ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

NGUYỄN MẠNH HÙNG
NGUYỄN VĂN LONG
LÊ ĐOÀN TRÀ MY
VŨ TUẤN SƠN

BÁO CÁO ĐỒ ÁN
HỆ THỐNG TÌM KIẾM, PHÁT HIỆN VÀ NGĂN NGỪA XÂM NHẬP
TRIỂN KHAI HỆ THỐNG IDPS PFSENSE & SURICATA

Thành phố Hồ Chí Minh, tháng 05 năm 2024

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

NGUYỄN MẠNH HÙNG – MSSV: 21520896
NGUYỄN VĂN LONG – MSSV: 21521097
LÊ ĐOÀN TRÀ MY – MSSV: 21521149
VŨ TUẤN SƠN – MSSV: 21521389

BÁO CÁO ĐỒ ÁN
HỆ THỐNG TÌM KIẾM, PHÁT HIỆN VÀ NGĂN NGỪA XÂM NHẬP
TRIỂN KHAI HỆ THỐNG IDPS PFSENSE & SURICATA

GIẢNG VIÊN HƯỚNG DẪN
ThS. ĐỖ HOÀNG HIỂN

Thành phố Hồ Chí Minh, tháng 05 năm 2024

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin phép gửi lời cảm ơn sâu sắc đến tập thể quý thầy cô trường Đại học Công nghệ Thông tin - Đại học Quốc gia TP.HCM, quý thầy cô khoa Mạng máy tính & Truyền thông đã tạo điều kiện, giúp chúng em học tập và có được những kiến thức nền tảng cũng như cung cấp các tài nguyên liên quan để hoàn thành được dự án này.

Đặc biệt, chúng em xin gửi lời cảm ơn chân thành đến thầy Đỗ Hoàng Hiển. Với sự tâm huyết, được sự tận tình giảng dạy và hỗ trợ hết lòng của thầy đã cho chúng em nhiều kiến thức bổ ích. Với tình cảm sâu sắc, chân thành, chúng em xin bày tỏ lòng biết ơn đến thầy đã nhiệt tình, hết mình với sinh viên. Đó là động lực rất lớn để chúng em có thể hoàn thành tốt đồ án lần này.

Với điều kiện thời gian cũng như kinh nghiệm còn hạn chế, chúng em đã cố gắng hết mình nhưng đồ án không thể tránh được những thiếu sót. Chúng em rất hy vọng nhận được sự chỉ bảo và đóng góp ý kiến từ thầy để bổ sung, nâng cao kiến thức của mình, phục vụ và hoàn thiện hơn trong những đồ án sau này và khóa luận tốt nghiệp trong lương lai.

Chúng em xin chân thành cảm ơn!

Nhóm thực hiện

MỤC LỤC

TÓM TẮT ĐỒ ÁN

Trong phạm vi đồ án môn học, nhóm thực hiện nghiên cứu về hệ thống IDPS, pfSense và Suricata. Nhóm tiến hành xây dựng và triển khai một mô hình IDPS đơn giản được xây dựng với 4 thành phần: attacker, hệ thống IDPS (kết hợp pfSense và Suricata), host (victim) và webserver. Sau khi tìm hiểu sâu về các công cụ, nhóm tiến hành thực hiện một số kịch bản như: tấn công brute force nhằm vào máy webserver, tấn công SQL injection nhằm vào ứng dụng DVWA trên máy webserver, tấn công khai thác lỗ hổng dịch vụ Samba trên máy victim để tạo một kết nối TCP reverse shell từ máy victim đến máy attacker, tấn công nhằm khai thác lỗ hổng trong phiên bản bị lỗi của dịch vụ VSFTPD để tạo một backdoor trên máy victim, cuộc tấn công Command Execution trên máy victim để thực thi lệnh hệ thống từ xa để tiến hành kiểm tra các rules và hệ thống có hiệu quả trong việc phát hiện và ngăn chặn các cuộc tấn công mạng phổ biến hay không. Các thử nghiệm thực tế với các kịch bản tấn công khác nhau đã cho thấy hệ thống không chỉ có khả năng phát hiện sớm mà còn ngăn chặn kịp thời, giúp giảm thiểu rủi ro và thiệt hại tiềm tàng. Tuy nhiên, hệ thống cũng bộc lộ một số hạn chế như khó khăn trong việc xử lý các cuộc tấn công phức tạp và yêu cầu cấu hình phần cứng, phần mềm phức tạp để duy trì hiệu suất hoạt động trong môi trường mạng lớn.

DANH MỤC HÌNH ẢNH

CHƯƠNG 1: GIỚI THIỆU

1.1 Giới thiệu vấn đề

Trong thời đại số hoá mạnh mẽ, mạng máy tính đã trở thành một phần vô cùng quan trọng trong chia sẻ thông tin, truy cập vào các tài nguyên và tương tác giao tiếp truyền thông giữa người với người trên toàn cầu nói chung, và trong các tổ chức, doanh nghiệp nói riêng. Tuy nhiên đi cùng với sự phát triển mạnh mẽ này là mối đe dọa và rủi ro bảo mật mạng cũng tăng lên đáng kể. Các cuộc tấn công mạng ngày càng trở nên phức tạp và tinh vi, gây thiệt hại nghiêm trọng cho cá nhân, tổ chức và xã hội.

Đứng trước thách thức lớn này, hệ thống phát hiện và ngăn ngừa xâm nhập (Intrusion Detection and Prevention System - IDPS) đã ra đời nhằm đáp ứng nhu cầu bảo mật mạng hiện đại. IDPS là hệ thống bảo mật mạng được sử dụng tập trung chủ yếu vào giám sát và phân tích lưu lượng mạng, xác định các sự cố có thể xảy ra, ghi nhật ký thông tin về các sự cố, cố gắng ngăn chặn và đưa ra cảnh báo cho quản trị viên.

Trong số các giải pháp IDPS hiện có, hệ thống được xây dựng từ pfSense và Suricata là một trong những lựa chọn đáng cân nhắc. Pfsense, là một tường lửa mã nguồn mở với tính linh hoạt và khả năng tùy chỉnh cao, kết hợp với Suricata, một giải pháp phát hiện xâm nhập dựa trên luật, tạo ra một hệ thống IDPS mạnh mẽ và hiệu quả. Pfsense giúp kiểm soát và quản lý lưu lượng mạng, trong khi Suricata phân tích và phát hiện các cuộc tấn công dựa trên các luật đã được xác định sẵn. Sự kết hợp này cung cấp cho người dùng khả năng bảo mật mạng cao hơn, đồng thời cung cấp thông tin chi tiết và cảnh báo để quản trị viên có thể đưa ra các biện pháp ngăn chặn kịp thời và bảo vệ mạng máy tính khỏi các mối đe dọa và tấn công ngày càng tinh vi.

1.2 Mục tiêu đề tài

Trong phạm vi đồ án môn học, nhóm hi vọng đồ án này sẽ cung cấp hiểu biết sâu hơn về hệ thống IDPS, pfSense và Suricata, cũng như phát triển kỹ năng triển khai và cài đặt, cấu hình các công cụ này để tiến hành triển khai một mô hình IDPS đơn giản, biết về cấu trúc luật và các thành phần trong luật của Suricata để xác định và thiết lập các quy tắc phát hiện xâm nhập phù hợp, và kiểm tra tính hiệu quả của hệ

thống với một số kịch bản thử nghiệm.

1.3 Phương pháp nghiên cứu

Trong đồ án Triển khai hệ thống IDPS pfSense & Suricata, nhóm sẽ thực hiện:

- *Tìm hiểu:* Tiến hành nghiên cứu sâu về hệ thống IDPS, pfSense và Suricata để hiểu về cấu trúc, tính năng và cách hoạt động của chúng để xác định được lợi ích và ứng dụng của hệ thống IDPS trong việc phát hiện và ngăn chặn xâm nhập mạng.
- *Thiết kế:* Dựa trên kiến thức đã tìm hiểu, tiến hành thiết kế một mô hình triển khai đơn giản cho hệ thống IDPS sử dụng pfSense và Suricata.
- *Triển khai và cấu hình:* Sau khi hoàn thành thiết kế, thực hiện triển khai và cấu hình hệ thống IDPS theo mô hình đã được xác định. Việc triển khai bao gồm: cài đặt pfSense và Suricata, thiết lập các quy tắc và luật phát hiện xâm nhập, cấu hình các thiết lập mạng và bảo mật liên quan.
- *Kiểm thử và đánh giá:* Tiến hành kiểm thử và đánh giá hệ thống IDPS đã triển khai với các kịch bản tấn công. Dựa vào kết quả kiểm tra khả năng phát hiện và ngăn chặn của hệ thống, thực hiện đánh giá hiệu suất và độ tin cậy của IDPS.
- *Đề xuất cải tiến:* Dựa trên kết quả kiểm thử và đánh giá, nhóm sẽ đề xuất các cải tiến và điều chỉnh để nâng cao khả năng phát hiện và ngăn chặn của hệ thống IDPS.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VỀ IDPS & CÁC CÔNG CỤ PFSENSE, SURICATA

2.1 IDPS

2.1.1 Sự xâm nhập

a, Khái niệm:

Xâm nhập là những hành động trái phép nhằm xâm phạm tính bảo mật, tính toàn vẹn và tính khả dụng hoặc vượt qua các cơ chế bảo mật của máy tính hoặc mạng.

b, Quy trình xâm nhập thường bao gồm các bước sau:

- *Trinh sát (Reconnaissance):* Kẻ tấn công thu thập thông tin về hệ thống mục tiêu, như địa chỉ IP, hệ điều hành, dịch vụ, lỗ hổng bảo mật,…
- *Quét (Scanning):* Kẻ tấn công sử dụng các công cụ quét để tìm kiếm các lỗ hổng có thể khai thác trong hệ thống.
- *Khai thác (Exploitation)*: Kẻ tấn công sử dụng các kỹ thuật và công cụ để khai thác các lỗ hổng được tìm thấy trong bước quét.
- *Nâng quyền (Privilege Escalation)*: Kẻ tấn công cố gắng nâng cao đặc quyền của mình, từ người dùng bình thường lên thành người dùng có đặc quyền cao hơn.
- *Duy trì truy cập (Maintaining Access)*: Kẻ tấn công cố gắng duy trì truy cập vào hệ thống bằng cách cài đặt backdoor hoặc rootkit.
- *Xóa dấu vết (Covering Tracks)*: Kẻ tấn công xóa các dấu vết của hoạt động xâm nhập để tránh bị phát hiện.

c, Một số nguyên nhân dẫn đến sự xâm nhập

Có rất nhiều nguyên nhân dẫn đến sự xâm nhập như: các phần mềm độc hại – malware (ví dụ: worms, spyware, trojan, ransomware, …); kẻ tấn công truy cập trái phép vào hệ thống qua mạng Internet; người dùng hợp lệ của hệ thống lợi dụng quyền hạn hoặc cố chiếm thêm một số quyền không được phép.

d, Các dấu hiệu của xâm nhập

- Một số dấu hiệu chung về xâm nhập: Log ngắn và không đầy đủ; Hiệu suất hệ thống thấp bất thường; Các tiến trình bất thường; Hệ thống bị crash hoặc reboot; Hiển thị hình ảnh hoặc đoạn tin nhắn bất thường.

- Một số dấu hiệu khi có xâm nhập hệ thống: Xuất hiện các file hoặc chương trình lạ; Các quyền truy cập file bị thay đổi; Kích thước file bị thay đổi bất thường; Những tên file lạ trong các thư mục; Thiếu file.
- Một số dấu hiệu khi có xâm nhập mạng: Thăm dò liên tục các service trên các máy tính; Kết nối từ các vị trí bất thường; Cố gắng đăng nhập liên tục từ host ở xa; Dữ liệu bất thường trong các file log, dấu hiệu của DoS hoặc hướng tới crash dịch vụ.

2.1.2 IDPS là gì?

IDPS (Intrusion detection and prevention systems – Hệ thống phát hiện và ngăn chặn xâm nhập) là sự kết hợp của IDS (Intrusion Detection System – Hệ thống phát hiện xâm nhập) và IPS (Intrusion Prevention System – Hệ thống ngăn chặn xâm nhập). Trong đó, IDS là hệ thống phần mềm hoặc phần cứng tự động thực hiện quy trình phát hiện xâm nhập, sẽ tiến hành theo dõi các sự kiện diễn ra trong một hệ thống máy tính hoặc mạng máy tính và phân tích để nhận biết các dấu hiệu của sự bất thường (hành vi xâm nhập – intrusion. IPS là hệ thống phần mềm hoặc phần cứng có tất cả chức năng của IDS và có thể dừng sự xâm nhập bằng nhiều cách khác nhau, chẳng hạn như chặn IP nguồn, drop traffic,…

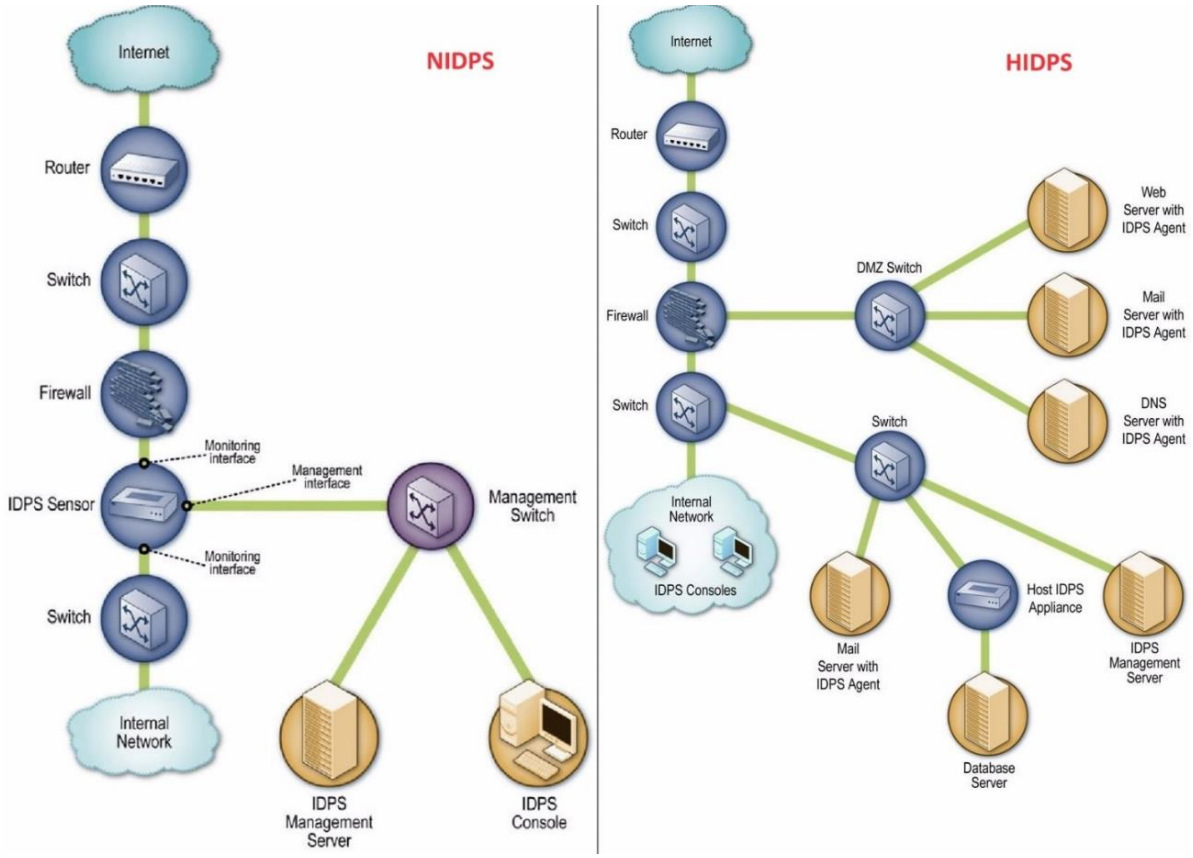
2.1.3 Các dạng IDPS

a, Phân loại dựa trên Nguồn dữ liệu và vị trí: IDPS được chia làm 3 loại chính

- Network-based IDPS (NIDPS): giám sát lưu lượng mạng cho một phần của mạng (network segment) hoặc các thiết bị, phân tích các hoạt động mạng và các giao thức, ứng dụng để xác định các hành vi bất thường. NIDPS thường triển khai ở biên mạng, như gần tường lửa hoặc router biên, server VPN, server remote access và mạng không dây. Nó gồm nhiều sensor đặt ở nhiều điểm khác nhau trong mạng để theo dõi lưu lượng mạng.

- Host-based IDPS (HIDPS): giám sát các đặc điểm của một host riêng lẻ và các sự kiện xảy ra trong host đó để phát hiện hoạt động bất thường, ví dụ: lưu lượng truy cập mạng (chỉ của host đó), nhật ký hệ thống, tiến trình đang chạy, hoạt động ứng dụng, truy cập và sửa đổi tệp, những thay đổi cấu hình của hệ thống, ứng dụng. HIDPS được

triển khai trên host quan trọng (các server có thể truy cập từ bên ngoài, các server chứa thông tin quan trọng).

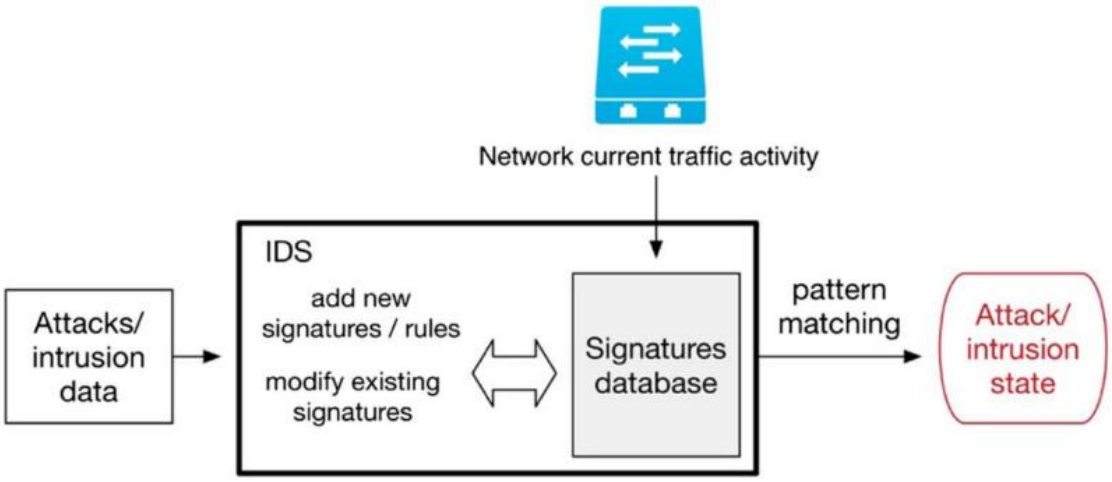


Hình 1: Mô hình triển khai NIDPS vs Mô hình triển khai HIDPS

- Hybird IDPS: được phát triển dựa trên dữ liệu được cung cấp bởi các sự kiện trên host và các phân đoạn mạng, đồng thời kết hợp những chức năng của cả 2 loại NIDPS và HIDPS.

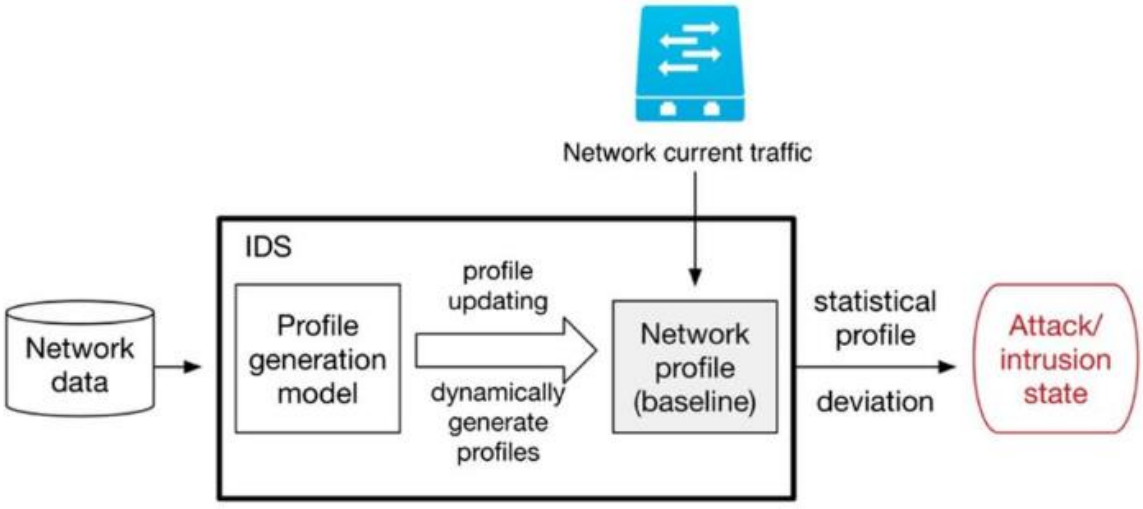
b, Phân loại dựa trên Các kỹ thuật phát hiện: IDPS được chia thành 4 loại chính

- Signature-based (hoặc misuse, knowledge-based): So sánh dữ liệu với các signature (mẫu - nguy cơ tấn công) đã biết. Ưu điểm là độ chính xác phát hiện cao đối với các mối đe dọa đã biết, tỷ lệ cảnh báo sai thấp. Tuy nhiên, nhược điểm là độ chính xác phụ thuộc vào cơ sở dữ liệu signature, không thể phát hiện những bất thường chưa xác định hoặc những biến thể của cuộc tấn công đã biết nên cần phải cập nhật liên tục cơ sở dữ liệu signature. Việc triển khai và cập nhật signature khó và tốn thời gian.



Hình 2 - Sơ đồ của kỹ thuật phát hiện signature-based

- Anomaly-Based (profile-based): Tạo và so sánh với baseline profile để xác định hành vi bất thường. Profiles đại diện cho hoạt động mạng bình thường hầu hết được tạo ra thông qua phân tích lịch sử lưu lượng mạng do đó bất kỳ hoạt động mạng đang xem xét nào có sai khác so với profile này đều bị xem là bất thường. Ưu điểm là có thể phát hiện được bất thường đã và chưa biết cũng như phát hiện được những tấn công mới. Hạn chế là tỷ lệ cảnh báo sai và bỏ sót cao, tiêu tốn nhiều tài nguyên.



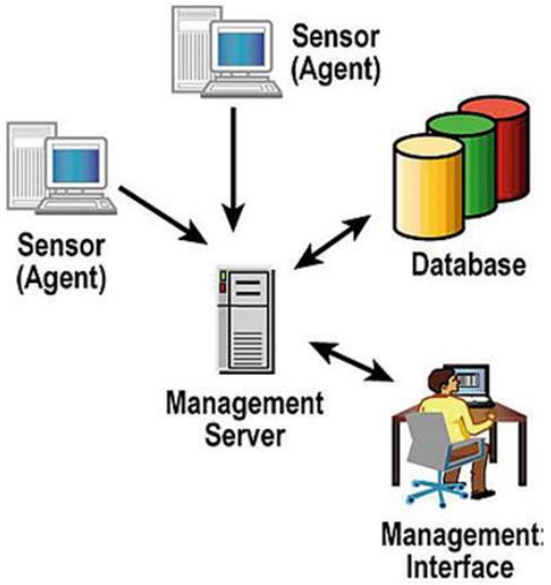
Hình 3 - Sơ đồ của kỹ thuật phát hiện Anomaly-Based

- Specification-based: Giám sát các chương trình đang thực thi để phát hiện bất thường từ các ràng buộc, các thông số kỹ thuật hợp lệ tương ứng. Ưu điểm là xác định được các chuỗi lệnh bất thường, kiểm tra được tính hợp lý của từng câu lệnh, tỷ lệ false

positive thấp. Tuy nhiên, nhược điểm là khó, thậm chí không thể phát triển các mô hình giao thức chính xác hoàn toàn; phức tạp, tốn tài nguyên và thời gian.

- *Hybrid:* Kết hợp các loại trên. Ưu điểm mang lại là chống được những tấn công có thay đổi tinh vi, tổng hợp những lợi ích của mỗi loại và khắc phục được nhiều nhược điểm. Tuy nhiên, nó bị giới hạn phạm vi vào hoạt động của một chương trình giao thức, độ phức tạp cao khi tích hợp 3 kỹ thuật riêng biệt có thể cùng tương tác và hoạt động trong cùng một hệ thống.

2.1.4 Các thành phần chính của IDPS



Hình 4 – Một số thành phần chính của IDPS

- *Sensor hoặc Agent:* module theo dõi, thu thập và phân tích các hoạt động.

+ Sensor: Network-based IDPS

+ Agent: Host-based IDPS

- *Server quản lý:* thiết bị trung tâm nhận các thông tin từ các sensor hoặc agent để quản lý; phân tích thông tin và kết luận có là tấn công hay không.

- *Server cơ sở dữ liệu:* nơi lưu trữ các thông tin sự kiện theo dõi được bởi các sensor hay agent và server quản lý (optional).

- *Management interface (Consoles):* giao diện tương tác với IDPS cho người dùng hoặc quản trị viên (GUI or CLI), dùng để giám sát các sự kiện hệ thống mà IDPS nhận được….

**** Các thành phần của IDPS có thể được kết nối với nhau thông qua:***

- *Các **mạng chuẩn** (standard networks) của tổ chức, hoặc*
- *Một mạng tách biệt được thiết kế riêng cho việc quản lý an toàn thông tin - **mạng quản lý**. Ưu điểm của mạng quản lý: Độc lập đối với mạng sản xuất của doanh nghiệp; Che giấu được sự tồn tại và dấu hiệu của IDPS với attacke; Bảo vệ IDPS khỏi tấn công và đảm bảo IDPS có đủ băng thông để hoạt động trong điều*

kiện bất lợi (do tấn công). Tuy nhiên, nó cần thêm chi phí cho hạ tầng mạng và các phần cứng khác, bất tiện cho người dùng và quản trị viên IDPS.

2.1.5 Các hoạt động của IDPS

- Thu thập thông tin: Thu thập thông tin trên host hoặc mạng từ các hoạt động quan sát được.

Ví dụ: OS, các host, các ứng dụng được dùng, xác định các đặc điểm chung của mạng,...

- Ghi log:

+ Ghi log các dữ liệu liên quan đến sự kiện phát hiện được

+ Ghi log các dữ liệu có thể được dùng để kiểm tra tính hợp lệ của cảnh báo, điều tra các sự số và liên kết các sự kiện giữa IDPS và các nguồn log khác.

+ Các trường dữ liệu phổ biến: thời gian xảy ra sự kiện, loại sự kiện, mức độ quan trọng (ví dụ độ ưu tiên, mức độ nghiêm trọng, tác động, độ tin cậy), hành động ngăn chặn đã thực hiện (nếu có), gói tin bắt được (NIDS), user ID (HIDS)

+ Thông thường, log nên được lưu trữ ở cả nội bộ và tập trung để đảm bảo tính toàn vẹn và sẵn sàng của dữ liệu.

- Phát hiện:

+ Sử dụng ngưỡng (thresholds): thiết lập 1 giá trị giới hạn giữa hành vi bình thường và bất thường, xác định mức độ tối đa có thể chấp nhận được (là bình thường).

Ví dụ: số lần đăng nhập lỗi trong một khoảng thời gian, độ dài tên file,...

→ Thường được dùng cho **kỹ thuật phát hiện anomaly-based** và **phân tích các stateful protocol.**

+ Sử dụng danh sách đen (blacklists/hot lists) và danh sách trắng (whitelists): để nhận dạng và chặn các hoạt động độc hại hoặc xác định các hoạt động bình thường. Blacklists có liên quan đến hoạt động độc hại, được dùng để nhận dạng và chặn các hoạt động có nguy cơ cao là có hại, và cũng dùng để gán độ ưu tiên cho hơn cho các cảnh báo có liên quan đến blacklist. Whitelist là liên quan đến hoạt động bình thường, được sử dụng nhằm giảm hoặc loại bỏ các trường hợp

false positive (dương tính giả) bao gồm các hoạt động bình thường từ các host đáng tin cậy.

Ví dụ về các thực thể: hosts, port TCP hay UDP, ICMP types và codes, ứng dụng, usernames, URLs, tên file, hay phần mở rộng file

→ Thường sử dụng cho kỹ **thuật phát hiện signature-based** và **phân tích các stateful protocol**.

+ Thiết lập cảnh báo:

- Bật hoặc tắt chức năng cảnh báo.
- Thiết lập mức độ ưu tiên và nghiêm trọng của cảnh báo.
- Xác định thông tin cần ghi lại và phương pháp thông báo (email, SMS,…)
- Xác định khả năng ngăn chặn của hệ thống.
- Ngưng cảnh báo nếu attacker tạo nhiều cảnh báo trong một khoảng thời gian ngắn và tạm thời bỏ qua lưu lượng từ attacker.

+ Xem và chỉnh sửa mã nguồn:

- Xem và chỉnh sửa mã nguồn liên quan đến phát hiện tấn công để tùy chỉnh khả năng phát hiện và giảm false positive.
- Kiểm tra, điều chỉnh các thiết lập định kỳ, đảm bảo hiệu quả của hệ thống.
- Kiểm tra và cập nhật whitelist và blacklist định kỳ.
- Sao chép lại các thay đổi trong mã nguồn khi có cập nhật sản phẩm.

- Ngăn chặn:

+ IDPS thường cho phép quản trị viên cấu hình khả năng ngăn chặn cho mỗi loại cảnh báo mà nó đưa ra.

+ Thường bao gồm bật/tắt khả năng ngăn chặn, cũng như chỉ định rõ loại khả năng ngăn chặn nào nên dùng.

2.2 pfSense

2.2.1 Giới thiệu pfSense

pfSense là nền tảng Firewall mã nguồn mở miễn phí, được xây dựng dựa trên hệ điều hành FreeBSD. Ngoài chức năng tường lửa, pfSense còn có đầy đủ các chức

năng định tuyến cao cấp của 1 Router chuyên dụng. Pfsense được cấu hình qua giao diện GUI trên web nên có thể quản lý, tùy chỉnh dễ dàng. Nó hỗ trợ lọc theo địa chỉ nguồn, đích, port nguồn hay port đích, hỗ trợ định tuyến và có thể hoạt động trong chế độ bridge hay transparent.

2.2.2 Một số tính năng của pfSense

- Aliases: gom nhóm các ports, host hoặc Network(s) khác nhau và đặt cho chúng một cái tên chung để thiết lập những quy tắc được dễ dàng và nhanh chóng hơn.
- Rules: Lưu các rules của Firewall, mặc định pfSense cho phép mọi traffic ra/vào hệ thống.
- NAT: cấu hình các thiết lập NAT nếu cần sử dụng cổng chuyển tiếp cho các dịch vụ hoặc cấu hình NAT tĩnh (1:1) cho các host cụ thể.
- Firewall schedules: lập lịch cho các quy tắc tường lửa hoạt động trong các khoảng thời gian cụ thể nhằm linh hoạt và quản lý hiệu quả cho các quy tắc tường lửa, tăng cường bảo mật và quản lý lưu lượng mạng.
- Traffice shapper (Quản lý băng thông): cho phép ưu tiên, giới hạn và ổn định lưu lượng mạng để đảm bảo sự công bằng và hiệu suất tốt cho các ứng dụng và dịch vụ trên mạng.
- Virtual IPs (VIP): chuyển tiếp lưu lượng cho những việc như chuyển tiếp cổng NAT, NAT Outbound và NAT 1:1; cho phép các tính năng như failover (chế độ dự phòng), và có thể cho phép dịch vụ trên router để gắn kết với địa chỉ IP khác nhau.
- VPN (Virtual Private Network): hỗ trợ nhiều giao thức VPN như IPsec, OpenVPN, L2TP, PPTP; cho phép tạo các kết nối VPN an toàn.
- Proxy ARP (Address Resolution Protocol): máy chủ proxy ARP trả lời các yêu cầu ARP thay mặt cho các máy chủ khác trong mạng; khi đó, pfSense như một gateway mạng và chuyển tiếp các yêu cầu từ một mạng nội bộ đến một mạng công cộng hoặc Internet, thực hiện chức năng NAT.
- CARP (Common Address Redundancy Protocol): CARP là một giao thức đồng bộ hóa địa chỉ IP được sử dụng để cung cấp tính sẵn sàng và cân bằng tải cho các dịch vụ mạng

để tạo cluster các thiết bị pfSense để cung cấp các dịch vụ mạng có tính sẵn sàng cao.
- Quản lý danh sách điều khiển truy cập (Access Control List - ACL): kiểm soát quyền truy cập Internet của người dùng trong mạng, như chặn truy cập vào các trang web đặc biệt đến việc quản lý thời gian truy cập.
- Bảo mật: hỗ trợ nhiều tính năng bảo mật như IDS/IPS (Intrusion Detection/Prevention System), DPI (Deep Packet Inspection),…

2.2.3 Một số dịch vụ

- Captive portal: cho phép admin có thể chuyển hướng client tới một trang web khác,từ trang web này client có thể phải chứng thực trước khi kết nối tới internet. Tính năng captive portal nằm ở mục Services/captive portal.
- DHCP Server: Dịch vụ này cho phép pfSense cấp địa chỉ IP và các thông tin cấu hình cho client trong mạng LAN. Tính năng này nằm trong Services/DHCP server .
- DHCP replay: cho phép pfSense forward yêu cầu cấp IP của client nằm trong một subnet nào đó tới một DHCP server cho trước. Chỉ được phép chạy một trong hai dịch vụ DHCP server và DHCP relay.
- Server Load Balancing: điều phối mạng hay còn gọi là cân bằng tải. Có 2 loại load balancing trên pfSense:
    + Gateway load balancing: được dùng khi có nhiều kết nối WAN. Client bên trong LAN khi muốn kết nối ra ngoài Internet thì pfSense lựa chọn card WAN để chuyển packet ra card đó giúp cho việc cân bằng tải cho đường truyền.
    + Server load balancing: cho phép cân bằng tải cho các server của mình. Được dùng phổ biến cho các web server, mail server và server không hoạt động nữa thì sẽ bị xóa.
- Một số dịch vụ khác:
    + System log: theo dõi hoạt động của hệ thống pfSense và các dịch vụ mà pfSense cung cấp. Mọi hoạt động của hệ thống và dịch vụ đều được ghi lại.
    + System Status: Liệt kê các thông tin và tình trạng của hệ thống.
    + Service Status: Hiển thị trạng thái của tất cả các service có trong hệ thống. Mỗi

service có hai trạng thái là: running, stopped.

+ Interface Status: Hiển thị thông tin của tất cả card mạng.

+ RRD Graph: Hiển thị các thông tin dưới dạng đồ thị. Các thông tin mà RRD Graph sẽ thể hiện là: System, Traffic, Packet, Quality, Queues.

2.3 Suricata

2.3.1 Giới thiệu Suricata

Suricata là hệ thống phát hiện và ngăn chặn xâm nhập dựa trên mã nguồn mở. Suricata là công cụ IDS/IPS miễn phí, được phát triển bởi Open Information Security Foundation (OISF), dựa trên luật (rules) để theo dõi lưu lượng mạng, đưa ra cảnh báo nếu có sự kiện bất thường xảy ra. Nó được thiết kế để tương thích với các thành phần an ninh mạng hiện có. 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.

2.3.2 Một số tính năng và chức năng nổi bật của Suricata

a, Các tính năng chính của Suricata bao gồm:

- Suricata có khả năng bắt gói tin ở cả chế độ không lưu trạng thái (stateless) và lưu trạng thái (stateful). Cung cấp khả năng kiểm tra mỗi gói tin độc lập hoặc dựa trên thông tin kết nối trước đó.
- Hỗ trợ nhiều giao thức khác nhau như ICMP, DNS, HTTP, TLS, FTP....
- Cung cấp nhiều tùy chọn cấu hình, quy tắc để tùy chỉnh theo nhu cầu của từng hệ thống, gồm: khả năng kích thích/giả mạo các cuộc tấn công để ghi log và phân tích,…
- Xử lý đa luồng mạnh mẽ, tăng khả năng mở rộng và hiệu suất của hệ thống trong môi trường mạng phức tạp
- Hệ thống ghi log đủ khả năng cung cấp các tệp tin log chi tiết về các sự kiện xâm nhập và các hoạt động của mạng, giúp nhận biết, phản ứng nhanh chóng, thực hiện truy vết đối với các mối đe dọa.
- Chạy được trên nhiều nền tảng: Linux, CentOS, Windows, MAC OS,..
- Các rules thường xuyên được bổ sung và cập nhật.

b, Các chức năng chính nổi bật của Suricata: Đa luồng (Multi threading), Phát hiện

giao thức tự động (Automatic Protocol Detection), Quản lý danh tiếng IP (IP Reputation).

2.3.3 Kiến trúc và hoạt động

Suricata hoạt động bằng cách giám sát lưu lượng mạng, phân tích các gói tin và so sánh chúng với bộ chữ ký (signatures) của các mối đe dọa đã biết để phát hiện và cảnh báo về các hoạt động bất thường. Quá trình này bao gồm các bước chính:

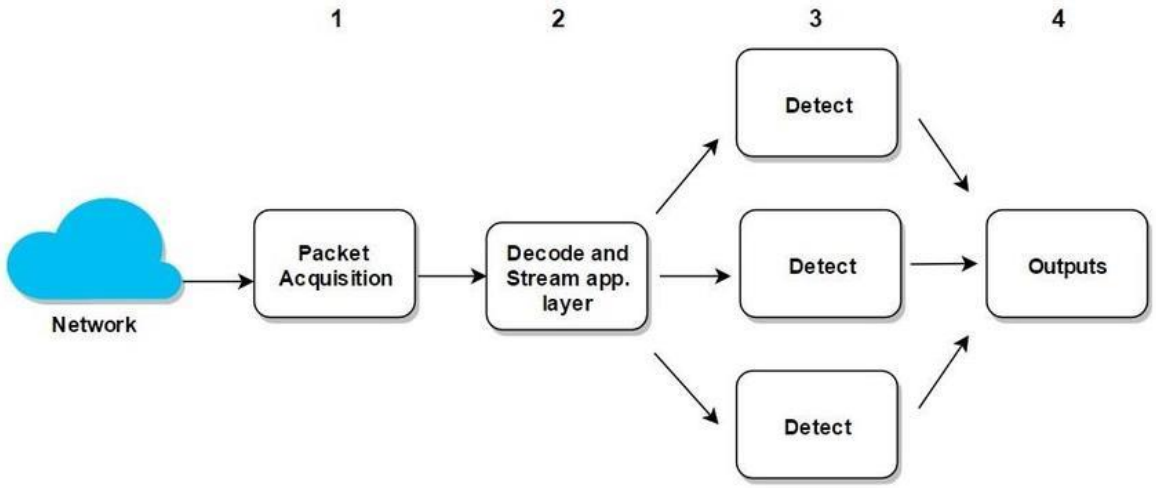
- Thu thập lưu lượng mạng (Network Traffic Acquisition): Suricata hoạt động ở chế độ sniffing hoặc chế độ promiscuous trên một giao diện mạng, để thu thập toàn bộ lưu lượng mạng đi qua.
- Phân tích và phân tách các gói tin (Packet Parsing and Analysis): Suricata sử dụng các thư viện bắt gói tin và phân tích để trích xuất thông tin từ các gói tin như tiêu đề (header), tải trọng (payload) và các cấu trúc dữ liệu cụ thể của giao thức.
- So sánh dữ liệu với bộ chữ ký để phát hiện mối đe dọa (Signature Matching): Suricata so khớp dữ liệu trích xuất từ gói tin với các chữ ký trong bộ quy tắc, để phát hiện các mối đe dọa đã biết.
- Thực hiện kiểm tra sâu gói tin (Deep Packet Inspection - DPI) (Tùy chọn): Suricata có thể được cấu hình để thực hiện kiểm tra gói tin sâu (DPI) để phát hiện các mối đe dọa có thể ẩn nội dung độc hại trong dữ liệu payload của gói tin.
- Hành động và ghi nhật ký: (Action and Logging): Khi phát hiện bất thường, Suricata kích hoạt các hành động được định nghĩa, như ghi nhật ký, cảnh báo, chặn lưu lượng.

Suricata bao gồm một số "khối xây dựng" - building blocks gồm luồng (threads), module luồng (thread-modules) và hàng đợi (queues). Threads tương tự như các tiến trình chạy trên máy tính, Suricata là ứng dụng đa luồng, sử dụng nhiều CPU/lõi CPU nên có thể xử lý đồng thời nhiều gói mạng tức sẽ có nhiều threads hoạt động cùng một lúc. Thread-modules là các phần chức năng riêng biệt của Suricata, như mô-đun giải mã gói tin, mô-đun phát hiện, mô-đun xuất kết quả,... Các gói tin được xử lý bởi một thread sẽ được chuyển qua thread tiếp theo thông qua các hàng đợi (queues).

Suricata xử lý và phát hiện chủ yếu dựa trên hoạt động của các thread và

thread-modules. Suricata có 4 thread-modules chính: Packet acquisition, decode and stream application layer, detection, and outputs.

Cụ thể như sau:



Hình 5 – Kiến trúc và hoạt động của Suricata

- Packet Acquisition: thu thập gói tin từ giao diện mạng, pcap,… và chuyển tiếp chúng đến để giải mã gói tin(decoder).
- Decoder & stream application layer:
    + Decoder: giải mã gói tin, xác đinh các loại liên kết, giao thức và chuẩn hóa dữ liệu cho các tiến trình khác.
    + Stream module: Thực hiện stream-tracking để đảm bảo tính toàn vẹn của kết nối mạng; nhóm các dạng dữ liệu và reassembly các gói dữ liệu thành luồng dữ liệu gốc; Phân tích các giao thức ứng dụng như HTTP và DCERPC và chuyển các dữ liệu đến model tiếp theo.
- Detect: nhận dữ liệu và phân tích gói tin từ Decoder & stream application layer để phát hiện các tân công mạng dựa trên các dấu hiệu (signature), và tạo cảnh báo khi phát hiện các hành vi tấn công. Có thể có nhiều luồng phát hiện chạy đồng thời.
- Outputs: xử lý và xuất kết quả của quá trình phát hiện tấn công, bao gồm các cảnh báo và sự kiện.

Mặc định, Suricata được cấu hình để chạy dưới dạng IDS, chỉ tạo cảnh báo và

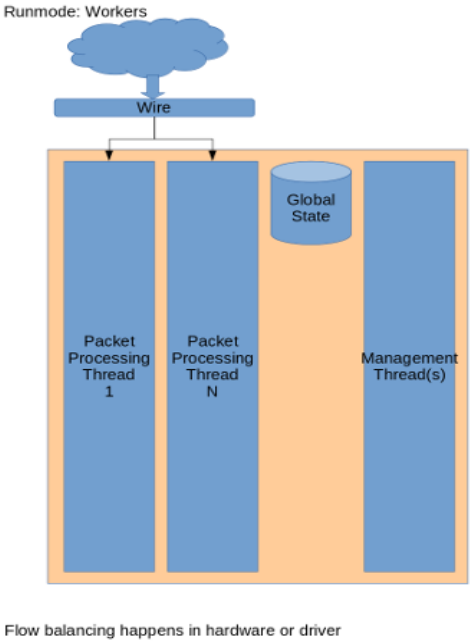
ghi nhật ký lại lưu lượng truy cập đáng ngờ. Khi bạn chế độ IPS, Suricata có thể chủ động loại bỏ lưu lượng truy cập mạng đáng ngờ ngoài việc chỉ tạo ra các cảnh báo

2.3.4 Runmodes - Các chế độ hoạt động của Suricata

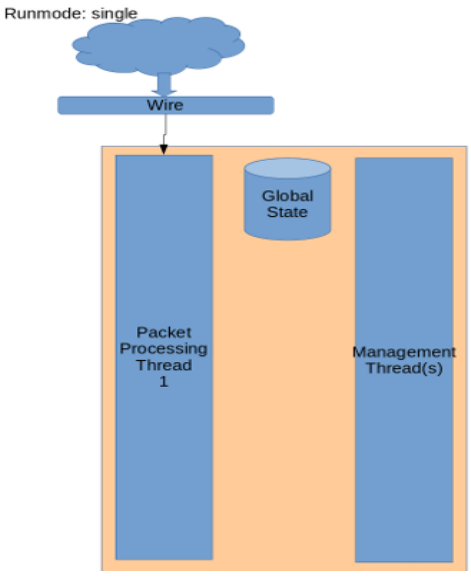
Cách sắp xếp các threads, thread-modules và queues được gọi là runmode (chế độ hoạt động). Suricata có 3 chế độ hoạt động, bao gồm: workers, single, autofp (có thể dùng lệnh --list-runmodes để hiển thị tất cả các runmodes có sẵn).

a, Chế độ hoạt động workers

- Là chế độ hoạt động phổ biến, có hiệu xuất xử lý tốt nhất của Suricata.
- Trong chế độ này, Suricata sử dụng nhiều luồng xử lý (worker threads) để tận dụng tối đa hiệu năng của nhiều lõi CPU.
- Các gói tin được NIC/driver phân phối cân bằng, hợp lý đến các luồng xử lý này thông qua các hàng đợi.
- Mỗi luồng xử lý bao gồm đầy đủ các module để thực hiện toàn bộ pipeline xử lý gói tin.



Hình 6 - Runmode workers

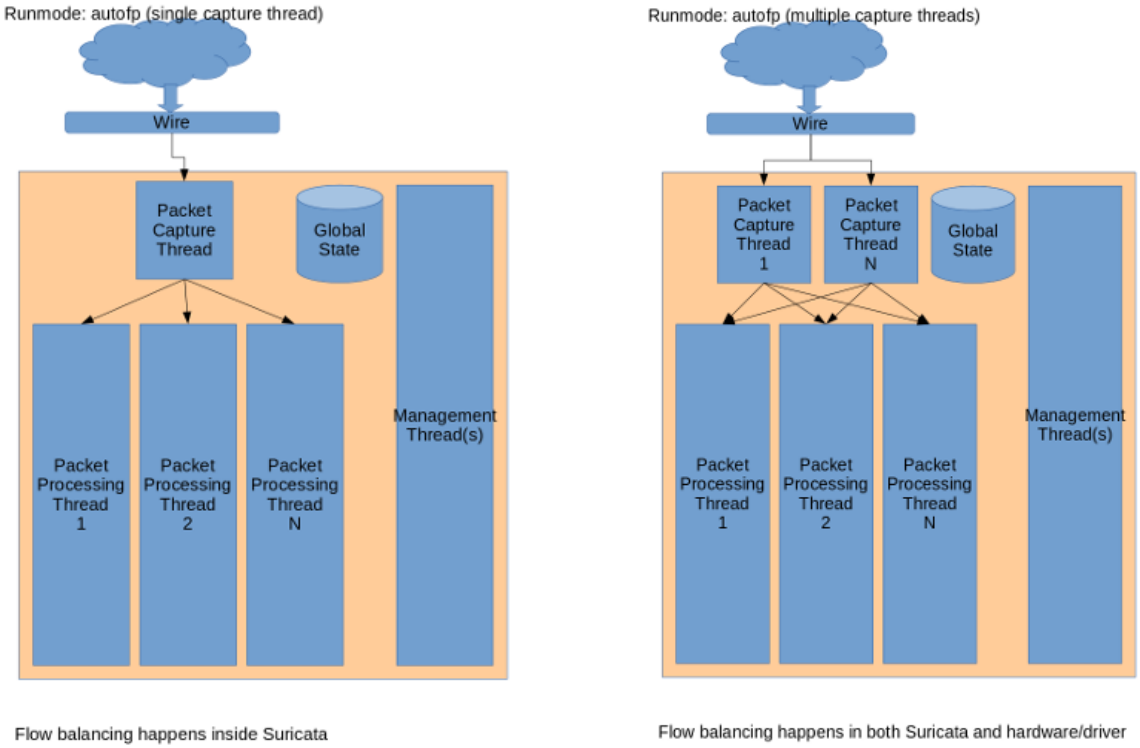


Hình 7 – Runmode single

b, Chế độ hoạt động single

- Đây là chế độ hoạt động đơn luồng, có một thread duy nhất thực hiện toàn bộ pipeline xử lý gói tin.
- Chế độ này đơn giản hơn và tiêu tốn ít tài nguyên hơn, nhưng hiệu suất xử lý thấp hơn so với chế độ Workers.
- Chế độ này thường được sử dụng cho các hệ thống có tài nguyên hạn chế hoặc trong quá trình phát triển/debugging.

c, Chế độ hoạt động autofb



Hình 8 – Single capture thread và multiple capture thread trong runmode autofp

- Suricata sử dụng một hoặc nhiều capture thread để bắt gói tin và giải mã chúng.
- Sau khi gói tin được giải mã, nó được chuyển đến các flow worker thread xử lý tiếp

+ Với Single capture thread:

- Chỉ có 1 luồng (thread) chịu trách nhiệm tiếp nhận tất cả các gói tin.
- Gói tin được tiếp nhận sẽ được chuyển vào các queue để các luồng xử lý (worker threads) khác xử lý.
- Điểm yếu là nếu luồng tiếp nhận gói tin bị quá tải, sẽ dẫn đến sự chậm trễ trong toàn bộ quá trình xử lý.

+ Với Multiple capture thread:

- Có nhiều luồng tiếp nhận gói tin, mỗi luồng sẽ tiếp nhận và xử lý một phần lưu lượng mạng để tránh tình trạng quá tải ở một luồng duy nhất.
- Các luồng tiếp nhận gói tin sẽ chuyển các gói tin vào các queue để các luồng xử lý (worker threads) khác xử lý.

- Điểm mạnh là có thể khai thác tối đa tài nguyên hệ thống, nâng cao hiệu suất xử lý tổng thể.

2.3.5 Rules - Quy tắc/luật trong Suricata

Rules đóng một vai trò rất quan trọng trong Suricata. Các quy tắc này giúp Suricata phân tích và phát hiện các hoạt động mạng bất thường hoặc nghi vấn. Một số vai trò chính của các quy tắc trong Suricata: Xác định và phát hiện các mối đe dọa, Phản ứng và ngăn chặn các cuộc tấn công, Lọc và phân loại lưu lượng mạng.

- Một quy tắc/chữ ký bao gồm:

+ Các Action, xác định những gì sẽ xảy ra khi các chữ ký khớp.

+ Các Header xác định giao thức, địa chỉ IP, cổng và hướng lưu lượng của các quy tắc.

+ Các Rules option, xác định các chi tiết cụ thể của quy tắc.

- Ví dụ về một quy tắc:

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"HTTP GET Request
Containing Rule in URI"; flow:established,to_server; http.method;
content:"GET"; http.uri; content:"rule"; fast_pattern; classtype:bad-
unknown; sid:123; rev:1;)
```

a, Action: xác định điều gì sẽ xảy ra khi quy tắc phù hợp

Suricata hỗ trợ một số các Action như sau:

alert	Tạo cảnh báo khi quy tắc được kích hoạt
pass	Cho phép gói tin đi qua mà không cần xử lý thêm
drop	Drop gói tin và tạo cảnh báo
reject	Drop gói tin và gửi lỗi không kết nối RST/ICMP tới người gửi
rejectsrc	Drop gói tin và gửi lỗi không kết nối RST/ICMP tới người gửi
rejectdst	Drop gói tin và gửi lỗi không kết nối RST/ICMP tới người nhận
rejectboth	Drop gói tin và gửi lỗi không kết nối RST/ICMP tới người gửi và nhận

b, Header: xác định protocol (giao thức), IP Adrress (địa chỉ IP), Port (cổng) và Direction (hướng lưu lượng) của quy tắc.

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"HTTP GET Request
Containing Rule in URI"; flow:established,to_server; http.method;
content:"GET"; http.uri; content:"rule"; fast_pattern; classtype:bad-
unknown; sid:123; rev:1;)
```

- Protocol (giao thức):

+ Có 4 giao thức cơ bản: *tcp, udp, icmp, ip* (ip là viết tắt của tất cả (all) hoặc bất kỳ (any)). Có một số tuỳ chọn giao thức liên quan đến TCP bổ sung như:

- tcp-pkt (để khớp nội dung trong các gói tcp riêng lẻ)
- tcp-stream (chỉ để khớp nội dung trong luồng tcp được tập hợp lại)

+ Ngoài ra còn có một số giao thức lớp ứng dụng layer7 như: http, http1, http2, ftp, tls (gồm cả ssl), smb, dns, ssh, smtp, imap, ntp, dhcp, rfp, snmp, tftp, sip, websocket, dcerpc….

Tính khả dụng của các giao thức này phụ thuộc vào việc giao thức có được bật trong tệp cấu hình suricata.yaml hay không.

- IP Address (địa chỉ IP)/Địa chỉ nguồn và đích:

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"HTTP GET Request
Containing Rule in URI"; flow:established,to_server; http.method;
content:"GET"; http.uri; content:"rule"; fast_pattern; classtype:bad-
unknown; sid:123; rev:1;)
```

+ Có thể chỉ định địa chỉ IP (cả IPv4 và IPv6 đều được hỗ trợ) và dải IP. Chúng có thể được kết hợp với các toán tử

Toán tử	Miêu tả
../..	Dải IP
!	Ngoại lệ, phủ định
[.., ..]	Phân nhóm

Ví dụ: !1.1.1.1 có nghĩa mọi địa chỉ IP trừ 1.1.1.1; [10.0.0.0/24, !10.0.0.5] có nghĩa 10.0.0.0/24 ngoại trừ 10.0.0.5,…

+ Ngoài ra, có thể sử dụng các biến, chẳng hạn như $HOME_NET và $EXTERNAL_NET. Trong tệp cấu hình suricata.yaml chỉ các địa chỉ IP có liên quan. Các

cài đặt $HOME_NET và $EXTERNAL_NET tương ứng sẽ được sử dụng thay cho các biến trong rules.

- *Port (cổng):* Xác định port nào đang nhận gói và port nào gửi gói tin

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"HTTP GET Request
Containing Rule in URI"; flow:established,to_server; http.method;
content:"GET"; http.uri; content:"rule"; fast_pattern; classtype:bad-
unknown; sid:123; rev:1;)
```

+ Ngoài việc gán cổng numbered, bạn cũng có thể sử dụng các toán tử đặc biệt

Toán tử	Miêu tả
:	Dãy cổng
!	Ngoại lệ, phủ định
[.., ..]	Phân nhóm

Ví dụ: [80, 81, 82] có nghĩa là cổng 80, 81 và 82; [1024:] có nghĩa là từ cổng 1024 đến số cổng cao nhất; [1:80,![2,4]] có nghĩa là phạm vi từ 1-80, ngoại trừ cổng 2 và 4,....

- *Direction (hướng lưu lượng):* xác định hướng đi của gói tin khi được phân tích

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"HTTP GET Request
Containing Rule in URI"; flow:established,to_server; http.method;
content:"GET"; http.uri; content:"rule"; fast_pattern; classtype:bad-
unknown; sid:123; rev:1;)
```

- -> : chỉ những gói có cùng hướng mới có thể khớp với quy tắc.
- <>: quy tắc áp dụng cho cả hai hướng đi của gói tin.
- Lưu ý: Không có hướng kiểu 'đảo ngược', tức là không có <-.

c, Rules option: xác định các chi tiết cụ thể của quy tắc, là trung tâm của việc phát hiện xâm nhập.

- Các tuỳ chọn của rules được bọc bởi dấu ngoặc đơn "()", tất cả các rules options sẽ được phân cách nhau bởi dấu chấm phẩy ";", phần tối số sẽ được tách ra bởi dấu hai chấm ":". Các tuỳ chọn có một thứ tự cụ thể, việc thay đổi thứ tự của chúng có thể làm thay đổi ý nghĩa của rules.

d, Một số thành phần khác về rules option

- Meta-settings:

+ msg (message): Được sử dụng để mô tả cảnh báo cho hệ thống, nó sẽ hiển thị khi người phân tích nhận các cảnh báo.

+ sid(signature id): Được sử dụng để xác định tính duy nhất các rules được tạo. Mỗi rules có một giá trị duy nhất (là số) và không trùng nhau. Một dải sid được xác định như sau:

- 0-1000000: Sử dụng bởi sourcefire VRT
- 2000001-2999999: Sử dụng bởi Emerging Threats
- 3000000+: Sử dụng khi tạo rules mới

+ rev (Revision): đại diện cho các phiên bản của signature, xác định khi một rules được thay đổi. Mỗi khi signature được sửa đổi thì số rev sẽ được tăng lên

+ gid (Group id): Gid giống như sid, Gid chỉ có thể nhận biết được bằng cách xem log cảnh báo

+ reference: Cung cấp thêm thông tin về luật được tạo. Thông thường nó được thêm vào là các đường dẫn ra ngoài cung cấp cụ thể về thông tin các luật được tạo

- Ví dụ: reference: url, www.info.nl
- Một số định dạng cho reference

```
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       (No longer available but you might still encounter this in signatures.)
                http://www.whitehats.com/info/IDS
mcafee          http://vil.nai.com/vil/dispVirus.asp?virus_k=
url             http://
```

+ Priority: Là thứ tự ưu tiên khi phân tích dữ liệu của các rules, nó có giá trị từ 1-255. Rules nào có độ ưu tiên cao hơn sẽ được kiểm tra trước. Priority cao nhất là 1.

+ 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.

- Payload Keyword:

+ content: dấu hiệu để xác định các mối hiểm họa an ninh mạng có thể xảy ra. Một rules có thể có nhiều content. 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 t thường, ký tự hoa, các ký tự đặc biệt, hay là các mã hexa.

- Ví dụ: content:“USER”; content:!“anonymous”;
- Một vài kí tự đặc biệt có thể được biểu diễn như sau:

+ nocase: Sử dụng trong content để giúp các luật không phân biệt chữ hoa và chữ thường.

- Ví dụ: content:"root";nocase; có nghĩa là phát hiện tất cả các từ khóa root mà không phân biệt chữ hoa và chữ thường.

+ 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.

+ offset: chỉ độ lệch byte trong tải trọng sẽ được kiểm tra.

- Ví dụ: offset là 3 thì sẽ kiểm tra từ byte thứ 4 trong tải trọng.

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

+ 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.

+ dsize: : được dùng để tìm một payload có độ dài bất kỳ.

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

- Định dạng của rpc như sau: rpc:<application number>, [<version number>|*], [<procedure number>|*]>;

+ replace: chỉ có thể sử dụng trong chế độ IPS. được dùng để thay đổi nội dung của payload, điều chỉnh lưu lượng mạng.

- Header Keywords:

+ IP Keyword:

- ttl: được dùng để kiểm tra thời gian sống time-to-live của gói tin IP.
  Lưu ý: Nếu ttl được thiết lập là 0, gói tin sẽ bị hủy ngay lập tức. Time-to-live dựa theo số hop.
- 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.

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

- 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.
- geoip: thiết lập xem gói tin xuất phát từ đâu, muốn đi đến đâu, xuất phát đến vùng nào, có vị trí địa lý ở đâu.

```
geoip: src, RU;           # source address khớp với vùng miền được khai báo
geoip: both, CN, RU;      # source và dst address khớp với vùng miền được khai báo
geoip: dst, CN, RU, IR;
geoip: both, US, CA, UK;
geoip: any, CN, IR;
```

+ Fragments:

- Fragbits: Được dùng để kiểm tra các phân mảnh của gói tin.

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.

  Các tuỳ chọn:

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

+ TCP Keywords:

- seq: được dùng để kiểm tra TCP sequence number. seq tăng lên 1 mỗi khi byte dữ liệu được truyền đi.
- ack: sử dụng để kiểm tra xem gói tin đã được nhận bởi nơi nhận hay chưa. 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 – cơ chế dùng để kiểm soát các dòng dữ liệu. Giá trị kích thước của cửa sổ có thể chạy từ 2 đến 65.535 byte.

- HTTP Keywords:

+ 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 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 yêu cầu từ máy trạm của máy chủ.

+ File_data: chỉ ra nội dung, đường dẫn tới file chứa dữ liệu được yêu cầu…

- Flow Keywords:

+ flowbits: gồm 2 phần, phần đầu tiên mô tả hành động mà nó sẽ thực hiện, phần thứ 2 chỉ ra tên của Flowbits.

Các hành động của flowbits:

- 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: Chỉ ra nơi mà gói tin xuất phát hoặc đích mà gói tin cần đến

Các hành động của flow

- to_client: Các gói tin từ server tới client.
- to_server: Các gói tin từ client tới server.
- from_client: Giống như to_server.
- from_server: Giống như to_client.
- established: Đang thiết lập kết nối.
- not_established: Ngược lại với established.
- stateless: áp dụng cho các gói tin độc lập mà không xem xét trạng thái kết nối.
- only_stream: áp dụng cho các gói tin đã được lắp ráp lại thành các luồng dữ liệu hoàn chỉnh.
- no_stream: Phù hợp với các gói tin chưa được lắp ráp lại bởi bộ xử lý luồng (stream engine).
- only_frag: Kết hợp gói tin từ các mảnh.
- no_frag: Kết hợp gói tin chưa được tập hợp từ các mảnh.

- Rule Thresholding:

+ threshold: đặt ngưỡng tối thiểu cho quy tắc trước khi quy tắc đó tạo cảnh báo.

Ví dụ: Rule này sẽ thiết lập cảnh báo khi máy chủ nhận được 10 emails từ cùng 1 máy trạm trong khoảng thời gian 1 phút.

```
alert tcp !$HOME_NET any -> $HOME_NET 25 (msg:"ET POLICY Inbound Frequent Emails - Possible Spambot Inbound"; \
flow:established; content:"mail from|3a|"; nocase;                                                            \
threshold: type threshold, track by_src, count 10, seconds 60;                                               \
reference:url,doc.emergingthreats.net/2002087; classtype:misc-activity; sid:2002087; rev:10;)
```

+ limit: Giới hạn số lần cảnh báo khi rule match

Ví dụ: Sẽ có 1 cảnh báo được tạo ra trong vòng 3 phút nếu nội dung MSIE 6.0 được phát hiện.

```
alert http $HOME_NET any -> any $HTTP_PORTS (msg:"ET USER_AGENTS Internet Explorer 6 in use - Significant Security Risk"; \
flow:to_server,established; content:"|0d 0a|User-Agent|3a| Mozilla/4.0 (compatible|3b| MSIE 6.0|3b|";                  \
threshold: type limit, track by_src, seconds 180, count 1;                                                           \
reference:url,doc.emergingthreats.net/2010706; classtype:policy-violation; sid:2010706; rev:7;)
```

+ both: kết hợp cả threshold và limit

Ví dụ: Cảnh báo chỉ được tạo ra nếu trong vòng 6 phút có từ 5 responses SIP

TCP 401 trái phép trở lên và nó chỉ cảnh báo 1 lần trong vòng 6 phút này.

```
alert tcp $HOME_NET 5060 -> $EXTERNAL_NET any (msg:"ET VOIP Multiple Unauthorized SIP Responses TCP"; \
flow:established,from_server; content:"SIP/2.0 401 Unauthorized"; depth:24;                          \
threshold: type both, track by_src, count 5, seconds 360;                                           \
reference:url,doc.emergingthreats.net/2003194; classtype:attempted-dos; sid:2003194; rev:6;)
```

+ dectection_filter: sử dụng để cảnh báo trên mỗi rule sau khi nó đạt đến giá trị ngưỡng. Nó khác với threshold và type threshold ở chỗ nó tạo ra cảnh báo cho mỗi rule khi nó đạt ngưỡng đã được set, trong khi đó lần thứ 2 sẽ đặt lại giá trị bộ đếm và cảnh báo lại khi ngưỡng lại đạt đến giá trị đã được set.

Ví dụ: Sau 15 lần match trong 2s thì sẽ tạo ra cảnh báo.

```
alert http $EXTERNAL_NET any -> $HOME_NET any \
     (msg:"ET WEB_SERVER WebResource.axd access without t (time) parameter - possible ASP padding-oracle exploit"; \
     flow:established,to_server; content:"GET"; http_method; content:"WebResource.axd"; http_uri; nocase;          \
     content:!"&t="; http_uri; nocase; content:!"&amp|3b|t="; http_uri; nocase;                                    \
     detection_filter:track by_src,count 15,seconds 2;                                                            \
     reference:url,netifera.com/research/; reference:url,www.microsoft.com/technet/security/advisory/2416728.mspx; \
     classtype:web-application-attack; sid:2011807; rev:5;)
```

2.3.6 Sự khác biệt với Snort của Suricata

a, Phát hiện giao thức tự động (Automatic Protocol Detection)

Như đã đề cập trong phần 2.3.3, bên cạnh các rules/signatures thì Suricata cung cấp khả năng Deep Packet Inspection - DPI để phân tích và quản lý lưu lượng mạng một cách sâu sắc hơn.Với DPI, Suricata không chỉ kiểm tra header mà còn thực hiện kiểm tra sâu vào payload của gói tin. Nhờ đó, Suricata có thể xác định và phân tích các giao thức ứng dụng thông qua việc kiểm tra payload, thay vì chỉ dựa vào các cổng (ports) cụ thể.

- Suricata tự động phát hiện giao thức của các giao thức lớp ứng dụng sau: dcerpc, dnp3, dns, http, imap (chỉ phát hiện theo mặc định; không phân tích cú pháp), ftp, modbus (bị vô hiệu hóa theo theo mặc định; trình phân tích cú pháp thăm dò tối giản; có thể dẫn đến kết quả false positive), smb, smb2 (bị vô hiệu hóa trong nội bộ của engine), smtp, ssh, tls (SSLv2, SSLv3, TLSv1, TLSv1.1 và TLSv1.2).
- Trong hầu hết các trường hợp, phát hiện giao thức trong Suricata không phụ thuộc vào cổng (port agnostic). Trong khi đó đối với Snort, để áp dụng các bộ tiền xử lý như http_inspect cho lưu lượng truy cập, nó phải qua một cổng được định cấu hình.

+ Theo mặc định, một số cấu hình cho lớp ứng dụng trong Suricata.yaml có thể/thực hiện chỉ định các cổng đích cụ thể (ví dụ: DNS).

+ Có thể thực hiện cấu hình rule với port "any" mà không phải lo lắng về tác động hiệu suất như với Snort.

- Ví dụ, nếu lưu lượng truy cập được Suricata phát hiện là HTTP thì thuộc tính http_* sẽ được điền và có thể được sử dụng, bất kể (các) cổng được chỉ định trong rules. Không nhất thiết phải kiểm tra đó có là giao thức HTTP để sử dụng các bộ đệm http_* (tức là rule được khai báo dạng: *alert http ....)*

Giả sử có rule Suricata như sau:

alert tcp any any -> any 80 (msg:"HTTP traffic to port 80"; http_uri; http_host; sid:1000; rev:1;)

→ Trong quy tắc này, tìm kiếm lưu lượng TCP đến cổng 80 và sử dụng các thuộc tính http_uri và http_host để kiểm tra nội dung của lưu lượng HTTP. Và có một số lưu lượng HTTP đến cổng 8080, thì dù cổng 8080 không được chỉ định trong quy tắc, nhưng Suricata vẫn sẽ phát hiện ra đây là lưu lượng HTTP và tự động điền vào các thuộc tính http_uri và http_host. Tức thuộc tính http_* có thể sử dụng để kiểm tra lưu lượng HTTP đến cổng 8080, ngay cả khi cổng 8080 không được liệt kê trong rule.

- Để phát hiện lưu lượng truy cập giao thức lớp ứng dụng được hỗ trợ và không muốn giới hạn cổng cụ thể, thì rule phải được viết là alert <protocol> với port any thay cho cổng giao thức thông thường.

Ví dụ, để phát hiện lưu lượng HTTP và không giới hạn port sử dụng thì:

+ Rule trong Snort:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS ...
```

+ Rule trong Suricata:

```
alert http $HOME_NET -> $EXTERNAL_NET any ...
```

Hoặc có thể dùng *app-layer-protocol:<protocol>;* trong rule, cụ thể:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (app-layer-protocol:http; ...
```

b, IP Reputation

IP Reputation là một công cụ quan trọng để tăng cường khả năng phát hiện và ứng phó với các mối đe dọa an ninh mạng dựa trên thông tin tình báo về địa chỉ IP. Mục đích của IP Reputation là xếp hạng địa chỉ IP trong công cụ Suricata. Nó sẽ thu thập, lưu trữ, cập nhật và phân phối thông tin về sự uy tín của một số lượng lớn địa chỉ IP, các thông tin có thể là tích cực hoặc tiêu cực được phân loại theo chuyên mục (categories). Kiến trúc "hub and spoke" sẽ cho phép cơ sở dữ liệu trung tâm (Hub) thu thập, lưu trữ và tổng hợp thông tin cập nhật về sự uy tín của IP, sau đó phân phối cho các cảm biến (Spoke) của người dùng để sử dụng trong hệ thống bảo mật.

Suricata IP Reputation cho phép sử dụng các danh sách IP, đặc biệt là khả năng gán các chuyên mục (categories) và điểm số uy tín (reputation score) cho các IP đó. Suricata sử dụng từ khóa "iprep" trong các quy tắc Suricata để kiểm tra: Hướng lưu lượng (direction), Chuyên mục (category), Giá trị (điểm số uy tín). Trong khi đó, bộ tiền xử lý “Reputation” của Snort được sử dụng để xác định các whitelist và blacklist của IP được sử dụng để tạo GID 136 alerts as well as block/drop/pass lưu lượng truy cập từ các IP được liệt kê tùy thuộc vào cách nó được định cấu hình.

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

3.1 Đặt vấn đề

Trong bối cảnh an ninh mạng ngày càng trở nên phức tạp và các cuộc tấn công mạng ngày càng tinh vi, việc xây dựng một hệ thống phát hiện và ngăn chặn xâm nhập mạnh mẽ là vô cùng cần thiết. Hệ thống IDPS mà nhóm xây dựng không chỉ cần phải có khả năng phát hiện và ngăn chặn các mối đe dọa mạng mà còn phải mô phỏng lại các điều kiện thực tế của hệ thống mạng. Điều này giúp đảm bảo rằng việc triển khai đồ án này có thể được áp dụng hiệu quả trong môi trường thực tế, nơi mà các cuộc tấn công mạng có thể diễn ra dưới nhiều hình thức khác nhau.

Để đáp ứng yêu cầu của một hệ thống IDPS hiệu quả, hệ thống cần phải có khả năng:

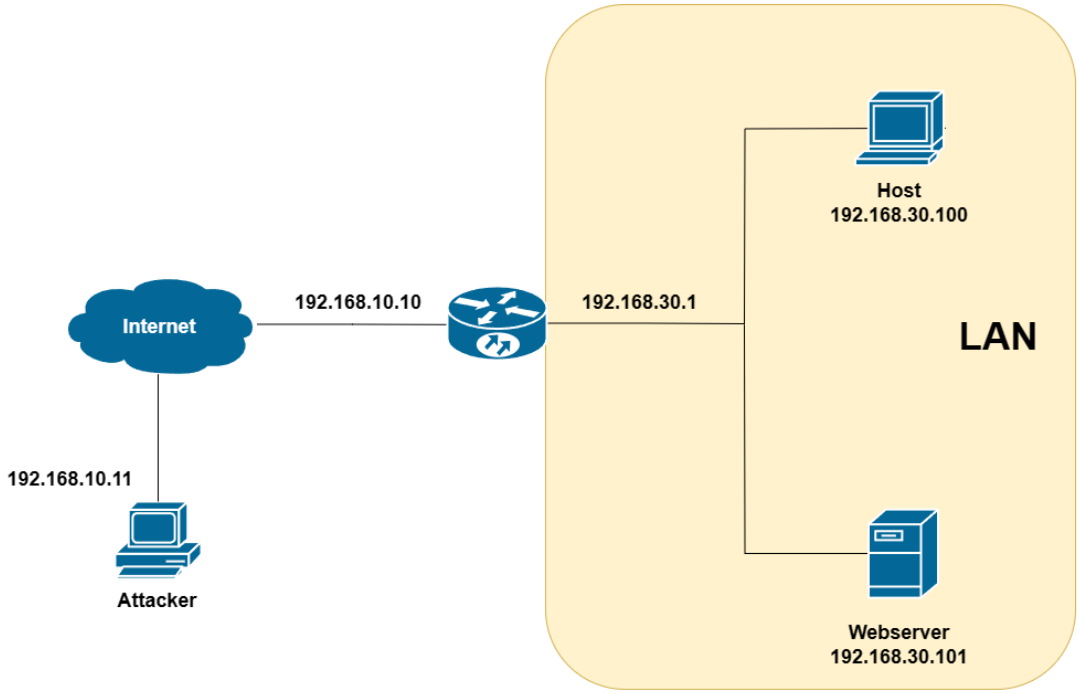
- *Mô phỏng mạng thực tế*: Hệ thống phải có khả năng tái tạo các điều kiện của một mạng thực tế, nơi người dùng Internet (có thể là kẻ tấn công) có thể tiến hành các cuộc tấn công. Điều này giúp kiểm tra hiệu quả của các chức năng bảo vệ và phản ứng của hệ thống trong các kịch bản tấn công khác nhau.
- *Phát hiện và ngăn chặn tấn công*: Hệ thống phải có khả năng phát hiện và ngăn chặn các cuộc tấn công mạng đơn giản.
- *Cấu hình đa dạng*: Hệ thống phải dễ dàng cấu hình để đáp ứng các kịch bản và yêu cầu bảo mật khác nhau. Việc này bao gồm khả năng thiết lập các luật phát hiện tấn công, quy tắc firewall, và các thông số hệ thống khác.
- *So sánh và đánh giá*: Hệ thống phải thể hiện được những chức năng nổi bật của pfSense và Suricata so với các hệ thống IDPS khác hiện có. Điều này bao gồm hiệu quả trong việc phát hiện tấn công, hiệu suất hệ thống, và khả năng tùy biến.

Mục tiêu chính của nghiên cứu này là xây dựng một hệ thống IDPS sử dụng pfSense và Suricata có khả năng mô phỏng lại các điều kiện thực tế của mạng, từ đó đánh giá và chứng minh hiệu quả của các công cụ này trong việc phát hiện và ngăn chặn các cuộc tấn công mạng. Hệ thống sẽ được cấu hình và kiểm thử với các kịch bản tấn công cụ thể để thể hiện các chức năng nổi bật của pfSense và Suricata, góp phần vào việc lựa chọn và triển khai các giải pháp bảo mật mạng hiệu quả trong thực tế.

Như vậy, chương này sẽ tập trung vào việc phân tích và thiết kế hệ thống IDPS dựa trên pfSense và Suricata, nhằm xây dựng một hệ thống bảo mật mạng mạnh mẽ và đáng tin cậy, đáp ứng được các yêu cầu bảo mật ngày càng cao trong môi trường mạng hiện đại.

3.2. Kiến trúc tổng quan

Sơ đồ kiến trúc hệ thống mà nhóm xây dựng được thể hiện như sau:



Hình 9: Kiến trúc tổng quan của hệ thống mạng

Hệ thống mà nhóm xây dựng gồm có:

- *Attacker*: Máy ảo Kali Linux, được xem như kẻ tấn công vào hệ thống mạng nội bộ. Attacker được dùng để thực hiện các loại tấn công khác nhau vào hệ thống mạng, nhằm đánh giá khả năng hoạt động của pfSense và Suricata. Attacker nằm bên ngoài hệ thống mạng nội bộ, kết nối vào hệ thống mạng qua Internet.
- *Router*: Máy ảo FreeBSD dùng để cài đặt pfSense và Suricata. Do pfSense có cung cấp chức năng định tuyến, nên ta có thể xem máy này như một router, quản lý các kết nối từ trong mạng nội bộ ra ngoài Internet và ngược lại.

- *Host*: Máy ảo Ubuntu, dùng để điều khiển, cấu hình và theo dõi các hoạt động của pfSense và Suricata. Máy được đặt trong môi trường mạng nội bộ (LAN), kết nối trực tiếp với *Router*.
- *Webserver*: Máy ảo Metasploitable 2, chứa các lỗ hổng để Attacker có thể khai thác từ bên ngoài. Máy được đặt trong môi trường mạng nội bộ (DMZ), kết nối trực tiếp với *Router*.

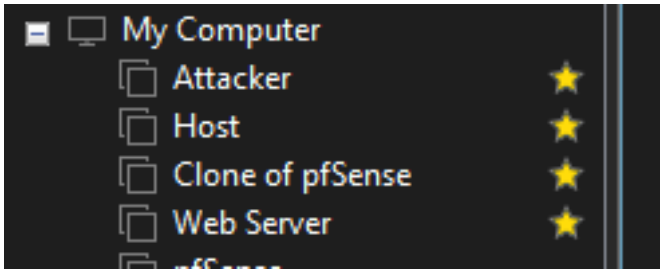
Bảng. Cấu hình IP cho các thiết bị trong hệ thống

Thiết bị	Interface	Địa chỉ IP	Netmask	Default Gateway
Attacker	N/A	192.168.10.11	255.255.255.0	192.168.10.2
Router	WAN	192.168.10.10	255.255.255.0	N/A
	LAN	192.168.30.1	255.255.255.0	N/A
Host	N/A	192.168.30.100	255.255.255.0	192.168.30.1
Webserver	N/A	192.168.30.101	255.255.255.0	192.168.30.1

CHƯƠNG 4: HIỆN THỰC HỆ THỐNG

Chương này sẽ trình bày chi tiết cách cài đặt hệ thống.

- Đầu tiên, cài đặt 4 máy ảo như đã trình bày ở chương 3



Hình 10: Hệ thống các máy ảo đã được cài đặt trên VMware

- Cấu hình IP cho các máy Attacker, Host và Webserver

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:fb:0d:e2
          inet addr:192.168.30.101  Bcast:192.168.30.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fefb:de2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:38 errors:0 dropped:0 overruns:0 frame:0
          TX packets:63 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4304 (4.2 KB)  TX bytes:6432 (6.2 KB)
          Interrupt:17 Base address:0x2000
```

Hình 11: IP máy Webserver

```
┌──(kali㉿kali)-[~]
└─$ ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.10.11  netmask 255.255.255.0  broadcast 192.168.10.255
        inet6 fe80::ad6e:6642:d5ee:aa80  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:5d:88:f7  txqueuelen 1000  (Ethernet)
        RX packets 4  bytes 804 (804.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 24  bytes 3126 (3.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Hình 12: IP máy Attacker

```
malware-testing-vm@ubuntu:~$ ifconfig ens33
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.30.100  netmask 255.255.255.0  broadcast 192.168.30.255
        inet6 fe80::d7b5:7209:537f:55c  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:7f:4d:0b  txqueuelen 1000  (Ethernet)
        RX packets 29  bytes 4388 (4.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 141  bytes 12699 (12.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Hình 13: Ip máy Host

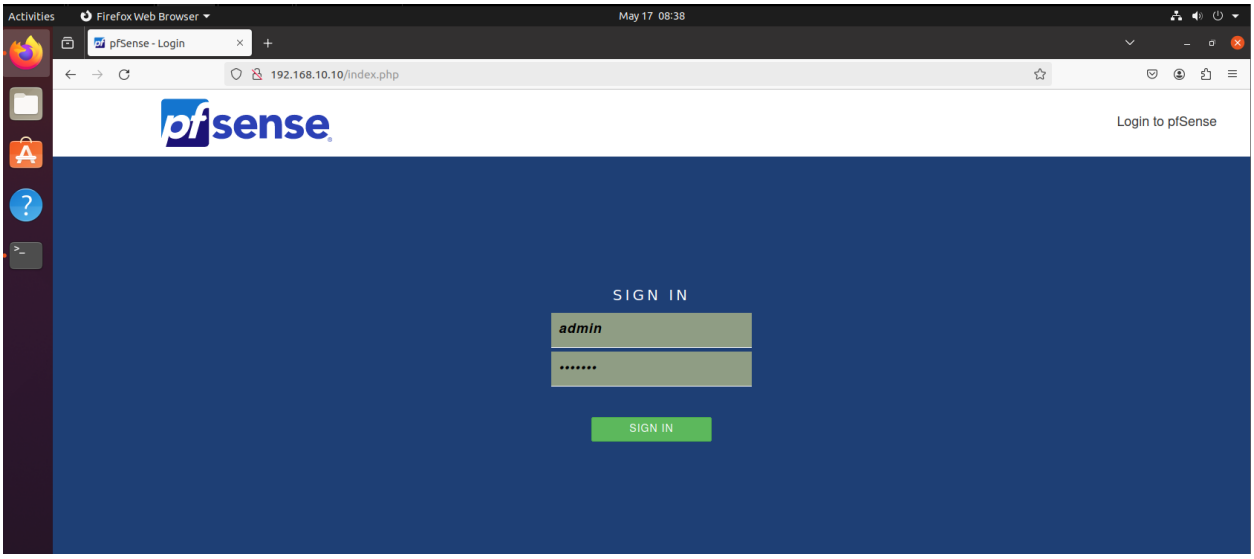
- Cài đặt pfSense trên máy Router: Tải file cài đặt (.iso) trên trang chủ của pfSense (https://www.pfsense.org/download/). Sau đó, cài đặt máy FreeBSD này trên Vmware.

- Sau khi cài đặt hoàn tất, thực hiện cấu hình IP cho các interface của máy

```
FreeBSD/amd64 (pfsense.home.arpa) (ttyv0)

VMware Virtual Machine - Netgate Device ID: e891b6e783f9f5264c71

*** Welcome to pfSense 2.7.2-RELEASE (amd64) on pfsense ***

 WAN (wan)       -> em0        -> v4/DHCP4: 192.168.10.10/24
 LAN (lan)       -> em1        -> v4: 192.168.30.1/24

 0) Logout (SSH only)                  9) pfTop
 1) Assign Interfaces                 10) Filter Logs
 2) Set interface(s) IP address       11) Restart webConfigurator
 3) Reset webConfigurator password    12) PHP shell + pfSense tools
```

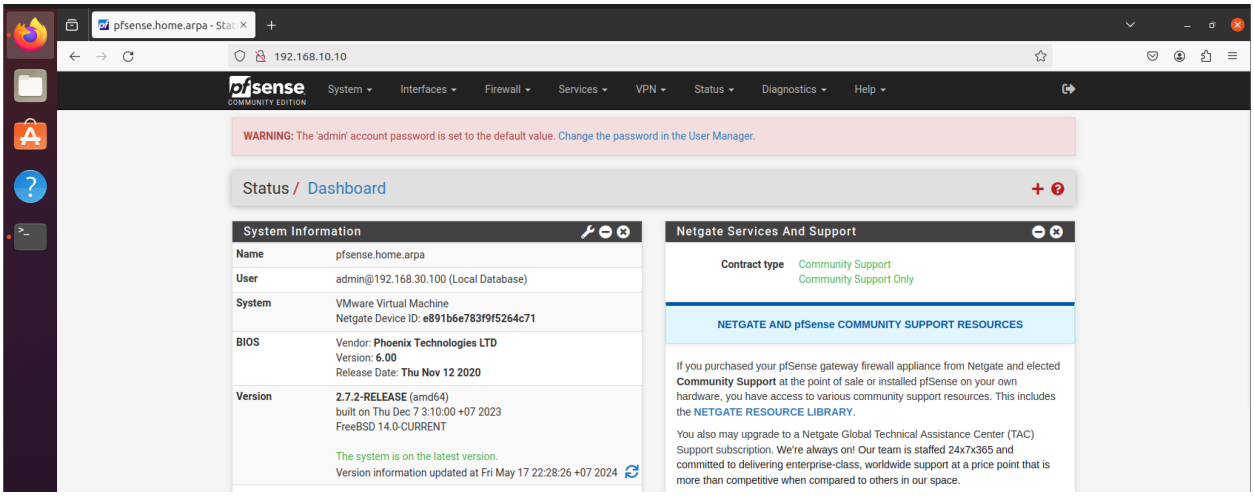
Hình 14: Cấu hình mạng của máy FreeBSD chứa pfSense

- Khi IP đã được cài đặt đúng cách, lúc này ta có thể truy cập vào giao diện điều khiển của pfSense từ máy host



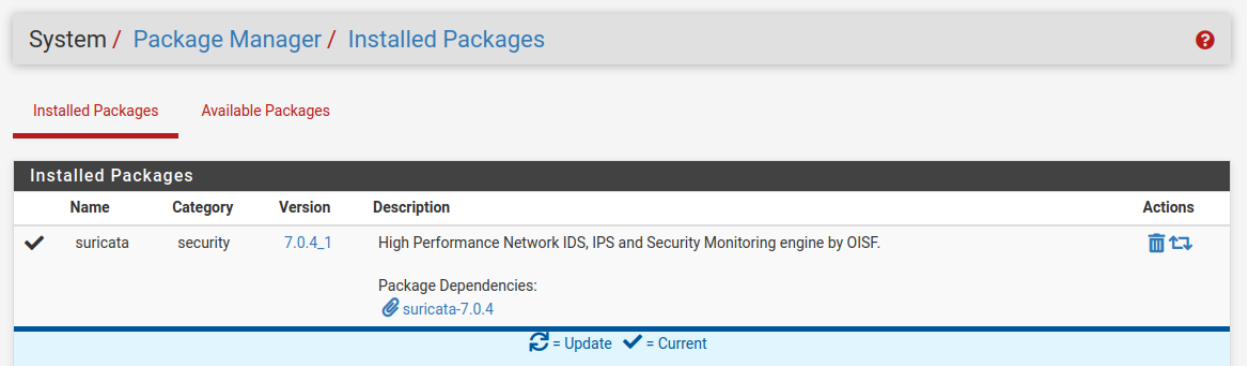
Hình 15: Giao diện ban đầu của pfSense

- Đăng nhập với tài khoản mặc định: **admin/pfsense**



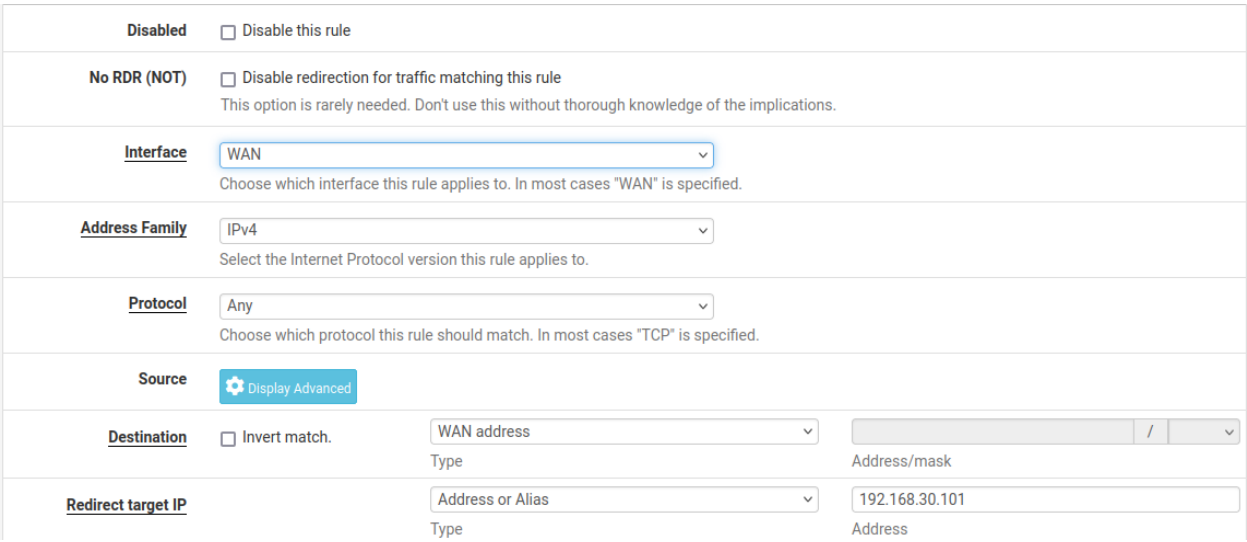
Hình 16: Giao diện sau khi đăng nhập

- Cài đặt Suricata trên pfSense: System -> Package Manager -> Available Packages -> Install

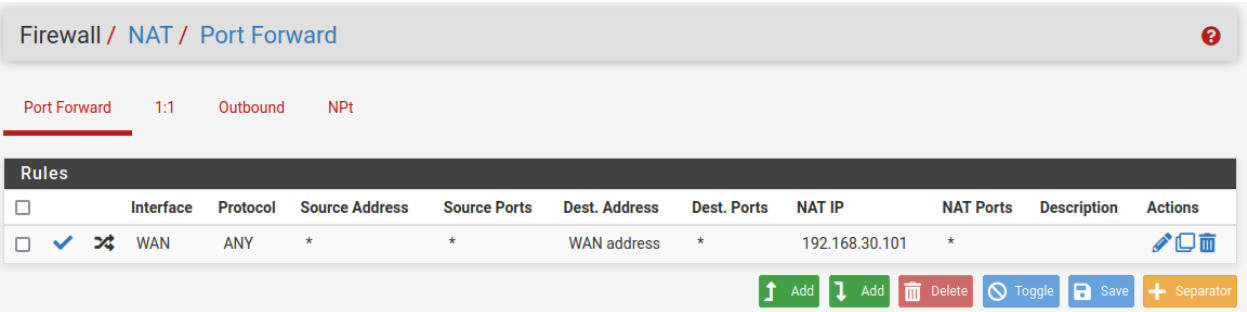


Hình 17: Giao diện cài đặt Suricata trên pfSense

- Cài đặt Port Forwarding cho pfSense để có thể cho phép máy ngoài kết nối đến Webserver: Firewall -> NAT -> Port Forward -> Add. Ở mục Interface, chọn **WAN**



Hình 18: Giao diện cài đặt Port Forwarding



Hình 19: Giao diện sau khi cài đặt Port Forwarding

CHƯƠNG 5: THỰC NGHIỆM VÀ ĐÁNH GIÁ

Chương này sẽ trình bày các kịch bản thực nghiệm và đánh giá kết quả thực nghiệm

5.1. Kịch bản thực nghiệm

Với hệ thống đã triển khai, nhóm đề xuất 5 kịch bản thực nghiệm nhằm đánh giá khả năng hoạt động của pfSense và Suricata như sau:

***5.1.1. Kịch bản 1*:** Chặn kỹ thuật tấn công bruteforce

Kịch bản này mô tả một cuộc tấn công brute force nhằm dò mật khẩu của một máy webserver nằm trong hệ thống mạng nội bộ. Cuộc tấn công sẽ được thực hiện từ máy attacker, với mục tiêu là kiểm thử khả năng phát hiện và ngăn chặn của hệ thống IDPS sử dụng pfSense và Suricata.

Mục tiêu của cuộc tấn công là dò được mật khẩu đăng nhập của dịch vụ web trên máy webserver bằng cách sử dụng kỹ thuật brute force. Hệ thống IDPS sẽ giám sát lưu lượng mạng để phát hiện và ngăn chặn cuộc tấn công này. Nếu Suricata phát hiện các dấu hiệu của brute force (nhiều yêu cầu đăng nhập thất bại trong một khoảng thời gian ngắn), nó sẽ kích hoạt cảnh báo và có thể ngăn chặn địa chỉ IP của máy attacker.

5.1.2. Kịch bản 2: Ngăn chặn tấn công SQL

Kịch bản này mô tả một cuộc tấn công SQL injection nhằm khai thác lỗ hổng bảo mật trên một máy chủ web chạy ứng dụng DVWA (Damn Vulnerable Web Application). Cuộc tấn công được thực hiện từ một máy Attacker nằm ngoài mạng con riêng biệt, với mục tiêu kiểm thử khả năng phát hiện và ngăn chặn của hệ thống IDPS sử dụng pfSense và Suricata.

Mục tiêu của cuộc tấn công là khai thác lỗ hổng SQL injection trên ứng dụng DVWA của máy webserver để lấy cắp thông tin cơ sở dữ liệu. Hệ thống IDPS sẽ giám sát lưu lượng mạng để phát hiện và ngăn chặn cuộc tấn công này. Nếu Suricata phát hiện các truy vấn bất thường hoặc các dấu hiệu của SQL injection, nó sẽ kích hoạt cảnh báo và có thể ngăn chặn địa chỉ IP của máy attacker.

5.1.3. Kịch bản 3: Khai thác lỗ hổng dịch vụ Samba trên máy victim để tạo TCP reverse shell

Kịch bản này mô tả một cuộc tấn công khai thác lỗ hổng dịch vụ Samba trên máy victim để tạo một kết nối TCP reverse shell, từ đó attacker có thể điều khiển máy victim từ xa. Cuộc tấn công được thực hiện từ một máy attacker nằm ngoài mạng con riêng biệt, với mục tiêu kiểm thử khả năng phát hiện và ngăn chặn của hệ thống IDPS sử dụng pfSense và Suricata.

Mục tiêu của cuộc tấn công là khai thác lỗ hổng dịch vụ Samba trên máy victim để tạo một TCP reverse shell, từ đó attacker có thể điều khiển máy victim từ xa. Hệ thống IDPS sẽ giám sát lưu lượng mạng để phát hiện và ngăn chặn cuộc tấn công này. Suricata phát hiện các yêu cầu khai thác lỗ hổng Samba và tạo kết nối reverse shell từ máy attacker và tạo cảnh báo. pfSense có thể tự động cập nhật quy tắc firewall để chặn IP của máy attacker nếu cuộc tấn công được xác định là nguy hiểm.

Attacker sử dụng công cụ Metasploit để khai thác lỗ hổng Samba. Metasploit sẽ thực hiện khai thác lỗ hổng Samba và tạo một kết nối reverse shell từ máy victim đến máy attacker.

5.1.4. Kịch bản 4: Chặn tấn công sử dụng VSFTPD để tạo backdoor

Kịch bản này mô tả một cuộc tấn công khai thác lỗ hổng trong phiên bản bị lỗi của dịch vụ VSFTPD (Very Secure FTP Daemon) để tạo một backdoor trên máy victim. Cuộc tấn công được thực hiện từ một máy attacker nằm ngoài mạng con riêng biệt, với mục tiêu kiểm thử khả năng phát hiện và ngăn chặn của hệ thống IDPS sử dụng pfSense và Suricata.

Mục tiêu của cuộc tấn công là khai thác lỗ hổng trong dịch vụ VSFTPD trên máy victim để tạo một backdoor, cho phép attacker truy cập vào hệ thống từ xa. Hệ thống IDPS sẽ giám sát lưu lượng mạng để phát hiện và ngăn chặn cuộc tấn công này. Suricata sẽ giám sát lưu lượng mạng và phát hiện các dấu hiệu của khai thác lỗ hổng VSFTPD và tạo backdoor. Nếu Suricata phát hiện các truy vấn FTP bất thường hoặc

các dấu hiệu của khai thác backdoor, nó sẽ kích hoạt cảnh báo và có thể ngăn chặn địa chỉ IP của máy attacker.

Attacker sử dụng công cụ Metasploit để thực hiện cuộc tấn công này. Metasploit sẽ thực hiện khai thác lỗ hổng và tạo một kết nối backdoor từ máy victim đến máy attacker.

5.1.5. Kịch bản 5: Cảnh báo tấn công Command Execution

Kịch bản này mô tả một cuộc tấn công Command Execution, trong đó attacker khai thác một lỗ hổng bảo mật trên máy victim để thực thi lệnh hệ thống từ xa. Cuộc tấn công được thực hiện từ một máy attacker nằm ngoài mạng con riêng biệt. Hệ thống IDPS sử dụng pfSense và Suricata sẽ được cấu hình để phát hiện và cảnh báo về cuộc tấn công này.

Mục tiêu của cuộc tấn công là khai thác lỗ hổng trong một ứng dụng web trên máy victim để thực thi lệnh hệ thống từ xa. Hệ thống IDPS sẽ giám sát lưu lượng mạng để phát hiện và cảnh báo về cuộc tấn công này. Suricata sẽ giám sát lưu lượng mạng và phát hiện các dấu hiệu của command execution từ máy attacker. Nếu Suricata phát hiện các truy vấn HTTP bất thường hoặc các dấu hiệu của command execution, nó sẽ kích hoạt cảnh báo và có thể ngăn chặn địa chỉ IP của máy attacker.

5.2. Chi tiết thực hiện

Quá trình thực hiện cụ thể các kịch bản tấn công trên được nhóm quay lại và đăng tải lên youtube. Link youtube: https://www.youtube.com/playlist?list=PLQoEfnnznXmRAQr70YhVNnZuobUD3LFcO

5.3. Kết quả

Sau khi thực hiện các kịch bản thực nghiệm, nhóm đã thu thập được các kết quả cho thấy khả năng phát hiện và ngăn chặn tấn công của hệ thống IDPS sử dụng pfSense và Suricata. Các kết quả này được phân tích chi tiết dưới đây cho từng kịch bản.

5.3.1. Kịch bản 1

Trong kịch bản này, cuộc tấn công brute force nhằm vào máy webserver đã được thực hiện từ một máy attacker. Hệ thống IDPS được cấu hình để giám sát lưu lượng mạng và phát hiện các dấu hiệu của brute force, như nhiều yêu cầu đăng nhập thất bại trong một khoảng thời gian ngắn.

Kết quả:

- *Phát hiện:* Suricata đã thành công trong việc phát hiện các dấu hiệu của tấn công brute force. Cụ thể, Suricata nhận diện được nhiều yêu cầu đăng nhập thất bại liên tiếp từ địa chỉ IP của máy attacker.
- *Cảnh báo:* Sau khi phát hiện, Suricata đã tạo ra các cảnh báo tức thì, ghi lại thông tin chi tiết về các yêu cầu đăng nhập thất bại và nguồn gốc của cuộc tấn công.
- *Ngăn chặn*: Dựa trên các cảnh báo từ Suricata, pfSense đã tự động cập nhật các quy tắc firewall để chặn địa chỉ IP của máy attacker, ngăn chặn cuộc tấn công tiếp tục.

5.3.2. Kịch bản 2

Kịch bản này mô tả một cuộc tấn công SQL injection nhằm vào ứng dụng DVWA trên máy webserver. Cuộc tấn công được thực hiện từ máy attacker nằm ngoài mạng con riêng biệt.

Kết quả:

- *Phát hiện:* Suricata đã phát hiện thành công các truy vấn SQL bất thường, bao gồm các lệnh SQL đặc biệt nhằm khai thác lỗ hổng SQL injection trên DVWA.
- *Cảnh báo:* Suricata ngay lập tức tạo ra các cảnh báo chi tiết về các truy vấn SQL bất thường, bao gồm các thông tin về nội dung truy vấn và địa chỉ IP của attacker.
- *Ngăn chặn*: pfSense đã cập nhật các quy tắc firewall dựa trên các cảnh báo từ Suricata để chặn địa chỉ IP của attacker, ngăn chặn việc tiếp tục thực hiện các cuộc tấn công SQL injection.

5.3.3. Kịch bản 3

Kịch bản này mô tả cuộc tấn công khai thác lỗ hổng dịch vụ Samba trên máy victim để tạo một kết nối TCP reverse shell từ máy victim đến máy attacker.

Kết quả:

- *Phát hiện:* Suricata đã phát hiện các yêu cầu bất thường nhắm vào dịch vụ Samba, bao gồm các lệnh khai thác lỗ hổng và thiết lập kết nối reverse shell.
- *Cảnh báo:* Ngay khi phát hiện, Suricata đã tạo ra các cảnh báo về các hoạt động khai thác lỗ hổng Samba và thiết lập kết nối reverse shell.
- *Ngăn chặn*: Dựa trên các cảnh báo, pfSense đã cập nhật các quy tắc firewall để chặn kết nối từ máy attacker, ngăn chặn việc điều khiển máy victim từ xa.

5.3.4. Kịch bản 4

Trong kịch bản này, một cuộc tấn công nhằm khai thác lỗ hổng trong phiên bản bị lỗi của dịch vụ VSFTPD để tạo một backdoor trên máy victim đã được thực hiện.

Kết quả:

- *Phát hiện:* Suricata đã phát hiện các dấu hiệu của cuộc tấn công nhằm khai thác lỗ hổng VSFTPD, bao gồm các truy vấn FTP bất thường và các hoạt động tạo backdoor.
- *Cảnh báo:* Các cảnh báo chi tiết về các hoạt động khai thác lỗ hổng VSFTPD và tạo backdoor đã được Suricata tạo ra ngay lập tức.
- *Ngăn chặn*: pfSense đã cập nhật các quy tắc firewall để chặn địa chỉ IP của máy attacker, ngăn chặn việc tiếp tục tấn công và truy cập vào hệ thống từ xa.

5.3.5. Kịch bản 5

Kịch bản này mô tả một cuộc tấn công Command Execution, trong đó attacker khai thác lỗ hổng bảo mật trên máy victim để thực thi lệnh hệ thống từ xa.

Kết quả:

- *Phát hiện:* Suricata đã phát hiện các truy vấn HTTP bất thường chứa các lệnh hệ thống, dấu hiệu của cuộc tấn công Command Execution.
- *Cảnh báo:* Ngay khi phát hiện, Suricata đã tạo ra các cảnh báo chi tiết về các truy vấn HTTP bất thường và các lệnh hệ thống được thực thi từ xa.
- *Ngăn chặn*: pfSense đã cập nhật các quy tắc firewall để chặn địa chỉ IP của máy attacker, ngăn chặn việc thực thi lệnh hệ thống từ xa.

5.4. Đánh giá

5.4.1. Về các hoạt động triển khai đã thực hiện

Kết quả thực nghiệm cho thấy hệ thống IDPS sử dụng pfSense và Suricata đã hoạt động hiệu quả trong việc phát hiện và ngăn chặn các cuộc tấn công từ bên ngoài. Suricata với khả năng giám sát và phân tích lưu lượng mạng đã thành công trong việc phát hiện các dấu hiệu tấn công, trong khi pfSense đã nhanh chóng cập nhật các quy tắc firewall để chặn các cuộc tấn công, bảo vệ mạng nội bộ.

Các kịch bản thực nghiệm đã minh chứng cho khả năng của hệ thống trong việc đối phó với nhiều loại tấn công khác nhau, từ brute force, SQL injection, khai thác lỗ hổng dịch vụ Samba, sử dụng VSFTPD để tạo backdoor, cho đến command execution. Qua đó, khẳng định tính hiệu quả và ưu việt của hệ thống IDPS sử dụng pfSense và Suricata trong việc bảo vệ hệ thống mạng khỏi các mối đe dọa tiềm ẩn.

5.4.2. Hạn chế

Mặc dù hệ thống IDPS sử dụng pfSense và Suricata đã cho thấy hiệu quả cao trong việc phát hiện và ngăn chặn các cuộc tấn công, vẫn tồn tại một số hạn chế cần được xem xét để nâng cao khả năng bảo mật và tối ưu hóa hiệu suất hệ thống.

Dưới đây là một số hạn chế mà nhóm rút ra được trong quá trình thực hiện đồ án:

- Suricata yêu cầu tài nguyên phần cứng mạnh để phân tích lưu lượng mạng trong thời gian thực. Điều này có thể trở thành một vấn đề trên các hệ thống có cấu hình phần cứng hạn chế hoặc khi lưu lượng mạng rất lớn. Hơn nữa, việc giám sát và phân tích lưu lượng mạng trong thời gian thực có thể gây ra độ trễ, ảnh hưởng đến hiệu suất mạng và trải nghiệm người dùng.
- Hệ thống mạng mà nhóm xây dựng còn tương đối đơn giản, để áp dụng được hoàn toàn vào thực tế cần cấu hình phức tạp hơn so với quy mô của đồ án này. Việc đó đòi hỏi nhiều kiến thức về cấu trúc hạ tầng mạng, kiến thức chuyên sâu về pfSense và Suricata. Những công việc, kỹ thuật trên rất phức tạp, dễ để lại lỗ hổng nếu không được cấu hình và triển khai đúng cách.
- Hệ thống có thể gặp phải tình trạng báo động sai (false positives) hoặc bỏ sót các cuộc tấn công thực sự (false negatives). False positives có thể làm tăng khối lượng công việc cho quản trị viên mạng, trong khi false negatives có thể dẫn đến việc bỏ sót

các cuộc tấn công nguy hiểm. Ngoài ra, Suricata dựa vào các quy tắc (rules) để phát hiện các cuộc tấn công. Nếu không cập nhật các quy tắc này thường xuyên, hệ thống có thể không phát hiện được các loại tấn công mới hoặc các biến thể của các tấn công cũ.

- Hệ thống IDPS có thể gặp khó khăn trong việc tích hợp với các thiết bị mạng và bảo mật khác nếu không có sự tương thích tốt. Việc này có thể làm giảm hiệu quả bảo mật tổng thể của hệ thống. Đồng thời, việc tích hợp pfSense và Suricata với các giải pháp bảo mật khác (như SIEM - Security Information and Event Management) có thể gặp khó khăn, đòi hỏi thêm các cấu hình phức tạp.
- Các cuộc tấn công phức tạp hơn, chẳng hạn như tấn công APT (Advanced Persistent Threat) hoặc các cuộc tấn công zero-day, có thể vượt qua khả năng phát hiện của hệ thống nếu không được cấu hình và quản lý tốt. Hơn nữa, các cuộc tấn công sử dụng lưu lượng mã hóa hoặc thông qua các kết nối VPN có thể làm giảm khả năng phát hiện của Suricata, vì lưu lượng này khó giám sát và phân tích hơn.
- Khi lưu lượng mạng tăng lên hoặc khi mạng mở rộng, hệ thống IDPS cần phải được nâng cấp về phần cứng và cấu hình để duy trì hiệu suất và khả năng phát hiện tấn công. Việc mở rộng hệ thống có thể phức tạp và tốn kém.

CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1. Kết luận

Nhóm đã tiến hành nghiên cứu, tìm hiểu sâu về hệ thống, chức năng, cách cài đặt, cấu hình và sử dụng hệ thống IDPS được trang bị pfSense và Suricata. Hệ thống IDPS sử dụng pfSense và Suricata mà nhóm nghiên cứu và triển khai đã chứng minh được khả năng hiệu quả trong việc phát hiện và ngăn chặn các cuộc tấn công mạng phổ biến.

Các thử nghiệm thực tế với các kịch bản tấn công khác nhau đã cho thấy hệ thống không chỉ có khả năng phát hiện sớm mà còn ngăn chặn kịp thời, giúp giảm thiểu rủi ro và thiệt hại tiềm tàng. Tuy nhiên, hệ thống cũng bộc lộ một số hạn chế như khó khăn trong việc xử lý các cuộc tấn công phức tạp và yêu cầu cấu hình phần cứng, phần mềm phức tạp để duy trì hiệu suất hoạt động trong môi trường mạng lớn.

6.2. Hướng phát triển

Để nâng cao hiệu quả của hệ thống và đáp ứng tốt hơn các yêu cầu bảo mật ngày càng cao, nhóm đề xuất một số hướng phát triển trong tương lai như sau:

- Tích hợp thêm các công cụ khác vào hệ thống: pfSense hỗ trợ cài đặt thêm nhiều package, do đó ta có thể tìm hiểu và cài đặt thêm nhiều package khác nhau nhằm tăng cường khả năng theo dõi và bảo vệ hệ thống mạng.
- Sử dụng thêm các công cụ, hệ thống liên quan đến Trí tuệ nhân tạo để phân tích lưu lượng mạng, phân tích log, làm tăng cường khả năng quản lý, nắm bắt và ngăn chặn kịp thời các nguy cơ mới có thể xuất hiện, từ đó tăng khả năng bảo mật của hệ thống IDPS.

TÀI LIỆU THAM KHẢO

STT	Tài liệu
1	Slide Môn học Hệ thống tìm kiếm, phát hiện và ngăn ngừa xâm nhập, Trường Đại học Công nghệ Thông tin, ĐHQG Thành phố Hồ Chí Minh
2	Suricata User Guide. https://docs.suricata.io/en/latest/index.html
3	pfSense Documentation. https://docs.netgate.com/pfsense/en/latest
4	Trung tâm tin học, Bộ Kế hoạch và Đầu tư. Báo cáo tóm tắt kết quả thực hiện đề tài khoa học cấp bộ, Nghiên cứu ứng dụng hệ thống giám sát mã nguồn mở Snorby để giám sát hệ thống mạng của bộ. https://fileportalcms.mpi.gov.vn/TinBai/DinhKem/39375/1. Bao_cao_tom_tat_Ketquathuchien.pdf
5	Dallon Robinette. How Does Suricata Work? (2023). https://www.stamus-networks.com/blog/how-does-suricata-work
6	Trần Quang Huy, Học viện Kỹ thuật Mật mã. Đề tài thực tập cơ sở Tìm hiểu hệ thống phát hiện xâm nhập mạng Suricata (2020). https://pdfcoffee.com/suricata-tqhdoc-5-pdf-free.html
7	Trương Hoài Nhân. Báo cáo Lab Quản trị mạng: Nghiên cứu và báo cáo PfSense. https://tailieu.vn/doc/bao-cao-lab-quan-tri-mang-nghien-cuu-va-bao-cao-pfsense-1769628.html#google_vignette
8	huongbn. Ghi-chep-Suricata- (2018). https://github.com/huongbn/Ghi-chep-Suricata-
9	Đồ án tổng hợp Hệ thống phát hiện xâm nhập Suricata trên Firewall pfSense (2021). https://123docz.net/document/10389042-dath-he-thong-phat-hien-xam-nhap-suricata.htm
10	PFsense là gì? Các Tính Năng và Lợi Ích của PFsense (2024). https://lanit.com.vn/pfsense-la-gi-tinh-nang-loi-ich.html
11	Nguyen Manh Dung. PFsense là gì – Tổng Quan về PFsense. https://mdungblog.wordpress.com/pfsense-la-gi-tong-quan-ve-pfsense/