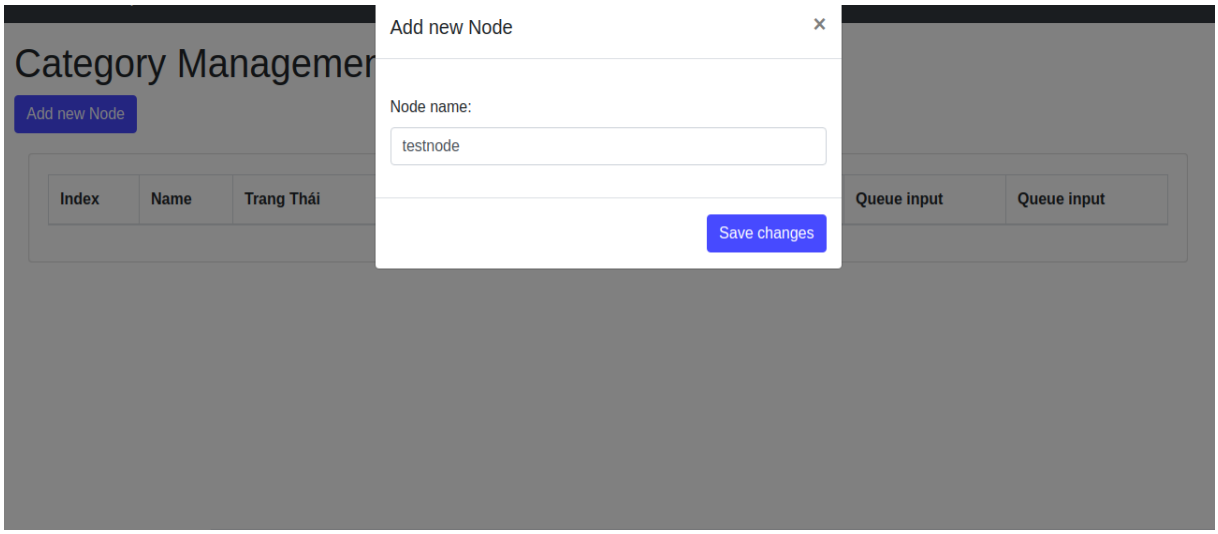
- Cuối cùng tiến hành xây dựng các module khác như controller để xử lý request từ frontend web, các model và repository để tương tác với cơ sở dữ liệu, các service hỗ trợ các tác vụ xử lý yêu cầu và dữ liệu.

- Web server cũng cần một giao diện web đơn giản để hiện thị các thông tin cũng như để quản trị viên có thể tiến hành các thao tác điều khiển một cách trực quan.

CHƯƠNG 4. THỰC THI VÀ KẾT QUẢ

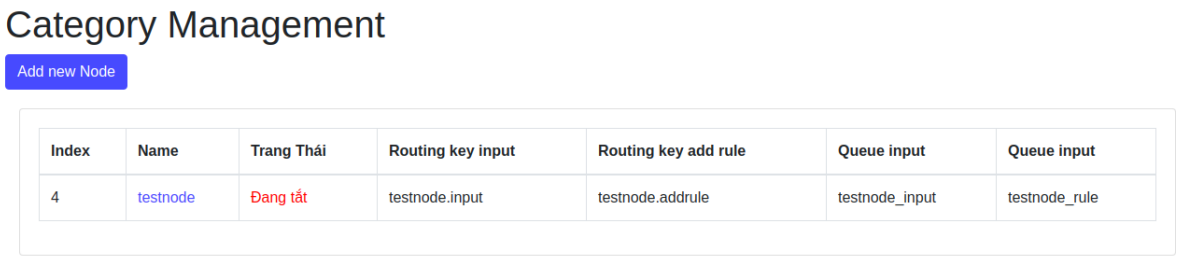
4.1 Thực thi chương trình.

- Khởi động và truy cập server web
  - Sau khi khởi động web server tiến hành truy cập vào giao diện web và thực hiện đăng ký một node..



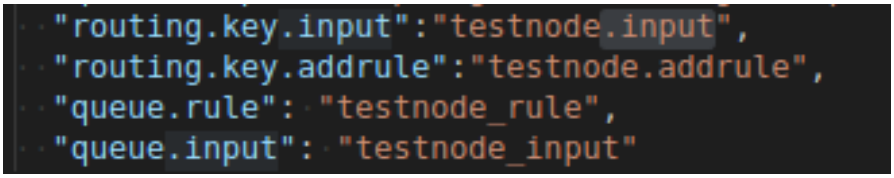
Hình 4.1 Đăng ký một node

- Sau khi node được tạo thành công, tiến hành lấy các thông tin cần thiết để config trước khi khởi chạy chương trình node.



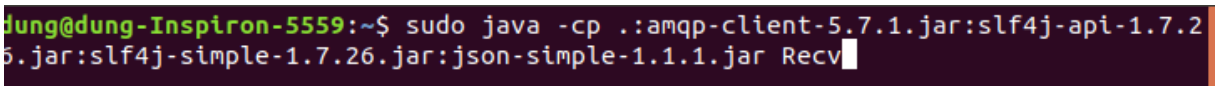
Hình 4.2 Node khi được đăng ký thành công

- Sử dụng các thông tin được cung cấp sau khi tạo một node trên web server để config vào file config của chương trình node. Đây là các thông tin dùng để thực hiện việc trao đổi message giữa server và node thông qua broker



Hình 4.3 Config node theo thông tin đăng ký

- Chạy chương trình node



Hình 4.4 Command line để chạy một node

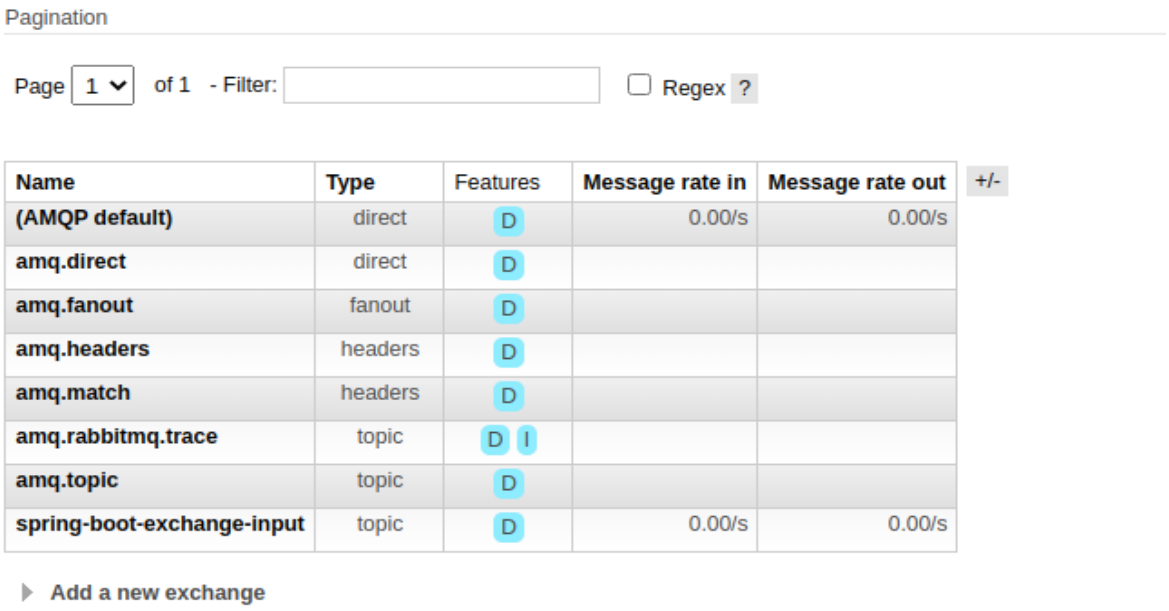
- Chương trình của node cần được thực thi dưới quyền sudo, điều này cho phép các tác vụ về sau của chương trình có thể được thực thi mà không cần quan tâm tới việc cấp quyền cho các tác vụ đặc biệt.

- Chương trình cũng cần có một số thư viện đi kèm vì vậy cần chạy với một số classpath của các thư viện.

4.2 Kết quả thực thi

4.2.1 Kiểm tra kết quả khởi tạo chương trình

- Sau khi cả web server và chương trình node đều đã được chạy tiến hành kiểm tra các thực thể AMQP đã được tạo đúng như yêu cầu hay chưa. Truy cập vào RabbitMQ Management Interface để kiểm tra.



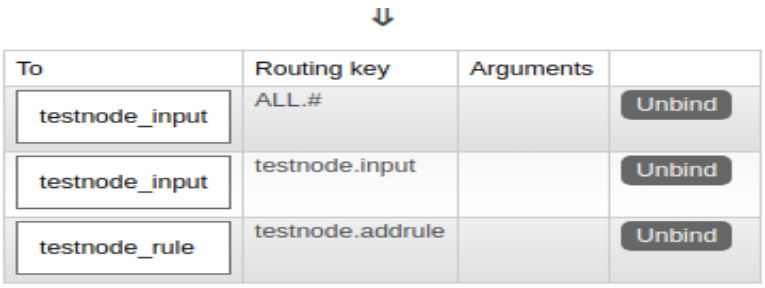
Hình 4.5 Exchange được tạo thành công

- Từ hình 4.5 đã thấy ngoài các exchange mặc định được khởi tạo bởi RabbitMQ thì một exchange khác với tên gọi spring-boot-exchange-input đã được khởi tạo. Đây sẽ là exchange để server có thể tiến hành gửi message đến các node.

Overview				Messages			Message rates			+/-
Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
spring-boot-exchange-output	classic		idle	0	0	0				
testnode_input	classic	D	idle	0	0	0				
testnode_rule	classic		idle	0	0	0	0.00/s	0.00/s	0.00/s	

Hình 4.6 Các queue được tạo thành công

- Các queue dùng để nhận message từ server gửi node cũng như ngược lại cũng đã được khởi tạo. Trong đó Spring-boot-exchange-output là queue để server nhận message phản hồi từ node, queue này sẽ tự động được tạo ràng buộc với direct exchange mặc định của RabbitMQ.

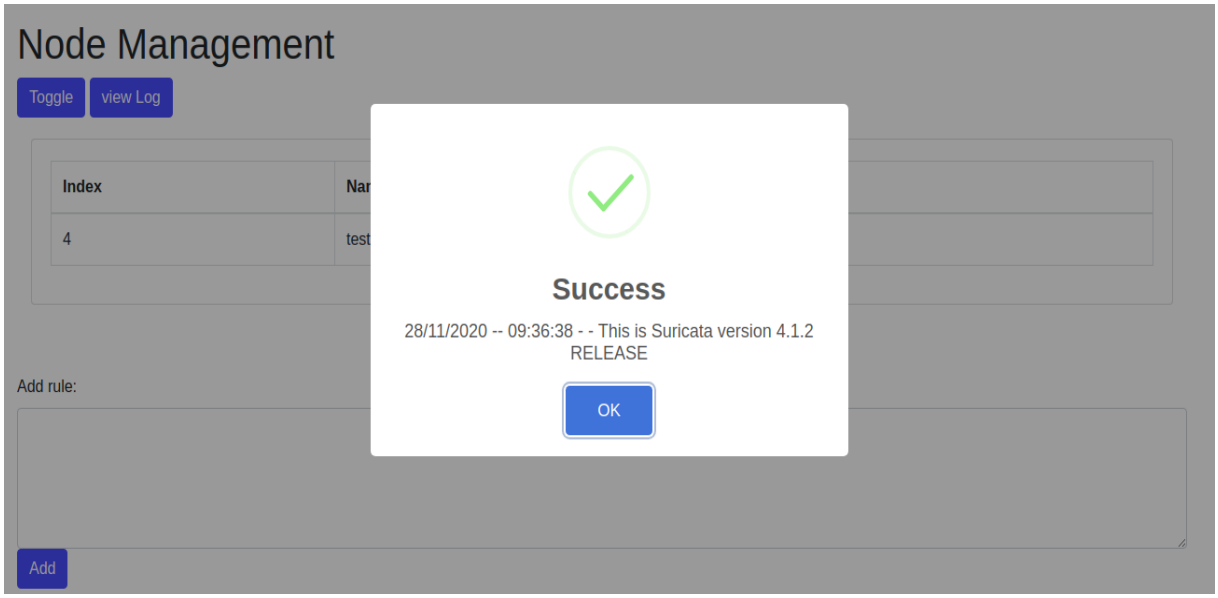


Hình 4.7 Các queue binding exchange

- Các binding cần thiết đã được khởi tạo để tạo các quy tắc ràng buộc từ các queue tới topic exchange đã được khởi tạo trước đó

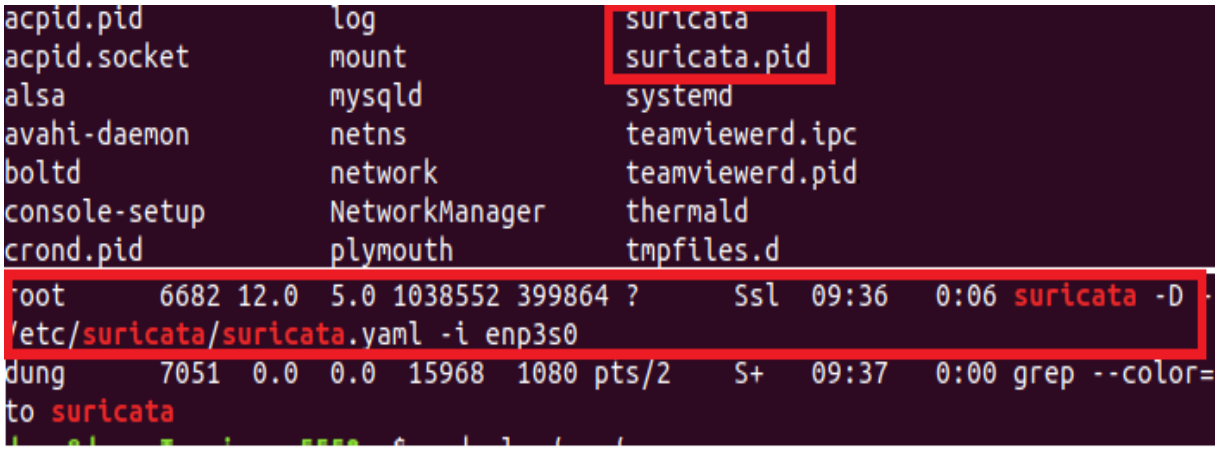
4.2.2 Thực hiện các chức năng của chương trình.

- Tiến hành bật Suricata thông qua node.



Hình 4.8 Khởi động Suricata thành công

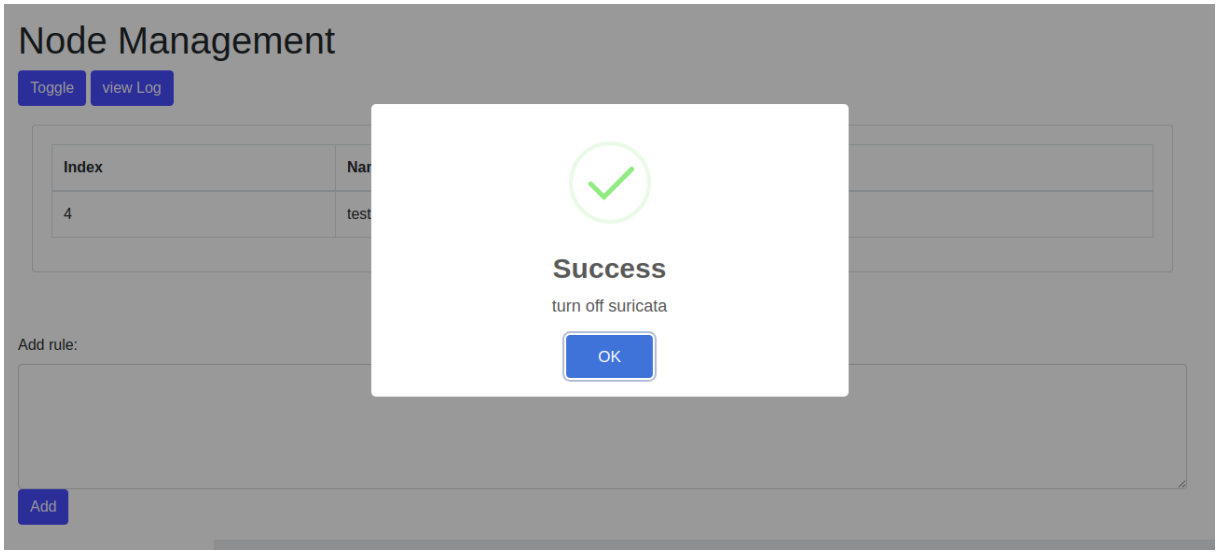
- Message yêu cầu node khởi động Suricata đã được gửi và nhận lại kết quả thành công, công cụ Suricata đã được khởi động.



Hình 4.9 Kiểm tra tiến trình khởi chạy Suricata

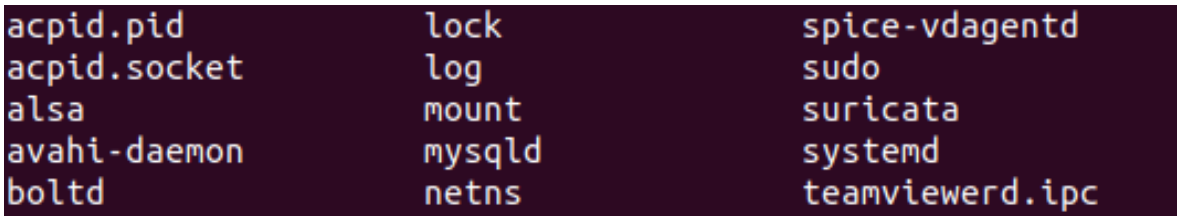
- Một tiến trình khởi chạy Suricata đã được thực thi đồng thời một file suricata.pid cũng đã được tạo.

- Tiến hành tắt Suricata.



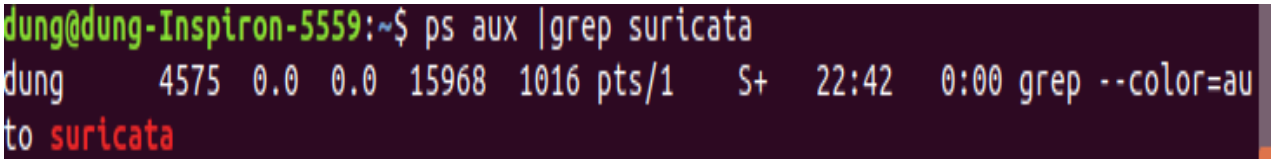
Hình 4.10 Nhận được thông báo tắt Suricata thành công

- Message yêu cầu tắt công cụ Suricata đã được gửi và nhận về kết quả thành công.



Hình 4.11 Kiểm tra file suricata.pid

- File suricata.pid đã được xóa bỏ đồng thời tiến trình khởi chạy Suricata trước đó cũng đã được tắt.

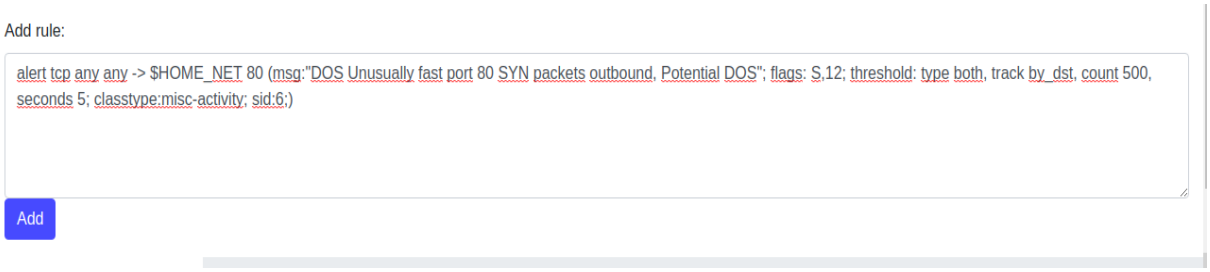


Hình 4.12 Kiểm tra tiến trình Suricata

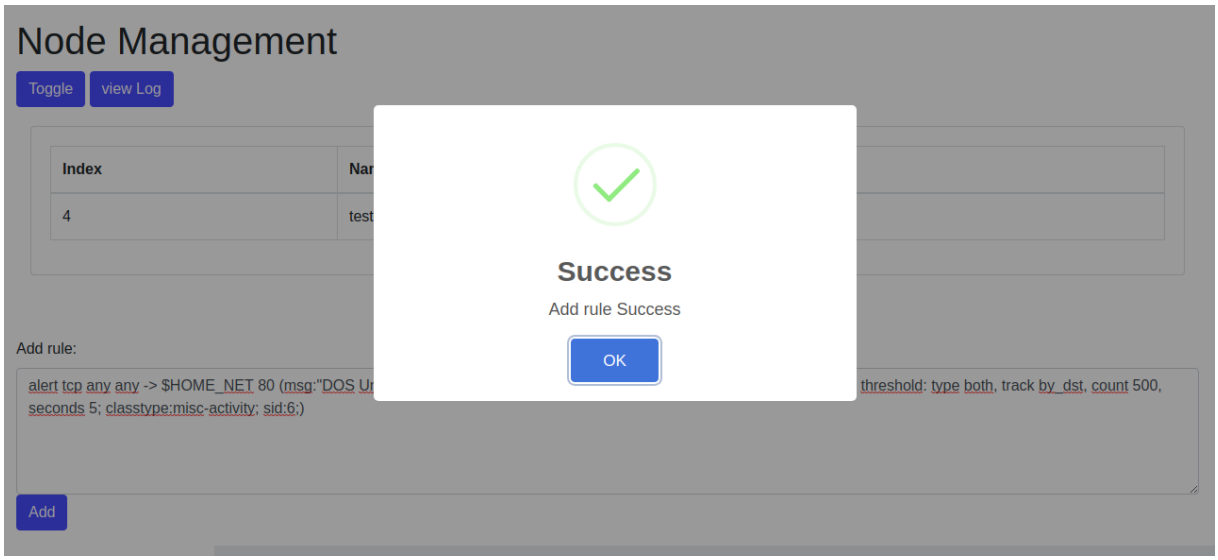
-Tiến trình chạy Suricata trước đó đã được tắt

- Thêm luật.

- Thực hiện viết hoặc dán một luật cần thêm cho công cụ Suricata trên giao diện web, tiến hành và tiến hành submit.

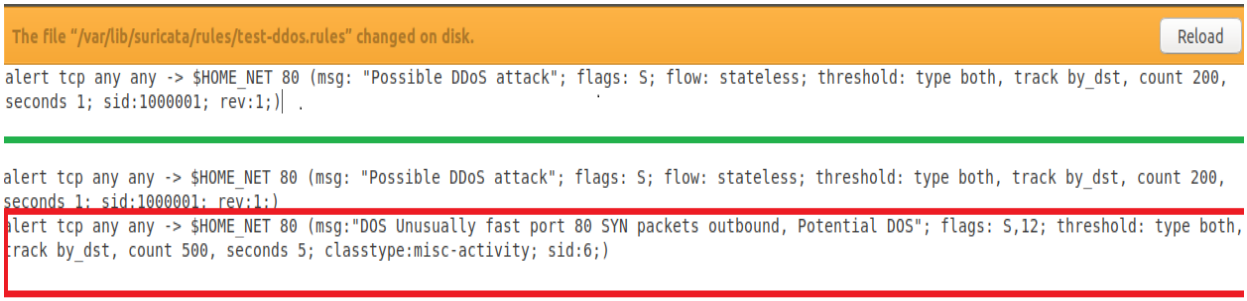


Hình 4.13 Viết luật vào form submit



Hình 4.14 Nhận được thông báo submit thành công

- Sau khi tiến hành submit thành công, luật đã được thêm vào một file custom rule, để có thể sử dụng được rule quản trị viên cần tiến hành khởi động lại công cụ Suricata.



Hình 4.15 Kiểm tra luật đã được ghi vào file thành công

- Xem log

- Sau khi yêu cầu xem log file được gửi tới server, một message sẽ được gửi tới node để thực hiện việc này, các dòng log file được sử dụng fast.log, nơi lưu những thông báo của Suricata khi có gói tin khớp với một signature.

Index	Name	Trang Thái
4	testnode	Đang bật

- 07/26/2020-16:18:24.348082 [**] [1:2210038:2] SURICATA STREAM FIN out of window [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.16.108:44928 -> 172.217.24.45:443
- 07/26/2020-16:19:15.808091 [**] [1:2210038:2] SURICATA STREAM FIN out of window [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.16.108:48086 -> 118.69.16.15:80
- 07/26/2020-16:19:28.092166 [**] [1:2210038:2] SURICATA STREAM FIN out of window [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.16.108:48086 -> 118.69.16.15:80
- 07/26/2020-16:19:28.092166 [**] [1:2210038:2] SURICATA STREAM FIN out of window [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.16.108:48086 -> 118.69.16.15:80
- 07/26/2020-16:19:42.182049 [**] [1:2210042:2] SURICATA STREAM TIMEWAIT ACK with wrong seq [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.16.108:48384 -> 182.161.72.130:443
- 07/26/2020-16:20:42.332124 [**] [1:2210038:2] SURICATA STREAM FIN out of window [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.16.108:48086 -> 118.69.16.15:80
- 07/26/2020-16:20:42.332129 [**] [1:2210038:2] SURICATA STREAM FIN out of window [**] [Classification: Generic Protocol Command Decode] [Priority: 3] {TCP}

Hình 4.16 Hiển thị log file

CHƯƠNG 5. KẾT LUẬN

Trong đồ án này em đã nghiên cứu phát triển một hệ thống nhỏ giúp các quản trị viên có thể quản lý các công cụ IDS/IPS Suricata một cách tập trung và từ xa, nhằm tìm cách giải quyết bài toán khi một hệ thống được cài đặt và tích hợp nhiều công cụ IDS/IPS. Đồ án cũng thành công trong việc triển khai xây dựng hệ thống này, có thể tiến hành các thao tác điều khiển cơ bản đối với công cụ Suricata.

Trong quá trình thực hiện đồ án em đã nghiên cứu tìm hiểu được một số công cụ và công nghệ hữu ích như Suricata, RabbitMQ, đồng thời cũng áp dụng được các kiến thức liên quan đến lập trình, mạng máy tính, v.v.v..

Phương hướng phát triển tiếp theo của đồ án này là có thể thực hiện được nhiều hơn các thao tác nâng cao để điều khiển và quản lý công cụ Suricata, ngoài ra có thể mở rộng để thực hiện việc quản lý đối với các công cụ IDS/IPS khác.

● TÀI LIỆU THAM KHẢO

Tiếng Việt

[1]https://securitydaily.net/network-hieu-ve-he-thong-phat-hien-xam-nhap-ids/

[2] https://securitydaily.net/network-security-he-thong-ngan-ngua-xam-nhap-ips/

[3] https://www.slideshare.net/ducmanhkthd/ids-183945922

[4]https://vi.compbs.com/what-s-best-intrusion-detection-software

[5]https://vietnetco.vn/firewall-utm/2971.html

[6] Nguyễn Hoàng Duy, Hệ thống phát hiện xâm nhập mạng Suricata,Đồ án chuyên ngành, Đại học công nghệ TP HCM, tr. 1-14.

[7]http://www.giaosucan.com/2018/01/kien-truc-message-queue-trong.html

https://securitydaily.net/tong-quan-ve-he-thong-giam-sat-an-ninh-mang/

[8]https://tapit.vn/5-giao-thuc-truyen-tai-du-lieu-trong-internet-things/

[9]https://viblo.asia/p/message-broker-la-gi-so-luoc-ve-rabbitmq-va-ung-dung-demo-djeZ1PVJKWz

[10]https://kipalog.com/posts/Tim-hieu-RabbitMQ---Phan-1

[11]https://gpcoder.com/6828-gioi-thieu-rabbitmq/

[12] https://medium.com/@locphamtan/rabbitmq-tìm-hiểu-về-exchanges-routing-keys-bindings-896d25386e14

[13]https://gpcoder.com/7095-su-dung-dead-letter-exchange-trong-rabbitmq/

[14]http://ipmac.vn/technology-corner/bao-mat-he-thong-voi-he-thong-idsips-phan-1

Tiếng Anh

[15]https://stackjava.com/spring/spring-boot-la-gi-gioi-thieu-spring-boot-framework.html

[16]https://suricata.readthedocs.io/en/suricata-6.0.0/

[17]https://www.rabbitmq.com/tutorials/tutorial-seven-java.html

[18]https://www.cloudamqp.com/blog/2015-09-03-part4-rabbitmq-for-beginners-exchanges-routing-keys-bindings.html

[19]https://www.cloudamqp.com/blog/2019-11-13-the-relationship-between-connections-and-channels-in-rabbitmq.html

[20]https://www.rabbitmq.com/tutorials/amqp-concepts.html

[21] https://aws.amazon.com/message-queue/benefits/

[22] https://suricata-ids.org/

[23]https://suricata.readthedocs.io/en/suricata-6.0.0/performance/runmodes.html

[24] https://securitydaily.net/tong-quan-ve-he-thong-giam-sat-an-ninh-mang/
[25] https://purplesec.us/
[26] https://wearesocial.com/
[27] https://vnetwork.vn/