HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN I

BÁO CÁO BÀI TẬP LỚN

MÔN HỌC: KỸ THUẬT THEO DÕI VÀ GIÁM SÁT AN TOÀN MẠNG

CHỦ ĐỀ: PHÁT HIỆN XÂM NHẬP VỚI SURICATA

Giảng viên hướng dẫn: *ThS Hoàng Mạnh Thắng*

Nhóm sinh viên thực hiện: *Nhóm 01 – INT1483-N03*

Danh sách sinh viên:	*Nguyễn Đức Hoằng*	*B17DCAT084*
	Vũ Tiến Hoà	*B17DCAT078*
	Nguyễn Công Thành	*B17DCAT167*
	Phạm Hải Vũ	*B17DCAT214*

Hà Nội, tháng 5 năm 2021

Mục lục

Danh mục hình ảnh

CHƯƠNG 1: TÌM HIỂU VỀ SURICATA

1.1 Giới thiệu về Suricata

Suricata là một hệ thống phát hiện xâm nhập dựa trên mã nguồn mở. Nó được phát triển bởi Open Information Security Foundation (OISF). Một phiên bản beta đã được phát hành vào tháng 12 năm 2009, bản chuẩn đầu tiên phát hành tiếp theo vào tháng 7 năm 2010.

Công cụ này được phát triển không nhằm cạnh tranh hay thay thế các công tụ hiện có, nhưng nó sẽ mang lại những ý tưởng và công nghệ mới trong lĩnh vực an toàn an ninh mạng.

Suricata là công cụ IDS/IPS dựa trên luật để theo dõi lưu lượng mạng và cung cấp cảnh báo đến người quản trị hệ thống khi có sự kiện đáng ngờ xảy ra. Nó được thiết kế đểtương thích với các thành phần an ninh mạng hiện có. Bản phát hành đầu tiên chạy trên nền tảng linux 2.6 có hỗ trợ nội tuyến (inline) và cấu hình giám sát lưu lượng thụ động có khả năng xử lý lưu lượng lên đến gigabit. Suricata là công cụ IDS/IPS miễn phí trong khi nó vẫn cung cấp những lựa chọn khả năng mở rộng cho các kiến trúc an ninh mạng phức tạp nhất.

Là một công cụ đa luồng, Suricata cung cấp tăng tốc độ và hiệu quả trong việc phân tích lưu lượng mạng. Ngoài việc tăng hiệu quả phần cứng (với phần cứng và card mạng giới hạn), công cụ này được xây dựng để tận dụng khả năng xử lý cao được cung cấp bởi chip CPU đa lõi mới nhất.

Người dùng có thể tải về và sử dụng Suricata tại Github của OISF tại: https://github.com/OISF/suricata. Phiên bản 6.0.2 là phiên bản mới nhất được Release của phần mềm vào ngày 02/3/2021.

User guide: Suricata User Guide — Suricata 6.0.2 documentation

Developer guide: Suricata Developers Guide - Suricata - Open Information Security Foundation (openinfosecfoundation.org)

Tại sao Suricata được nhiều người sử dụng?

- Dễ dàng cấu hình: Suricata làm việc như thế nào, tập tin cấu hình ở đâu, các luật như thế nào người quản trị đều có thể biết và cấu hình theo ý mình được. Kể cả việc tạo ra các luật mới.

- Suricata là phần mềm mã nguồn mở: Sricata được phát hành dưới giấy phép GNU/GPL điều này có nghĩa là bất cứ ai cũng có thể sử dụng Suricata một cách miễn phí dù đó là doanh nghiệp hay người dùng cá nhân. Ngoài ra vì là phần mềm mã nguồn mở nên Suricata có một cộng đồng người sử dụng lớn, sẵn sàng hỗ trợ nếu có bất cứ thắc mắc gì.

- Chạy trên nhiều nền tảng khác nhau: Chạy trên các hệ điều hành nguồn mở như Linux, CentOS. Debian, Fedora, FreeBSD, Window, Mac OS X (10.5.8 and 10.6.8)…

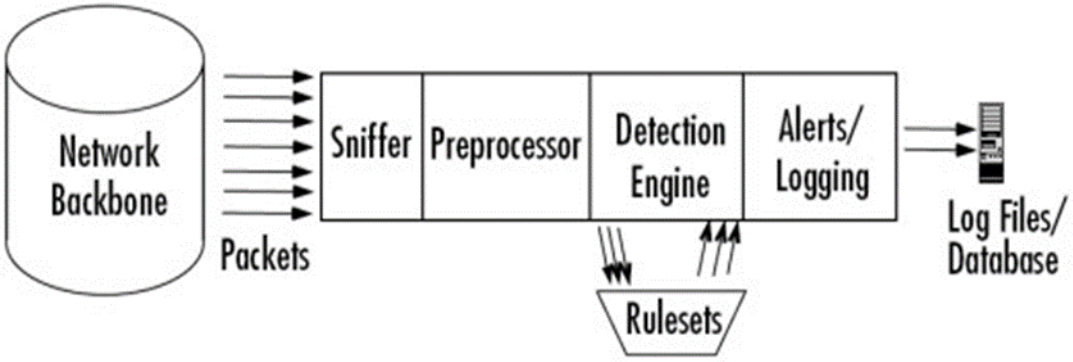
- Luật của Suricata thường xuyên được cập nhật: Các luật của Suricata thường xuyên được bổ sung và cập nhật các hình thức xâm nhập mới. Người sử dụng có thể dễ dàng tải về từ https://rules.emergingthreats.net/open/suricata/.

1.2 Cấu trúc Suricata

Suricata được phát triển dựa trên Snort nên nó vẫn giữ nguyên kiến trúc bên trong của Snort. Kiến trúc của nó có nhiều thành phần, với mỗi thành phần có một chức năng riêng.

Kiến trúc của Suricata gồm 4 phần cơ bản sau:

- The Sniffer (Packet Decoder).
- The Preprocessors.
- The Detection Engine.
- The Output:
    - Modul Alert/Logging.
    - Modul kết xuất thông tin.



Hình 1.1: Kiến trúc của Suricata

- Khi Suricata hoạt động nó sẽ thực hiện lắng nghe và thu bắt tất cả các gói tin nào di chuyển qua nó.

- Các gói tin sau khi bị bắt được đưa và module Sniffer, tại đây các gói tin sẽ được giải mã.

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

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

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

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

1.2.1 Packet Sniffer (Decoder)

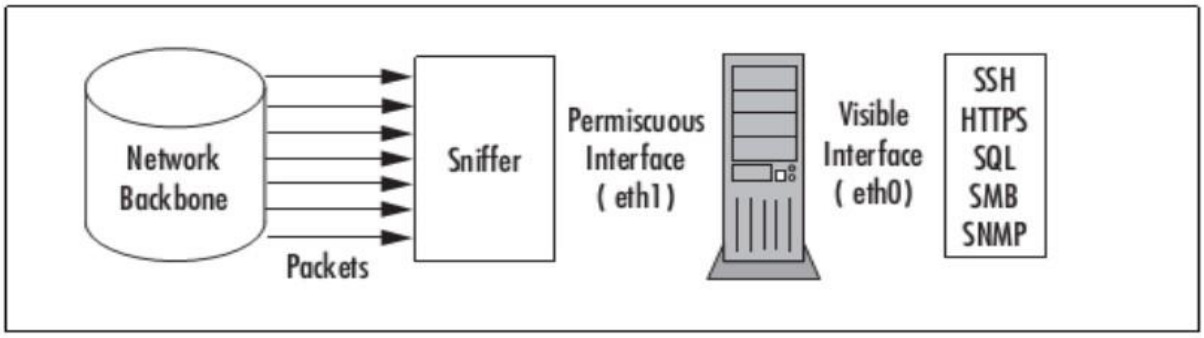
Packet Sniffer là một thiết bị phần cứng hoặc phần mềm được đặt vào trong mạng. Chức năng của nó tương tự như việc nghe lén trên điện thoại di động, nhưng thay vì hoạt động trên mạng điện thoại nó nghe lén trên mạng dữ liệu. Bởi vì trong mô hình mạng có nhiều giao thức cao cấp như TCP, UDP, ICMP... nên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

- Phân tích mạng và troubleshooting.

- Performance network and bechmarking

- Nghe lén mật khẩu clear-text và những dữ liệu khác.

Mã hóa lưu lượng mạng có thể tránh được việc sniffer các gói tin. Tùy vào mục đích mà packet sniffer có thể sử dụng cho mục đích tốt hoặc xấu.



Hình 1.2: Các gói tin đi vào Sniffer

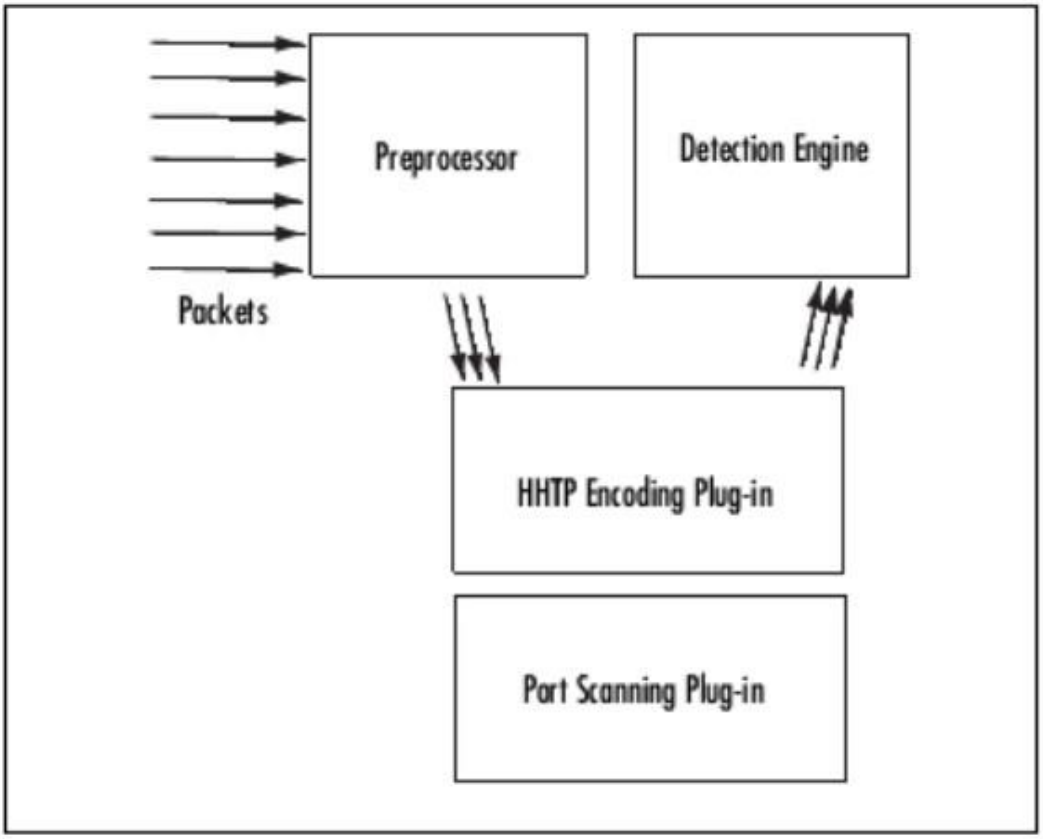
Khi Suricata đã nhận các gói tin từ quá trình sniffer nó sẽ đi vào quá trình giải mã. Chính xác thì nơi mà gói tin đi vào bộ giải mã phụ thuộc vào lớp liên kết mà trước đó đọc được. Snort hỗ trợ một số lớp liên kết từ pcap: Ethernet, 802.11, Token ring, FDDI, Cisco HDLC, SLIP, PPP và OpenBSD's PF. Ở trên lớp liên kết Suricata hỗ trợ giải mã các giao thức khác nhau, bao gồm: IP, ICMP, TCP, UDP (chi tiết trong mã nguồn src/decode.c)

Bất kể là lớp liên kết nào đang được sử dụng, tất cả các bộ giải mã sẽ đều làm việc theo một kiểu chung. Đối với trường hợp các lớp cụ thể, con trỏ trong cấu trúc của gói tin sẽ được thiết lập trỏ tới một phần khác của gói tin. Dựa vào các thông tin đã giải mã được, nó sẽ gọi các lớp cao hơn và giải mã cho đến khi không còn bộ giải mã nào nữa.

1.2.2 Preprocessors

Preprocessors là plug-in cho phép phân tích cú pháp dữ liệu theo những cách khác nhau. Nếu chạy Suricata mà không có bất cứ cấu hình nào về preprocessors trong tập tin cấu hình sẽ chỉ thấy từng gói dữ liệu riêng rẽ trên mạng. Điều này có thể làm IDS bỏ qua một số cuộc tấn công, vì nhiều loại hình tấn công hiện đại cố tình phân mảnh dữ liệu hoặc có tình đặt phần độc hại lên một gói tin và phần còn lại lên gói tin khác (kỹ thuật lẩn trốn).

Dữ liệu sẽ được đưa vào Preprocessors sau khi đi qua bộ giải mã gói tin (packet decoder). Suricata cung cấp một loạt các Preprocessors ví dụ như: Frag3 (một module chống phân mảnh gói tin IP), sfPortscan (module được thiết kế chống lại các cuộc trinh sát, như scan port, xác định dịch vụ, scan OS), Stream5 (module tái gộp các gói tin ở tầng TCP).



Hình 1.3: Quá trình xử lí ở Preprocessors

1.2.3 Detection Engine

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

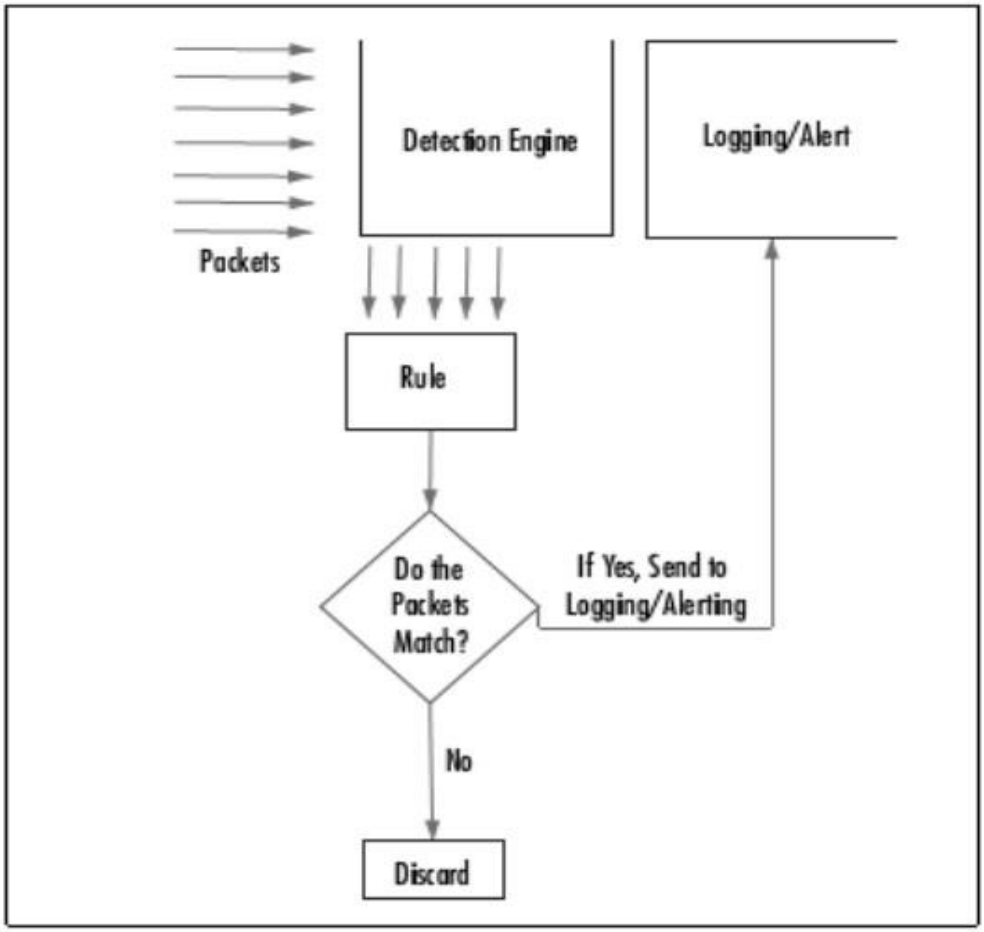
Các luật có thể được chia thành 2 phần:

- Phần Hearder: gồm các hành động (log/ alert), loại giao thức (TCP, UDP, ICMP...), địa chỉ IP nguồn, địa chỉ IP đích và port.

- Phần Options: là phần nội dung của gói tin được tạo ra để phù hợp với luật.

Luật là phần quan trọng mà bất cứ ai tìm hiểu về Suricata cần phải nắm rõ. Các luật trong Suricata có một cú pháp cụ thể. Cú pháp này có thể liên quan đến giao thức, nội dung, chiều dài, hearder và một vài thông số khác. Một khi hiể u được cấu trúc các luật trong Suricata, người quản trị có thể dễ dàng tinh chỉnh và tối ưu hóa chức năng phát hiện xâm

nhập của Suricata. Từ đó có thể định nghĩa các luật phù hợp với từng môi trường và hệ thống mạng.



Hình 1.4: Gói tin được xử lý ở Detection Engine bằng các luật

1.2.4 Thành phần cảnh báo/logging

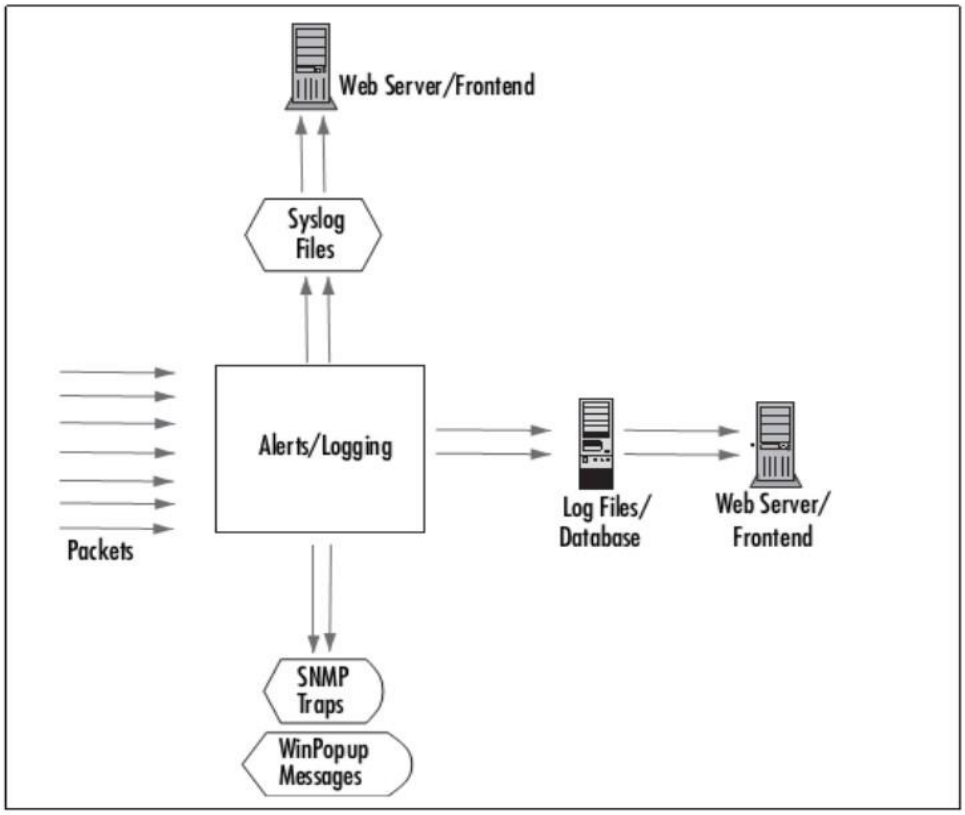
Cuối cùng sau khi các luật đã phù hợp với dữ liệu, chúng sẽ được chuyển tới thành phần cảnh báo và ghi lại (alert and loggin component). Cơ chế log sẽ lưu trữ các gói tin đã kích hoạt, các luật còn cơ chế cảnh báo sẽ thông báo các phân tích bị thất bại.

Giống như Preprocessors, chức năng này được cấu hình trong tập tin suricata.yaml, có thể chỉ định cảnh báo và ghi lại trong tập tin cấu hình nếu muốn kích hoạt.

Dữ liệu là giá trị cảnh báo, nhưng có thể chọn nhiều cách để gửi các cảnh báo này cũng như chỉ định nơi ghi lại các gói tin. Có thể gửi cảnh báo thông qua SMB (Server Message Block) pop- up tới máy trạm Windows, ghi chúng dưới dạng logfile, gửi qua mạng thông qua UNIX socket hoặc thông qua giao thức SNMP.

Cảnh báo cũng có thể lưu trữ dưới dạng cơ sở dữ liệu SQL như MySQL hoặc PostgerSQL. Thậm chí một vài hệ thống của các hãng thứ ba có thể gửi cảnh báo thông qua SMS tới điện thoại di động.

Có rất nhiều các add-on giúp người quản trị nhận các cảnh báo cũng như phân tích các dữ liệu một cách trực quan.



Hình 1.5: Thành phần cảnh báo và logging

1.3 Chức năng của Suricata

1.3.1 HTTP layer support

- Lắng nghe DNS packet.

- Register the HTTP protocol and state handling functions to APP layer of the engine.

- Register the Unit tests for the HTTP protocol.

- Print the stats of the HTTP requests (In các số liệu thống kê của các yêu cầu HTTP).

- Clears the HTTP server configuration memory used by HTP library (Xóa bộ nhớ cấu hình máy chủ HTTP được sử dụng bởi thư viện HTTP).

- Function callback to append chunks for Requests (chức năng gọi lại để thêm các đoạn dữ liệu cho các yêu cầu).

- Print the information and chunks of a Body (In thông tin các đoạn dữ liệu của một khối).

- Free the information held in the request body (cố định (giữ) các thông tin tự do trong các khối yêu cầu).

- Function to frees the HTTP state memory and also frees the HTTP connection parser memory which was used by the HTP library (giải phóng bộ nhớ HTTP và giải phóng các kết nối phân tích cú pháp HTTP được sử dụng bởi các thư viện HTTP).

- Sets a flag that informs the HTP app layer that some module in the engine needs the http request body data (Thiết lập cờ thông báo cho lớp ứng dụng HTTP là một số module trong cơ cấu cần khối dữ liệu http request).

- Sets a flag that informs the HTP app layer that some module in the engine needs the http request file (Thiết lập cờ thông báo cho lớp ứng dụng HTTP rằng một số module trong cơ cấu cần tập tin HTTP request).

1.3.2 Packet decoding

- Return a malloced packet (trả về gói tin malloced).

- Finalize decoding of a packet (Hoàn thành giải mã gói tin).

- Get a malloced packet (Nhận gói tin malloced).

- Return a packet to where it was allocated (trả về gói tin tới nơi nó được phân bổ).

- Get a packet (We try to get a packet from the packetpool first, but if that is empty we alloc a packet that is free'd again after processing).

- Copy data to Packet payload at given offset (Copy dữ liệu tới các khoảng trống đã xác định của gói dữ liệu).

- Copy data to Packet payload and set packet length (Sao chép dữ liệu tới gói dữ liệu và thiết lập chiều dài của gói).

- Set up a pseudo packet (tunnel) (Thiết lập gói tin giả).

- Setup a pseudo packet (reassembled frags) -> mức độ đệ quy ít hơn so với (tunnel).

- Inform defrag "parent" that a pseudo packet is now assosiated to it.

- Debug print function for printing addresses (gỡ lỗi in cho các địa chỉ in ấn).

- Alloc and setup DecodeThreadVars.

- Set data for Packet and set length when zero copy is used (Đặt dữ liệu cho gói dữ liệu và thiết lập chiều dài cho gói khi không sao chép)

1.3.3 State support

- Frees a DetectEngineState object.

- Check if we need to inspect this state (Kiểm tra nếu cần kiểm tra trạng thái).

- Match app layer sig list against app state and store relevant match information.

- Continue DeState detection of the signatures stored in the state.

- Update flow's inspection id's (Kiểm tra cập nhật dòng chảy các id).

- Reset a DetectEngineState state.

- Reset de state for active tx' To be used on detect engine reload.

- Get string for match enum.

1.3.4 Thresholding (Ngưỡng cảnh báo)

Tính năng này được sử dụng để giảm số lượng cảnh báo sai quy định. Nó có thể được điều chỉnh để giảm đáng kể số lượng cảnh báo sai, và cũng có thể được sử dụng để sinh ra các quy tắc mới. Các lệnh cảnh báo giới hạn số lần một sự kiện cụ thể trong một khoảng thời gian xác định.

- Return next DetectThresholdData for signature.

- Remove timeout threshold hash elements.

- Make the threshold logic for signatures.

- Init threshold context hash tables (Ngưỡng bối cảnh dữ liệu hỏng khởi tạo).

- Destroy threshold context hash tables (ngưỡng huỷ bỏ bối cảnh dữ liệu hỏng).

- This function will free all the entries of a list DetectTagDataEntry (giải phóng tất cả các lối vào của list DetectTagDataEntry).

1.4 Luật trong Suricata

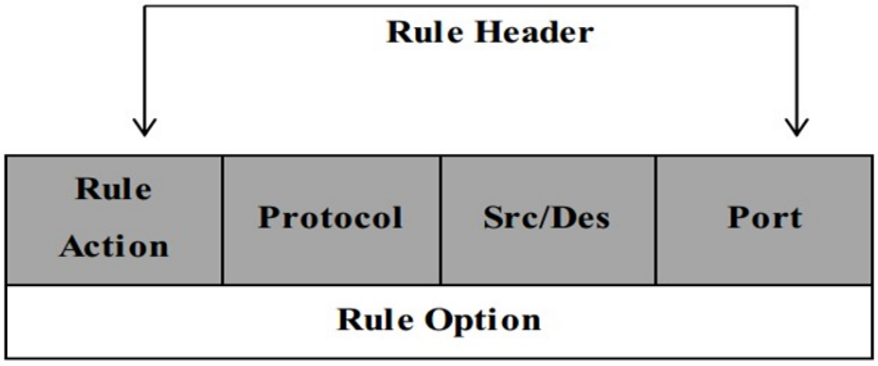
1.4.1 Giới thiệu

Luật" trong Suricata ta có thể hiểu một cách đơn giản nó giống như các quy tắc và luật lệ trong thế giới thực. Nghĩa là nó sẽ có phần mô tả một trạng thái và hành động gì sẽ xảy ra khi trạng thái đó đúng. Một trong những điểm đáng giá nhất của Suricata đó là khả

năng cho phép người sử dụng có thể tự viết các luật của riêng mình hoặc tùy biến các luật có sẵn cho phù hợp với hệ thống mạng của mình. Ngoài một cơ sở dữ liệu lớn mà người sử dụng có thể download từ trang chủ của Suricata, người quản trị có thể tự phát triển các luật cho hệ thống của mình. Thay vì phải phụ thuộc vào nhà cung cấp, một cơ quan bên ngoài, hoặc phải cập nhật khi có một cuộc tấn công mới hay một phương pháp khai thác lỗ hổng mới được phát hiện. Người quản trị có thể viết riêng một luật dành cho hệ thống của mình khi nhìn thấy các lưu lượng mạng bất thường và so sánh với bộ luật được cộng đồng phát triển. Ưu điểm của việc tự viết các luật là có thể tùy biến và cập nhật một cách cực kỳ nhanh chóng khi hệ thống mạng có sự bất thường.

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

1.4.2 Rule header



Hình 1.6: Cấu trúc luật trong Suricata

1.4.2.1 Rule Action

Phần Header sẽ chứa các thông tin xác định ai, ở đâu, cái gì của một gói tin, cũng như phải làm gì nếu tất cả các thuộc tính trong luật được hiện lên. Mục đầu tiên trong một luật đó chính là phần rule action, rule action sẽ nói cho Suricata biết phải làm gì khi thấy các gói tin phù hợp với các luật đã được quy định sẵn. Có 4 hành động mặc định trong Suricata đó là: pass (cho qua), drop (chặn gói tin), reject, alert (cảnh báo).

- **Pass:** nếu signature được so sánh trùng khớp và chỉ ra là pass thì Suricata sẽ thực hiện dừng quét gói tin và bỏ qua tất cả các luật phía sau đối với gói tin này.

- **Drop:** nếu chương trình tìm thấy một signature hợp lệ và nó chỉ ra là drop thì gói tin đó sẽ bị hủy bỏ và dừng truyền ngay lập tức, khi đó gói tin không thể đến được nơi nhận.

- **Reject:** là hành động bỏ qua gói tin, bỏ qua ở cả bên nhận và bên gửi. Suricata sẽ tạo ra một cảnh báo với gói tin này.

- **Alert:** nếu signature được so sánh là hợp lệ và có chứa một alert thì gói tin đó sẽ được xử lý giống như với một gói tin không hợp lệ. Suricata sẽ tạo ra một cảnh báo.

1.4.2.2 Protocol

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

1.4.2.3 IP Address

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

Một địa chỉ ip sẽ được viết dưới dạng ip_address/netmask. Điều này có nghĩa là nếu netmask là /24 thì lớp mạng đó là lớp mạng C, /16 là lớp mạng B hoặc /32 là chỉ một máy đơn. Ví dụ: địa chỉ 192.168.1.0/24 có nghĩa là một dải máy có địa chỉ IP từ 192.168.1.1 tới 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")

1.4.2.4 Port

Port có thể được định nghĩa bằng nhiều cách. Với từ khóa "any" giống như địa chỉ IP để chỉ có thể sử dụng bất kỳ port nào. Gán một port cố định, ví dụ như gán kiểm tra ở port 80 http hoặc port 22 ssh. Ngoài ra ta cũng có thể sử dụng toán tử phủ định để bỏ qua một port nào đó hoặc liệt kê một dải các port.

Ví dụ:

Alert UDP any any → 192.168.1.0/24 1:1024 – port bất kỳ tới dãy port từ 1 - 1024.

Alert UDP any any → 192.168.1.0/24 :6000 – port bất kỳ tới dãy port nhỏ hơn 6000.

Alert UDP any any → 192.168.1.0/24 !6000:6010 – port bất kỳ tới bất kỳ port nào, bỏ qua dãy port từ 6000 – 6010.

1.4.2.5 Điều hướng

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

Ví dụ:

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

1.4.3 Rule Option

Rule options chính là trung tâm của việc phát hiện xâm nhập. Nội dung chứa các dấu hiệu để xác định một cuộc xâm nhập. Nó nằm ngay sau phần Rule Header và được bọc bởi dấu ngoặc đơn "()". Tất cả các rule options sẽ được phân cách nhau bởi dấu chấm phẩy ";", phần đối số sẽ được tách ra bởi 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.

1.4.3.1 General

❖ **Msg**

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

msg: "....."";

❖ **Sid**

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

sid:123;

❖ **Rev**

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

rev:123

❖ **Reference**

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

reference: url, www.info.nl

system	URL Prefix
bugtraq	http://www.securityfocus.com/bid
cve	http://cve.mitre.org/cgi-bin/cvename.cgi?name=
nessus	http://cgi.nessus.org/plugins/dump.php3?id=
arachnids	http://www.whitehats.com/info/IDS
mcafee	http://vil.nai.com/vil/dispVirus.asp?virus_k=
url	http://

Hình 1.7: Bảng các tuỳ chọn của Reference

❖ **Classtype**

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

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

Signature	Classification.config	Alert
web-attack	web-attack, Web Application Attack, priority:1	Web Application Attack
not-suspicious	not-suspicious, Not Suspicious Traffic, priority:3	Not Suspicious Traffic

Hình 1.8: Thông tin phân loại lớp quy tắc

❖ **Priority**

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

priority:1;

❖ **Metadata**

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

metadata:.....;

1.4.3.2 Payload

❖ **Content**

Content: thể hiện nội dung chúng ta cần viết trong signature, nội dung này được đặt giữa 2 dấu nháy kép. Nội dung là các byte dữ liệu, có 256 giá trị khác nhau (0-255). Chúng có thể là các ký tự thường, ký tự hoa, các ký tự đặc biệt, hay là các mã 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: ".....";

Một số ký tự đặc biệt, khi cho trong nội dung chỉ có thể sử dụng các mã hexa để biểu diễn nó.

" |22|

; |3B|

: |3A|

| |7C|

❖ **Nocase**

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

content: "abC"; nocase;

❖ **Depth**

Depth: sau từ khóa depth là một số, chỉ ra bao nhiêu byte từ đầu một payload cần được kiểm tra. Depth cần được đặt sau một nội dung. Ví dụ:

Ta có một payload: *abCdefghij*

Ta thực hiện kiểm tra 3 bytes đầu của payload.

content: "abC"; depth:3;

❖ **Offset**

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

content: “def”; offset:3;

Ví dụ:

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

❖ **Distance**

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

Ví dụ:

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

❖ **Within**

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

Ví dụ:

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

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

❖ **Dsize**

Dsize: được dùng để tìm một payload có độ dài bất kỳ. Cách sử dụng:

dsize:min<>max;

❖ **Rpc**

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

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

❖ **Replace**

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

1.4.3.3 Non-payload

a) IP

❖ **ttl**

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

ttl:<number>;

❖ **ipopts**

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

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 1.9: Một số tuỳ chọn của ipopts

Định dạng của ipopts như sau:

ipopts: <name>;

❖ **sameip**

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

sameip;

❖ **ip_proto**

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

1	ICMP	Internet Control Message
6	TCP	Transmission Control Protocol
17	UDP	User Datagram
47	GRE	General Routing Encapsulation
50	ESP	Encap Security Payload for IPv6
51	AH	Authentication Header for Ipv6
58	IPv6-ICMP	ICMP for Ipv6

Định dạng của ip_proto như sau:

ip_proto:<number/name>;

❖ **id**

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

id:<number>;

❖ **geoip**

Cho phép xác định địa chỉ nguồn, đích để gói tin lưu thông trên mạng.

❖ **fragbits**

Được dùng để kiểm tra các fragbits của gói tin. Nó bao gồm các cơ chế sau:

M	More Fragments
D	Do not Fragment
R	Reserved Bit
+	Match on the specified bits, plus any others
*	Match if any of the specified bits are set

/	Match if the specified bits are not set

Định dạng của một Fragbits như sau:

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

❖ **fragoffset**

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

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

Định dạng của fragoffset:

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

b) TCP

❖ **Sed**

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

❖ **Ack**

Được sử dụng để kiểm tra xem gói tin đã được nhận bởi nơi nhận hay chưa trong giao thức kết nối TCP. Số thứ tự của ACK sẽ tăng lên tương ứng với số byte dữ liệu đã được nhận thành công.

❖ **Window**

Được sử dụng để kiểm tra kích thước của cửa sổ TCP. Kích thước cửa sổ TCP là một cơ chế dùng để kiểm soát các dòng dữ liệu. Cửa sổ được thiết lập bởi người nhận, nó chỉ ra số lượng byte có thể nhận để tránh tình trạng bên nhận bị tràn dữ liệu. Giá trị kích thước của cửa sổ có thể chạy từ 2 đến 65.535 byte.

c) ICMP

❖ **Itype**

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

Định dạng của itype như sau:

itype:min<>max; itype:[<|>]<number>;

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 1.10: Bảng Type của ICMP Header

❖ **Icode**

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

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

❖ **Icmp_id**

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

icmp_id:<number>;

❖ **Icmp_seq**

Được sử dụng để kiểm tra số thứ tự của ICMP. Định dạng của icmp_seq như sau:

icmp_seq:<number>;

d) HTTP

❖ **http_method**

Chỉ ra các phương thức được áp dụng với các request http. Các phương thức http: GET, POST, PUT, HEAD, DELETE, TRACE, OPTIONS, CONNECT và PATCH.

❖ **http_uri và http_raw_uri**

Chỉ ra đường dẫn tới nơi chứa nội dung yêu cầu.

❖ **http_header**

Chỉ ra phương thức sử dụng, địa chỉ cần truy cập tới và tình trạng kết nối.

❖ **http_cookie**

❖ **http_user_agent**

Là một phần của http_header, chỉ ra thông tin về trình duyệt của người dùng.

❖ **http_client_body**

Chỉ ra các yêu cầu của máy trạm.

❖ **http_stat_code**

Chỉ ra mã trạng thái của server mà máy trạm yêu cầu kết nối tới.

❖ **http_stat_msg**

Các dòng tin thông báo về tình trạng máy chủ, hay tình trạng về việc đáp ứng các yêu cầu kết nối của máy trạm.

❖ **http_server_body**

Chỉ ra nội dung đáp trả các yêu cầu từ máy trạm của máy chủ.

❖ **File_data**

Chỉ ra nội dung, đường dẫn tới file chứa dữ liệu được yêu cầu.

e) Flow

❖ **Flowbits**

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

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:

to_client	established/stateless
from_client	established/stateless
to_server	established/stateless
from_server	established/stateless

CHƯƠNG 2: CÀI ĐẶT SURICATA

2.1 Cài đặt Suricata

2.1.1 Hệ thống

Máy chủ chạy Ubuntu 16.04 được cài đặt ảo áo trên VMWare Workstations.

2.1.2 Cài đặt gói phần mềm

Truy cập vào link từ OISF để chọn phiên bản Suricata cần cài đặt.

Trước khi cài đặt phần mềm chúng ta tiến hành cài đặt các thành phần cần thiết khác cho Suricata:

```
apt-get install libpcre3 libpcre3-dbg libpcre3-dev build-essential libpcap-dev \
                libnet1-dev libyaml-0-2 libyaml-dev pkg-config zlib1g zlib1g-dev \
                libcap-ng-dev libcap-ng0 make libmagic-dev \
                libnss3-dev libgeoip-dev liblua5.1-dev libhiredis-dev libevent-dev \
                python-yaml rustc cargo
```

Mặc định, Suricata làm việc như một IDS. Nếu người dùng muốn cài đặt tính năng IPS cho hệ thống cần cài đặt thêm một số thành phần khác:

```
apt-get install libnetfilter-queue-dev libnetfilter-queue1  \
                libnetfilter-log-dev libnetfilter-log1      \
                libnfnetlink-dev libnfnetlink0
```

Tải về Suricata:

```
wget http://www.openinfosecfoundation.org/download/suricata-4.0.1.tar.gz
tar -xvzf suricata-4.0.1.tar.gz cd suricata-4.0.1
```

Nếu muốn xây dựng Suricata với khả năng của một IPS, chạy dòng lệnh:

```
./configure --enable-nfqueue --prefix=/usr --sysconfdir=/etc --localstatedir=/var
```

Cài đặt và cấu hình cho Suricata:

```
./configure --prefix=/usr --sysconfdir=/etc --localstatedir=/var
make sudo make install
sudo ldconfig
```

Tạo thư mục chứa thông tin về Suricta:

```
sudo mkdir /var/log/suricata
```

Tạo thư mục cài đặt:

```
sudo mkdir /etc/suricata
```

Copy các file classification.config, reference.config and suricata.yaml from the base build/installation directory vào thư mục vừa tạo /etc/suricata:

```
sudo cp classification.config /etc/suricata
sudo cp reference.config /etc/suricata
sudo cp suricata.yaml /etc/suricata
```

Để quản lý luật chúng ta cài Oinkmaster:

```
sudo apt-get install oinkmaster
```

2.1.3 Cấu hình cho Suricata

Tiến hành truy xuất vào đường dẫn /etc/suricata/suricata.yaml (suricata.yaml là file cho phép thiết lập cấu hình hoạt động của suricata).

- Thiết lập các tham số địa chỉ báo gồm HOME_NET (cho phép thiết lập các giải địa chỉ mạng cho phép giám sát) EXTERNAL_NET (cho phép thiết lập các giải mạng từ kết nối bên ngoài vào cho phép giám sát và như thiết lập mặc định các địa chỉ sẽ khác so với địa chỉ HOME_NET) ngoài ra còn cho phép thiết lập một số địa chỉ của server như:

```
HTTP_SERVERS: "$HOME_NET", SMTP_SERVERS: "$HOME_NET", SQL_SERVERS: "$HOME_NET",
DNS_SERVERS: "$HOME_NET" , TELNET_SERVERS: "$HOME_NET", AIM_SERVERS:"$EXTERNAL_NET",
DC_SERVERS: "$HOME_NET", DNP3_SERVER: "$HOME_NET", DNP3_CLIENT: "$HOME_NET",
MODBUS_CLIENT: "$HOME_NET", MODBUS_SERVER: "$HOME_NET", ENIP_CLIENT: "$HOME_NET",
ENIP_SERVER: "$HOME_NET"
```

```
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]"
  #HOME_NET: "any"

  EXTERNAL_NET: "!$HOME_NET"
  #EXTERNAL_NET: "any"

  HTTP_SERVERS: "$HOME_NET"
  SMTP_SERVERS: "$HOME_NET"
  SQL_SERVERS: "$HOME_NET"
  DNS_SERVERS: "$HOME_NET"
  TELNET_SERVERS: "$HOME_NET"
  AIM_SERVERS: "$EXTERNAL_NET"
  DC_SERVERS: "$HOME_NET"
  DNP3_SERVER: "$HOME_NET"
  DNP3_CLIENT: "$HOME_NET"
  MODBUS_CLIENT: "$HOME_NET"
  MODBUS_SERVER: "$HOME_NET"
  ENIP_CLIENT: "$HOME_NET"
  ENIP_SERVER: "$HOME_NET"
```

Hình 2.1: Thiết lập tham số địa chỉ

- Thiết lập port:

```
port-groups:
  HTTP_PORTS: "80"
  SHELLCODE_PORTS: "!80"
  ORACLE_PORTS: 1521
  SSH_PORTS: 22
  DNP3_PORTS: 20000
  MODBUS_PORTS: 502
  FILE_DATA_PORTS: "[$HTTP_PORTS,110,143]"
  FTP_PORTS: 21
```

Hình 2.2: Thiết lập tham số port

- Thiết lập các loại đầu ra trong khi giám sát bao gồm các file log cho dịch vụ DNS, SSH, TLS, HTTP, các traffic bị DROP, fast, …

```
default-log-dir: /var/log/suricata/

# global stats configuration
stats:
  enabled: yes
  # The interval field (in seconds) controls at what interval
  # the loggers are invoked.
  interval: 8
  # Add decode events as stats.
  #decoder-events: true
  # Decoder event prefix in stats. Has been 'decoder' before, but that leads
  # to missing events in the eve.stats records. See issue #2225.
  decoder-events-prefix: "decoder.event"
  # Add stream events as stats.
  #stream-events: false

# Configure the type of alert (and other) logging you would like.
outputs:
  - alert-json-log:
      enabled: yes
      filename: alert-json.log
      payload: yes
      packet: yes

  - dns-json-log:
      enabled: yes
      filename: dns-json.log
  - drop-json-log:
      enabled: yes
      filename: drop-json.log
  - http-json-log:
      enabled: yes
      filename: http-json.log
  - ssh-json-log:
      enabled: yes
      filename: ssh-json.log
  - tls-json-log:
      enabled: yes
      filename: tls-json.log
  # a line based alerts log similar to Snort's fast.log
```

Hình 2.3: Thiết lập đầu ra

- Đầu ra eve.json cho phép xuất ra các cảnh báo, metadata, thông tin các file các bản ghi cụ thể về giao thức thông qua file json.

```
- fast:
    enabled: yes
    filename: fast.log
    append: yes
    #filetype: regular # 'regular', 'unix_stream' or 'unix_dgram'

# Extensible Event Format (nicknamed EVE) event log in JSON format
- eve-log:
    enabled: yes
    filetype: regular #regular|syslog|unix_dgram|unix_stream|redis
    filename: eve.json
    #prefix: "@cee: " # prefix to prepend to each log entry
    # the following are valid when type: syslog above
    #identity: "suricata"
    #facility: local5
    #level: Info ## possible levels: Emergency, Alert, Critical,
                 ## Error, Warning, Notice, Info, Debug
```

Hình 2.4: Thiết lập eve.json

- Xuất cảnh báo đầu ra cho Barnyard2:

```
# alert output for use with Barnyard2
- unified2-alert:
    enabled: yes
    filename: unified2.alert
```

Hình 2.5: Xuất cảnh báo cho Barnyard2

Unified2-alert: là một định dạng đầu ra dưới dạng nhị phân cho phép xuất dưới ba chế độ dạng console, file, syslog. Sau khi xuất được đầu ra Barnyard2 cho phép lấy đầu ra này để ghi vào CSDL SQL.

2.1.4 Thiết lập các luật

```
default-rule-path: /etc/suricata/rules

rule-files:
 - botcc.rules
 # - botcc.portgrouped.rules
 - ciarmy.rules
 - compromised.rules
 - drop.rules
 - dshield.rules
# - emerging-activex.rules
 - emerging-attack_response.rules
 - emerging-chat.rules
 - emerging-current_events.rules
 - emerging-dns.rules
 - emerging-dos.rules
 - emerging-exploit.rules
 - emerging-ftp.rules
# - emerging-games.rules
# - emerging-icmp_info.rules
# - emerging-icmp.rules
 - emerging-imap.rules
# - emerging-inappropriate.rules
# - emerging-info.rules
 - emerging-malware.rules
 - emerging-misc.rules
 - emerging-mobile_malware.rules
 - emerging-netbios.rules
 - emerging-p2p.rules
 - emerging-policy.rules
 - emerging-pop3.rules
 - emerging-rpc.rules
# - emerging-scada.rules
# - emerging-scada_special.rules
 - emerging-scan.rules
# - emerging-shellcode.rules
 - emerging-smtp.rules
 - emerging-snmp.rules
 - emerging-sql.rules
 - emerging-telnet.rules
 - emerging-tftp.rules
 - emerging-trojan.rules
 - emerging-user_agents.rules
 - emerging-voip.rules
 - emerging-web_client.rules
 - emerging-web_server.rules
# - emerging-web_specific_apps.rules
 - emerging-worm.rules
 - tor.rules
# - decoder-events.rules # available in suricata sources under rules dir
# - stream-events.rules  # available in suricata sources under rules dir
 - http-events.rules    # available in suricata sources under rules dir
 - smtp-events.rules    # available in suricata sources under rules dir
 - dns-events.rules     # available in suricata sources under rules dir
 - tls-events.rules     # available in suricata sources under rules dir
# - modbus-events.rules  # available in suricata sources under rules dir
# - app-layer-events.rules  # available in suricata sources under rules dir
# - dnp3-events.rules       # available in suricata sources under rules dir
```

Hình 2.6: Thiết lập các luật trong /etc/suricata/rules

2.2 Cài đặt Snorby

Cài đặt các phụ thuộc:

```
apt-get install libyaml-dev git-core default-jre imagemagick libmagickwand-dev
wkhtmltopdf build-essential libssl-dev libreadline-gplv2-dev zlib1g-dev libsqlite3-
dev libxslt1-dev libxml2-dev libmysqlclient-dev libmysql++-dev libcurl4-openssl-dev
ruby ruby-dev mysql-server imagemagick apache2 libxml2-dev libxslt-dev
```

Cài đặt Bundler và Rails:

```
gem install bundler rails wkhtmltopdf
gem install rake --version=0.9.2
```

Cài đặt Snorby:

```
cd /opt
git clone http://github.com/Snorby/snorby.git
cd /opt/snorby
bundle install
```

Thiết lập cấu hình Snorby qua /opt/snorby/snorby_config.yml:

```
production:
  domain: 'demo.snorby.org'
  wkhtmltopdf: /usr/bin/wkhtmltopdf
  ssl: false
  mailer_sender: 'snorby@snorby.org'
  geoip_uri: "http://geolite.maxmind.com/download/geoip/database/GeoLiteCountry/GeoIP.dat.gz"
  rules:
    - "/etc/suricata/rules/botcc.portgrouped.rules"
    - "/etc/suricata/rules/emerging-netbios.rules"
    - "/etc/suricata/rules/botcc.rules"
    - "/etc/suricata/rules/emerging-p2p.rules"
    - "/etc/suricata/rules/emerging-policy.rules"
    - "/etc/suricata/rules/ciarmy.rules"
    - "/etc/suricata/rules/emerging-pop3.rules"
    - "/etc/suricata/rules/emerging-rpc.rules"
    - "/etc/suricata/rules/emerging-scada.rules"
    - "/etc/suricata/rules/compromised.rules"
    - "/etc/suricata/rules/emerging-scan.rules"
    - "/etc/suricata/rules/drop.rules"
    - "/etc/suricata/rules/emerging-shellcode.rules"
    - "/etc/suricata/rules/dshield.rules"
    - "/etc/suricata/rules/emerging-smtp.rules"
    - "/etc/suricata/rules/emerging-activex.rules"
    - "/etc/suricata/rules/emerging-snmp.rules"
    - "/etc/suricata/rules/emerging-attack_response.rules"
    - "/etc/suricata/rules/emerging-sql.rules"
    - "/etc/suricata/rules/emerging-chat.rules"
    - "/etc/suricata/rules/emerging-telnet.rules"
    - "/etc/suricata/rules/emerging-current_events.rules"
    - "/etc/suricata/rules/emerging-tftp.rules"
    - "/etc/suricata/rules/emerging-deleted.rules"
    - "/etc/suricata/rules/emerging-trojan.rules"
    - "/etc/suricata/rules/emerging-dns.rules"
    - "/etc/suricata/rules/emerging-user_agents.rules"
    - "/etc/suricata/rules/emerging-dos.rules"
    - "/etc/suricata/rules/emerging-voip.rules"
    - "/etc/suricata/rules/emerging-exploit.rules"
    - "/etc/suricata/rules/emerging-web_client.rules"
    - "/etc/suricata/rules/emerging-ftp.rules"
    - "/etc/suricata/rules/emerging-web_server.rules"
    - "/etc/suricata/rules/emerging-games.rules"
    - "/etc/suricata/rules/emerging-web_specific_apps.rules"
    - "/etc/suricata/rules/emerging-icmp_info.rules"
    - "/etc/suricata/rules/emerging-worm.rules"
    - "/etc/suricata/rules/emerging-icmp.rules"
    - "/etc/suricata/rules/emerging-imap.rules"
    - "/etc/suricata/rules/emerging-inappropriate.rules"
    - "/etc/suricata/rules/emerging-info.rules"
    - "/etc/suricata/rules/emerging-malware.rules"
    - "/etc/suricata/rules/emerging-misc.rules"
    - "/etc/suricata/rules/tor.rules"
    - "/etc/suricata/rules/emerging-mobile_malware.rules"
  authentication_mode: database
```

Hình 2.7: Cấu hình Snorby

Cấu hình kết nối Snorby với MySQL:

```
# Snorby Database Configuration
#
# Please set your database password/user below
# NOTE: Indentation is important.
#
snorby: &snorby
  adapter: mysql
  username: snorby
  password: "snorby" # Example: password: "12345678"
  host: localhost

development:
  database: snorby
  <<: *snorby

test:
  database: snorby
  <<: *snorby

production:
  database: snorby
  <<: *snorby
```

Hình 2.8: Kết nối Snorby với MySQL

2.3 Cài đặt Barnyard2

Cài đặt các gói hỗ trợ:

```
apt-get install dh-autoreconf libpcap-dev libmysqld-dev libdaq-dev mysql-client
autoconf
```

Cài đặt daq:

```
apt-get install bison flex
wget https://www.snort.org/downloads/snort/daq-2.0.6.tar.gz tar xvf daq-2.0.6.tar.gz
cd /source/daq-2.0.6
./configure
make
make install
```

Cài đặt libdnet:

```
wget   http://downloads.sourceforge.net/project/libdnet/libdnet/libdnet-
1.11/libdnet1.11.tar.gz?r=http%3A%2F%2Flibdnet.sourceforge.net%2F&ts=1461552505&use_m
irro r=iweb

mv libdnet-1.11.tar.gz?r=http:%2F%2Flibdnet.sourceforge.net%2F libdnet-1.11.tar.gz
tar xvf libdnet-1.11.tar.gz
cd /source/libdnet-1.11
./configure make
make install
```

Cài đặt Barnyard2:

```
cd /opt
git clone https://github.com/firnsy/barnyard2.git
cd /opt/barnyard2 ./autogen.sh
./configure --with-mysql --with-mysql-libraries=/usr/lib/x86_64-linux-gnu/
make
make install
```

Tiến hành cấu hình thông qua /etc/suricata/barnyard2.conf:

Hình 2.9: Cấu hình /etc/suricata/barnyard2.conf

Tạo đường dẫn để lưu log cho barnyard2:

```
mkdir -p /var/log/barnyard2
```

Để nạp đầu ra của Suricata vào CSDL ta có thể sử dụng lệnh sau:

barnyard2 -c /etc/suricata/barnyard2.conf -d /var/log/suricata -f unified2.alert -w /var/log/suricata/suricata.waldo

2.4 Thiết lập lịch trình tự động xuất hoá quá trình

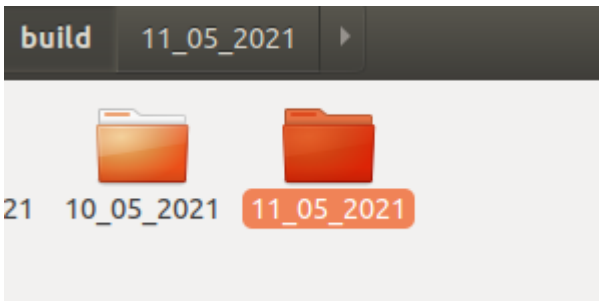
```
  GNU nano 2.5.3                  File: /tmp/crontab.KAwW4W/crontab

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 0 * * * sh /home/ash/build/ngay_thang.sh
0 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23 * * * sh /home/ash/build/gio.sh
1 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23 * * * sh /home/ash/build/run1.sh
0 0 * * * sh /home/ash/runtest/ngay_thang.sh
0 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23 * * * sh /home/ash/runtest/gio.sh
0 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23 * * * sh /home/ash/runtest/run_suricata.sh
0 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23 * * * sh /home/ash/runtest/run_barnyards.sh
```

Hình 2.10: Tạo Cron cho quá trình tự động

Quá trình cho phép tạo ra các thư mục theo từng ngày cho phép lưu kết quả các traffic network theo từng giờ và được lưu dưới dạng pcap file:

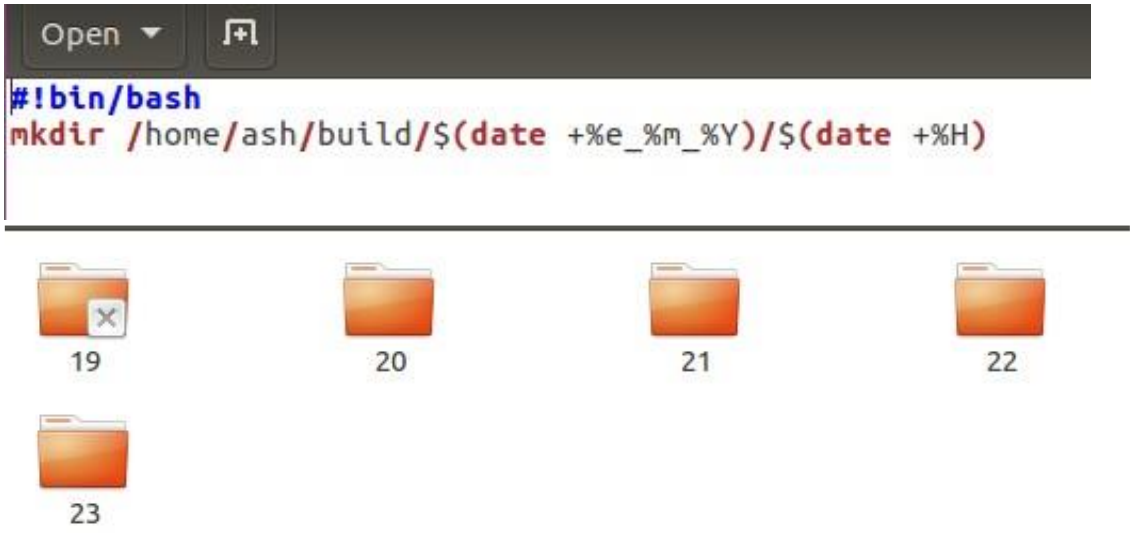
- Tạo thư mục theo ngày:

```
#!bin/bash
mkdir /home/ash/build/$(date +%e_%m_%Y)
```



Hình 2.11: Tạo thư mục theo ngày

- Tạo thư mục giờ trong ngày:



Hình 2.12: Tạo thư mục theo giờ trong ngày

- Xuất file pcap theo giờ:

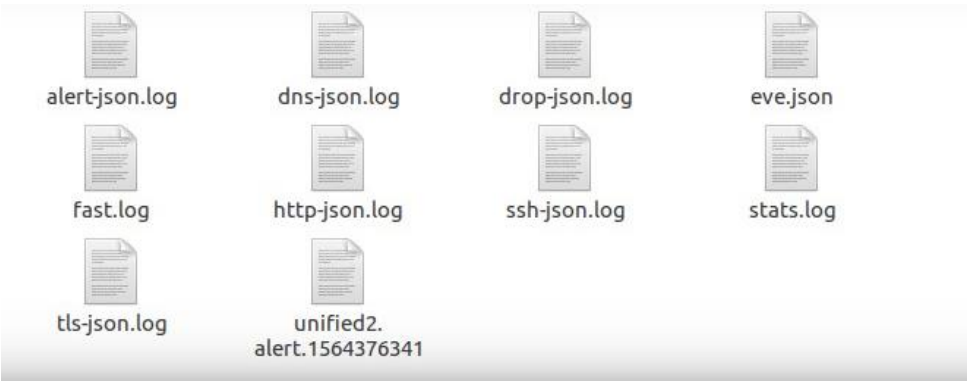
Hình 2.13: Xuất file pcap theo giờ

- Sau đó tiến hành khởi chạ suricata theo từng giờ trong ngày cho phép xuất ra các log cũng như json file và Unified2-alert:

```
#!bin/bash
suricata -c /etc/suricata/suricata.yaml -i ens33 -l /home/ash/runtest/$(date +%e_%m_%Y)/$(date +%H)
```

Hình 2.14: Khởi chạy Suricata theo giờ

- Kết quả:



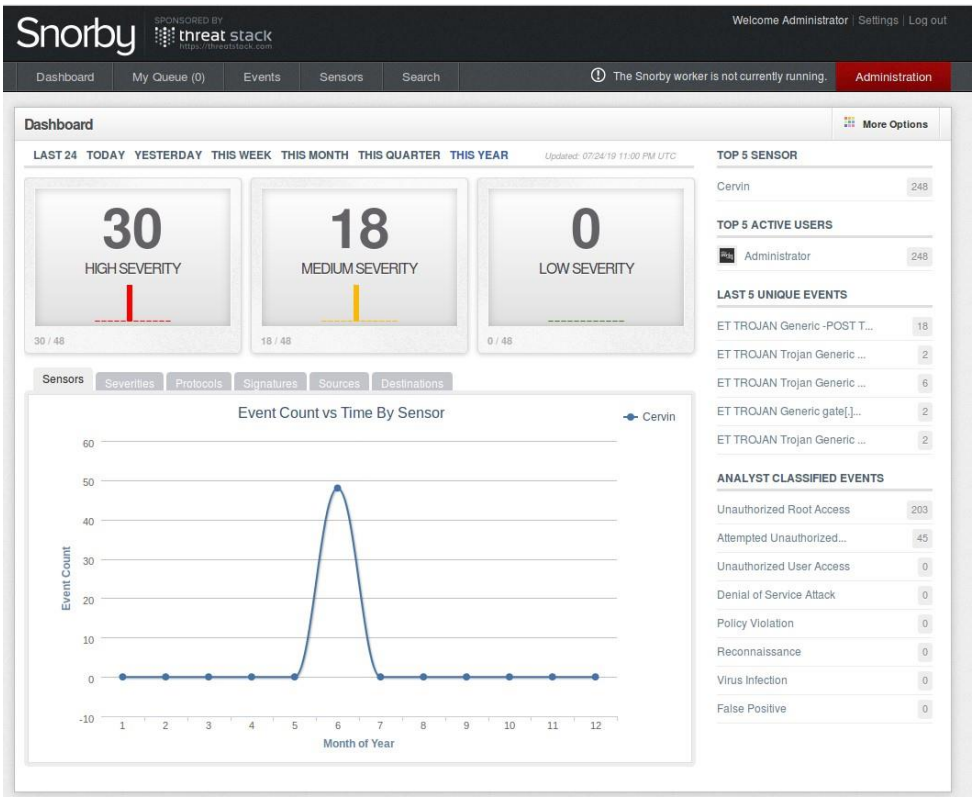
Hình 2.15: Kết quả chạy Suricata theo giờ

- Xuất kết quả vào MySQL:

```
#!bin/bash
barnyard2 -c /etc/suricata/barnyard2.conf -d /home/ash/runtest/$(date +%e_%m_%Y)/$(date +%H) -f unified2.alert -w /var/log/suricata/
suricata.waldo
```

Hình 2.16: Xuất kết quả vào MySQL

- Chạy thử với một pcap file thực tế và kiểm tra thông tin trên Snorby:



Hình 2.17: Kết quả trên Snorby