HỌC VIỆN KỸ THUẬT MẬT MÃ

KHOA AN TOÀN THÔNG TIN

BÁO CÁO CHUYÊN ĐỀ CƠ SỞ

NHÓM 83 – ĐỀ TÀI:

XÂY DỰNG VÀ TRIỂN KHAI HỆ THỐNG PHÒNG CHỐNG TẤN CÔNG IDS/IPS DỰA TRÊN CÔNG CỤ MÃ NGUỒN MỞ

Ngành: An toàn thông tin

Mã số: 7.48.02.02

Sinh viên thực hiện: **Phan Anh Duy** **AT170615**

Phạm Ngọc Thái **AT170645**

Trần Đức Thắng **AT170646**

Người hướng dẫn: **Phạm Văn Hưởng**

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

Hà Nội, 2023

LỜI CẢM ƠN

Trong suốt thời gian học tập tại chúng em đã nhận được rất nhiều sự quan tâm và tận tình chỉ bảo của các thầy cô. Nhờ đó mà chúng em đã tiếp thu được không ít các kiến thức cùng kinh nghiệm quý báu trong lĩnh vực An toàn thông tin. Các kiến thức và kinh nghiệm này là một hành trang vững chắc cho chúng em trong học tập cũng như trong công việc sau này.

Chúng em xin chân thành cảm ơn thầy Phạm Văn Hưởng đã tận tình hướng dẫn, truyền đạt kiến thức và chỉ bảo cho chúng em trong suốt thời gian thực hiện đề tài để chúng em có thể hoàn thành bài báo cáo này một cách tốt nhất. Mặc dù có nhiều cố gắng nhưng với lượng kiến thức hạn hẹp nên bài báo cáo của chúng em không thể tránh khỏi nhiều thiếu sót. Chúng em rất mong nhận được sự góp ý, chỉ bảo của thầy để bài báo cáo của chúng em được hoàn thiện hơn. Chúng em xin chân thành cảm ơn!

Hà Nội, tháng 5 năm 2023

MỤC LỤC

MỞ ĐẦU

Trong bối cảnh hiện nay, khi Internet phủ khắp toàn cầu và trực tiếp tác động đến sự phát triển toàn diện của hầu hết các lĩnh vực trong đời sống như: Kinh tế, chính trị, văn hóa, xã hội, an ninh quốc phòng... thì an ninh mạng và bảo mật dữ liệu trở thành một vấn đề mang tính chất "sống còn" của các quốc gia, tổ chức và doanh nghiệp. Bên cạnh đó, các cuộc tấn công mạng ngày càng gia tăng, với mức độ tinh vi cũng như sức tàn phá mạnh hơn bao giờ hết. Chính vì thế đòi hỏi các quốc gia, tổ chức, doanh nghiệp trên toàn thế giới cần phải đầu tư vào các chính sách an ninh và bảo mật mạng nhiều hơn nữa. Ngày nay, an ninh thông tin đã trở thành một vấn đề quan trọng và cấp thiết trong môi trường kỹ thuật số. Các tổ chức và doanh nghiệp ngày càng phải đối mặt với nguy cơ tấn công mạng từ các hacker và kẻ xâm nhập.

Trong một thế giới mạng liên kết, các cuộc tấn công mạng ngày càng phức tạp và đa dạng. Các hacker và kẻ xâm nhập không ngừng nỗ lực tìm kiếm lỗ hổng trong hệ thống và khai thác chúng để đánh cắp thông tin quan trọng hoặc gây hại cho tổ chức. Các hệ thống mạng truyền thống không đủ mạnh mẽ để ngăn chặn những cuộc tấn công này, do đó, cần có hệ thống phòng chống tấn công IDS/IPS hiệu quả để bảo vệ mạng và dữ liệu.. Việc sử dụng mã nguồn mở trong xây dựng hệ thống IDS/IPS mang lại tính minh bạch và sự tin cậy. Nhờ tính chất công khai, ai cũng có thể kiểm tra mã nguồn, tìm lỗi và đóng góp vào việc phát triển và cải thiện công cụ. Điều này góp phần vào việc tạo ra một môi trường mạng an toàn hơn, mà cộng đồng có thể cùng nhau đóng góp và hưởng lợi từ đó.

Triển khai một hệ thống mạng và xây dựng được cơ chế bảo mật chặt chẽ, an toàn là biện pháp duy nhất duy trì tính bền vững cho hệ thống của quốc gia, tổ chức, doanh nghiệp đó.

Như đã đề cập đến, các cuộc tấn công mạng ngày càng tăng, tinh vi hơn, tàn phá nghiêm trọng hơn và gây tổn thất nặng nề hơn. Vì thế, cần phải am hiểu các cách thức tấn công và phát hiện một cách kịp thời để có thể tìm ra các biện pháp phòng chống cũng như ngăn chặn, để làm giảm các tổn thất gây ra ở mức thấp nhất. Đó là lý do để nhóm chúng em thực hiện bài luận văn về "Xây dựng và triển khai hệ thống phòng chống tấn công IDS/IPS dựa trên công cụ mã nguồn mở" này.

DANH MỤC CÁC CỤM TỪ VIẾT TẮT

API	Application Programming Interface
ARP	Address Resolution Protocol
ASCII	American Standard Code for Information Interchange
BASE	Basic Analysis and Security Engine
BSD	Berkeley Software Distribution
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DIDS	Distributed IDS
DMZ	Demilitarized Zone
DNS	Domain Name System
DoS	Denial of Service
FDDI	Fiber Distributed Data Interface
FTP	File Transfer Protocol
HIDS	Host-Based IDS
HIPS	Host-Based IPS
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System

MTU	Maximum Transmission Unit
NAT	Network Address Translation
NIDS	Network-Based IDS
NIPS	Network-Based IPS
RFC	Request for Comments
RPC	Remote Procedure Call
SLIP	Serial Line Internet Protocol
SMB	Server Message Block
SMS	Short Message Service
SNMP	Simple Network Management Protocol
TCP	Transport Control Protocol
TTL	Time to Live
UDP	User Datagram Protocol
URL	Uniform Resource Locator
WIDS	Wireless IDS
XML	Extensible Markup Language

DANH MỤC CÁC BẢNG

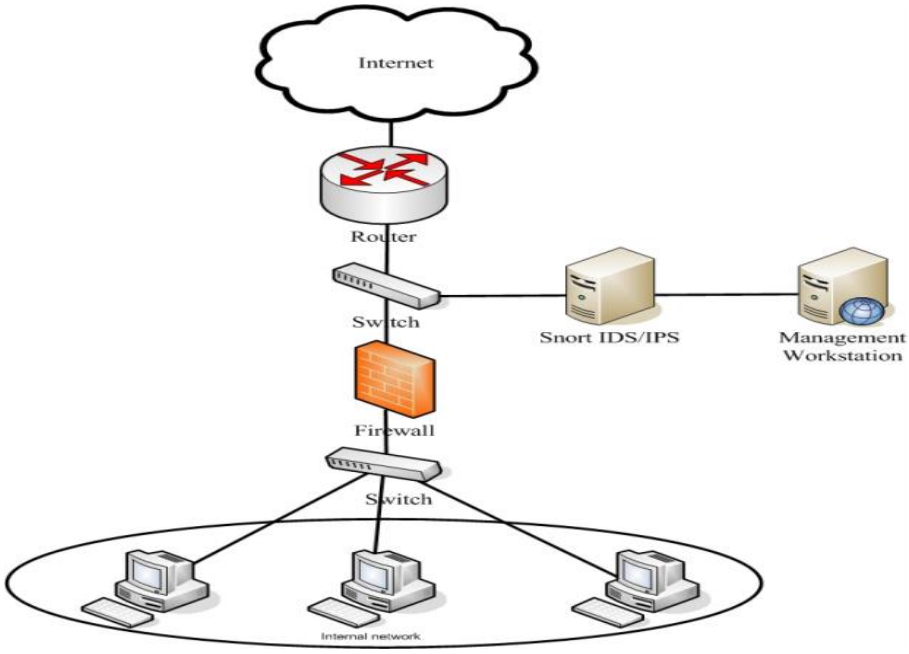
DANH MỤC HÌNH VẼ

CHƯƠNG I. TỔNG QUAN VỀ HỆ THỐNG IDS/IPS

1 Khái niệm và vai trò của hệ thống IDS

1.1 Giới thiệu về IDS.

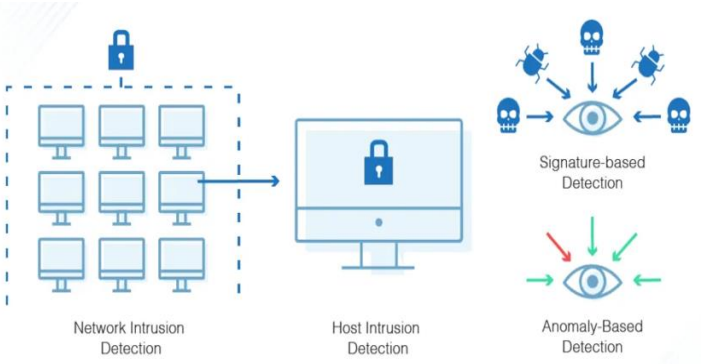
IDS (Intrusion Detection System: Hệ thống phát hiện xâm nhập) là hệ thống an ninh (có thể là phần mềm, phần cứng hoặc kết hợp cả hai). Nó giám sát, phân tích lưu thông mạng và dữ liệu để nhận biết, cung cấp thông tin và đưa ra cảnh báo khi có những hành động khả nghi, xâm nhập trái phép cũng như khai thác tài nguyên bất hợp pháp trên hệ thống mạng cho nhà quản trị bằng việc hiển thị cảnh báo, ghi logfile... và phản ứng lại bằng các hành động đã được thiết lập trước .Ngoài ra, IDS còn có thể xác định được nguồn gốc của các nguy cơ trên là từ bên ngoài hay là từ bên trong hệ thống.



Hình 1.1. Hệ thống IDS

Trong một số trường hợp, do không hiểu rõ về bản chất cũng như cách thức hoạt động của IDS nên nó thường bị nhầm lẫn với hệ thống kiểm tra lưu lượng mạng (được sử dụng để phát hiện các cuộc tấn công DoS), các bộ quét bảo mật (nhằm tìm ra các lỗ hổng trong mạng), các phần mềm chống virus (phát hiện các mã độc), tường lửa...

Hoạt động của IDS nhìn chung rất đa dạng, nhưng mục đích cuối cùng của nó là phát hiện, ngăn chặn kịp thời các hoạt động của kẻ tấn công trước khi chúng gây tổn hại đến hệ thống.



Hình 1.2. Cách IDS hoạt động

1.2 Chức năng của IDS

IDS có 3 chức năng quan trọng nhất là: giám sát, cảnh báo và bảo vệ.

+ Giám sát: IDS giám sát lưu lượng mạng, hoạt động của hệ thống và hành vi của người dùng. Nhờ vào đó mà nó có thể phát hiện ra hành động khả nghi.

+ Cảnh báo: Khi phát hiện ra những dấu hiệu bất thường, IDS sẽ đưa ra các cảnh báo cho hệ thống hay người quản trị

+ Bảo vệ: IDS dùng những thiết lập mặc định và những cấu hình từ nhà quản trị mà có những hành động thiết thực chống lại kẻ xâm nhập và phá hoại.

Ngoài ra IDS còn có những chức năng mở rộng như:

+ Phân biệt cuộc tấn công vào hệ thống là từ bên trong hay bên ngoài.

+ Phát hiện dựa vào sự so sánh lưu lượng mạng hiện tại với các thông số chuẩn của hệ thống và những dấu hiệu đã biết, IDS có thể phát hiện ra những dấu hiệu bất thường và đưa ra các cảnh báo và bảo vệ ban đầu cho hệ thống.

1.3 Các thành phần cơ bản của IDS

IDS có các thành phần cơ bản sau:

+ Sensor/ Agent: là các bộ cảm biến được đặt trong hệ thống nhằm phát hiện những xâm nhập hoặc dấu hiệu bất thường trên toàn mạng. Nhiệm vụ chính của nó là giám sát và phân tích các hoạt động. "Sensor" thường được dùng cho dạng Network-Based IDS (NIDS) trong khi "Agent" thường được dùng cho dạng Host-Based IDS (HIDS).

+ Management Server: là một thiết bị trung tâm dùng thu nhận các thông tin từ Sensor/ Agent và quản lý chúng.

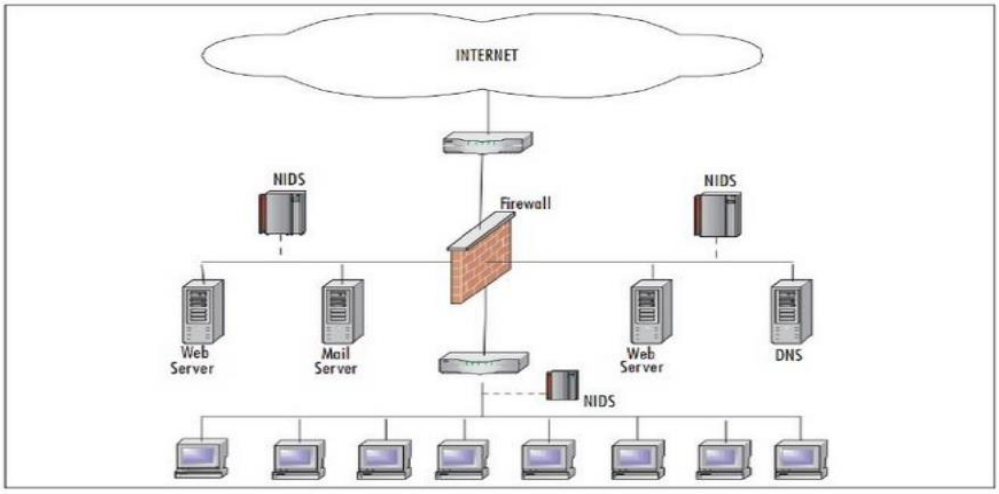
+ Database Server: dùng lưu trữ thông tin từ Sensor/ Agent hay Management Server.

+ Console: là chương trình cung cấp giao diện cho IDS users/ Admins. Có thể cài đặt trên một máy tính bình thường dùng để phục vụ cho tác vụ quản trị, hoặc để giám sát, phân tích.

1.4 Phân loại IDS

Có nhiều cách phân loại hệ thống IDS, dựa vào phạm vi giám sát có 3 loại chính:

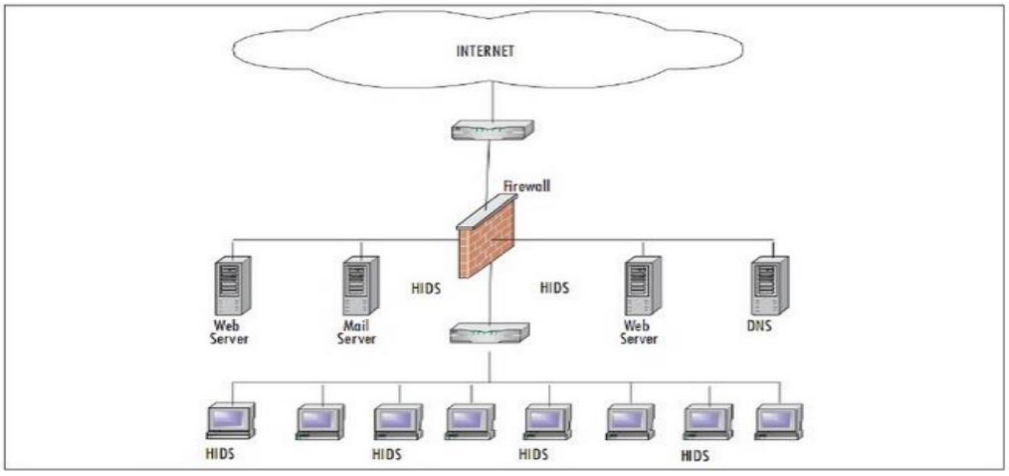
a) Network-Based IDS (NIDS)



Hình 1.3 Network-Based IDS

NIDS là một loại IDS phổ biến, nó giám sát lưu lượng mạng ở tất cả các tầng của mô hình OSI. NIDS sử dụng các Sensor đặt ở các phân đoạn mạng cần quản lý, giám sát mọi luồng thông tin, dữ liệu ra vào trên phân đoạn mạng đó. Khi có dấu hiệu nghi ngờ, Sensor sẽ phát ra cảnh báo gửi về trạm quản lý, trạm quản lý sẽ phân tích gói tin và thực hiện theo các kịch bản đã được thiết lập sẵn.

b) Host-Based IDS (HIDS)

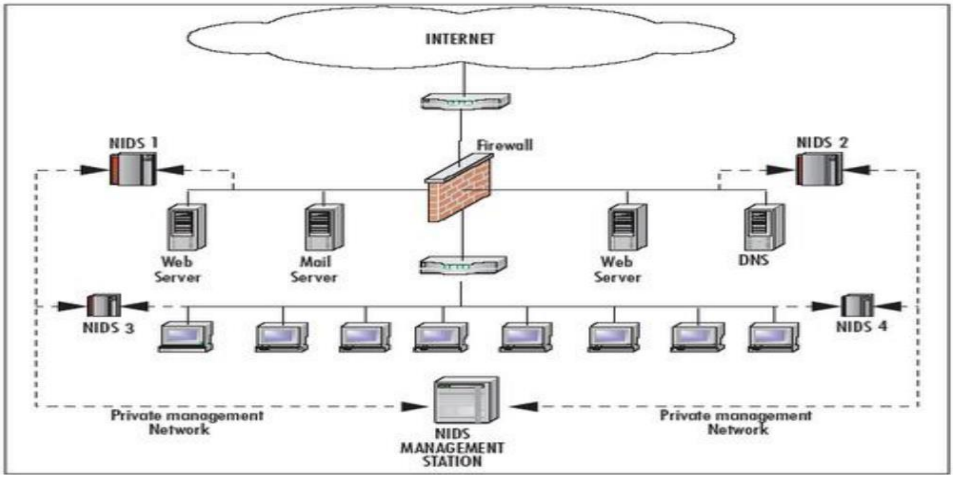


Hình 1.4 Host-based IDS

Host-based IDS tìm kiếm dấu hiệu của xâm nhập vào một host cục bộ; thường sử dụng các cơ chế kiểm tra và phân tích các thông tin được logging. Nó tìm kiếm các hoạt động bất thường như login, truy nhập file không thích hợp, bước leo thang các đặc quyền không được chấp nhận.

Kiến trúc IDS này thường dựa trên các luật (rule-based) để phân tích các hoạt động. Ví dụ đặc quyền của người sử dụng cấp cao chỉ có thể đạt được thông qua lệnh su-superuser, như vậy những cố gắng liên tục để login vào tài khoản root có thể được coi là một cuộc tấn công.

c) Distributed IDS (DIDS)



Hình 1.5 Distributed IDS

DIDS là dạng kết hợp của HIDS và NIDS, sử dụng các Sensor HIDS, NIDS hoặc kết hợp cả hai. Các Sensor được đặt từ xa và báo cáo với một trạm quản lý tập trung, logfile ghi các cuộc tấn công định kỳ được gửi lên các trạm quản lý và có thể được lưu trữ trong một cơ sở dữ liệu trung tâm.

Mạng liên lạc giữa các Sensor và trạm quản lý có thể là mạng riêng hoặc đường truyền mạng có sẵn. Khi thông tin truyền đi trên mạng chung sẽ được mã hóa hoặc sử dụng mạng riêng ảo (Virtual Private Network) để đảm bảo vấn đề an ninh.

Ngoài ra còn có một số dạng khác của IDS:

- Wireless IDS (WIDS).

- Network Behavior Analysis System (NBAS).

- Honeypot IDS.

2. Giới thiệu về IPS

2.1 Khái niệm

Một hệ thống chống xâm nhập (Intrusion Prevention System – IPS) là một phần mềm hoặc một thiết bị chuyên dụng có khả năng phát hiện sự xâm nhập, các cuộc tấn công và tự động ngăn chặn các cuộc tấn công. Phần lớn hệ thống IPS được đặt ở vành đai mạng, đủ khả năng bảo vệ tất cả các thiết bị trong mạng.

2.2 Phân loại IPS

+ NIPS (Network-Based IPS): thiết bị cảm biến được kết nối với các phân đoạn mạng để giám sát nhiều máy.

+ HIPS (Host-Based IPS): các phần mềm quản lý trung tâm được cài đặt trên mỗi máy chủ lưu trữ. Các máy chủ được bảo vệ và báo cáo với trung tâm quản lý giao diện điều khiển. HIPS cung cấp máy chủ lưu trữ cá nhân phát hiện và bảo vệ. HIPS không đòi hỏi phần cứng đặc biệt

2.3 Kiến trúc hệ thống IPS

Một hệ thống IPS được xem là thành công nếu chúng hội tụ được các yếu tố: thực hiện nhanh, chính xác, đưa ra các thông báo hợp lý, phân tích được toàn bộ thông lượng, cảm biến tối đa, ngăn chặn thành công và chính sách quản lý mềm dẻo.

Hệ thống IPS gồm 3 module chính: module phân tích luồng dữ liệu, module phát hiện tấn công, module phản ứng.

+ Module phân tích luồng dữ liệu: Module này có nhiệm vụ lấy tất các gói tin đi đến mạng để phân tích. Thông thường các gói tin có địa chỉ không phải của một card mạng thì sẽ bị card mạng đó huỷ bỏ nhưng card mạng của IPS được đặt ở chế độ thu nhận tất cả. Tất cả các gói tin qua chúng đều được sao chụp, xử lý, phân tích đến từng trường thông tin. Bộ phân tích đọc thông tin tìm trường trong gói tin, xác định chúng thuộc kiểu gói tin nào, dịch vụ gì... Các thông tin này được chuyển đến module phát hiện tấn công.

+ Module phát hiện tấn công: Đây là module quan trọng nhất trong hệ thống có nhiệm vụ phát hiện các cuộc tấn công. Có hai phương pháp để phát hiện các cuộc tấn công, xâm nhập là dò sự lạm dụng và dò sự bất thường.

+ Module phản ứng: Khi có dấu hiệu của sự tấn công hoặc thâm nhập, module phát hiện tấn công sẽ gửi tín hiệu báo hiệu có sự tấn công hoặc thâm nhập đến module phản ứng. Lúc đó module phản ứng sẽ kích hoạt tường lửa thực hiện chức năng ngăn

chặn cuộc tấn công hay cánh báo tới người quàn trị. Tại module này, nếu chỉ đưa ra các cánh báo tới các người quản trị và dừng lại ở đó thì hệ thống này được gọi là hệ thống phòng thủ bị động. Module phản ứng này tùy theo hệ thống mà có các chức năng và phương pháp ngăn chặn khác nhau.

3. So sánh IDS và IPS

Hệ thống IDS hoạt động thụ động, theo dõi dữ liệu truyền qua mạng, so sánh các traffic này với các rules được thiết lập khi phát hiện bất kì dấu hiệu bất thường nào, hệ thống sẽ ghi lại và gởi cảnh báo tới người quản trị. Một hệ thống IDS có thể phát hiện hầu hết các loại traffic độc hại bị tường lửa bỏ qua bao gồm các cuộc tấn công từ chối dịch vụ, tấn công dữ liệu trên các ứng dụng, đăng nhập trái phép máy chủ và các phần mềm độc hại như virus, trojan và worms.

Hầu hết các hệ thống IDS thường dựa trên các dấu hiệu xâm nhập và phân tích trạng thái của giao thức để phát hiện các đe dọa. Ưu điểm của IDS là không làm chậm mạng như IPS vì không phải là hệ thống nội tuyến. Tuy nhiên vấn đề chính của IDS là thường đưa ra báo động gia

Hệ thống IPS phát hiện và ngăn chặn các cuộc tấn công, là giải pháp hoàn chỉnh ngăn chặn tấn công. IPS có thể ngắt kết nối của kẻ tấn công vào hệ thống bằng cách chặn tài khoản người dùng, địa chỉ IP, hoặc các thuộc tính liên kết đến kẻ tấn công hoặc chặn tất cả các truy cập vào máy chủ, dịch vụ, ứng dụng.

Tuy nhiên có nhiều lí do để các công ty chọn hệ thống IDS thay vì IPS, mặc dù họ chắc chắn muốn có một hệ thống phát hiện và ngăn chặn các cuộc tấn công chứ không đơn thuần là ghi lại và cảnh báo. Trong đó hai lí do chính là: Hệ thống IPS nếu đưa ra các báo động sai đối với lưu lượng hợp pháp trên mạng chúng sẽ ngăn chặn các luồng thông tin này, trong khi hệ thống IDS chỉ cảnh báo và ghi lại các cuộc tấn công giả này. Thứ hai là một số quản trị viên hay nhà quản lí không muốn có một hệ thống tiếp nhận và thực hiện các quyết định thay cho họ, họ muốn nhận được cảnh báo, nhìn nhận vấn đề và đưa ra quyết định

CHƯƠNG 2. TÌM HIỂU VỀ SNORT

1. Giới thiệu về hệ thống Snort

Snort là một phần mềm IDS mã nguồn mở, được phát triển từ năm 1998 bởi Martin Roesch, người sáng lập của Sourcefire. Với tốc độ ấn tượng, sức mạnh cũng như hiệu năng mà Snort đã đạt được đà phát triển nhanh chóng. Bên cạnh đó, kiến trúc của Snort được thiết kế ở dạng module nên người dùng có thể tăng cường tính năng cho hệ thống Snort của mình bằng việc cài đặt hay viết thêm các module mới. Tính đến nay, với hơn 4 triệu lượt tải về và gần 400 ngàn người dùng đăng ký, Snort đã trở thành công nghệ phát hiện và phòng chống xâm nhập được sử dụng rộng rãi nhất trên thế giới.

Hình 2.1 Logo Snort

** Các đặc điểm của Snort:

+ Có thể chạy trên nhiều platform như: Linux, Windows, OpenBSD, FreeBSD, NetBSD, Solaris ...

+ Chạy được trên nhiều giao thức mạng: Ethernet, 802.11, Token, Ring, FDDI ...

+ Có khả năng phát hiện ra nhiều kiểu thăm dò và tấn công: buffer, overflow, DoS, portscan, ICMP ...

+ Phát hiện nhanh các nguy cơ theo thời gian thực.

+ Cung cấp cho nhà quản trị các thông tin cần thiết để có thể kịp thời xử lý các sự cố.

** Snort có thể hoạt động ở một số cơ chế:

+ **Sniffer mode**: là chế độ cho phép bạn có thể theo dõi và đọc các luồng dữ liệu ra vào hệ thống mạng được hiển thị trên màn hình điều khiển.

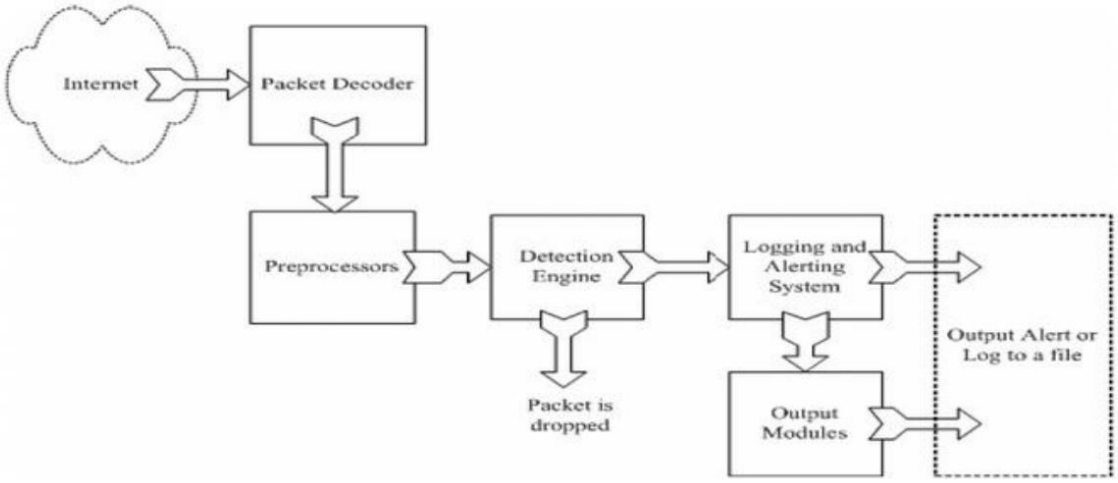
+ **Packet Logger mode**: cho phép ghi các logs dữ liệu vào đĩa lưu trữ.

+ **Network Intrusion Detection System (NIDS) mode**: là cơ chế được cấu hình phức tạp nhất, cho phép Snort phân tích các luồng dữ liệu, trong đó kiểm soát cho (hay không) cho phép các dữ liệu ra vào hệ thống mạng dựa vào các bộ qui tắc được định nghĩa bởi người quản trị, đồng thời thực hiện một vài hành động dựa vào những gì mà Snort nhìn thấy.

+ **Inline mode:** các gói tin thu từ iptables thay vì libpcap, sau đó iptables thực hiện hành động hủy hay cho phép các gói tin đi qua dựa trên những qui tắc được qui định và sử dụng bởi Snort.

2 Kiến trúc của Snort

Snort được chia thành nhiều module, mỗi module đảm nhận một chức năng riêng. Xét về mặt luận lý, kiến trúc của Snort được chia thành 5 module chính: Packet Decoder (Module Giải mã gói tin), Preprocessors (Module Tiền xử lý), Detection Engine (Module Phát hiện), Logging and Alerting System (Module Nhật ký và Cảnh báo), Output Modules (Module Kết xuất thông tin).



Hình 2.2 Kiến trúc Snort

2.1 Packet Decoder

Snort bắt các gói thông tin lưu thông trên mạng, Module Giải mã sẽ tiến hành giải mã các gói tin (lấy từ các giao diện mạng khác nhau: Ethernet, SLIP, PPP...).

Quá trình giải mã gói tin (Packet Decoder) trong Snort là một bước quan trọng để trích xuất thông tin từ các gói tin mạng và chuẩn bị dữ liệu cho quá trình phát hiện xâm nhập.

Dưới đây là các bước cơ bản trong quá trình giải mã gói tin của Module Decoder trong Snort:

+ Nhận gói tin mạng: Snort nhận các gói tin mạng từ nguồn dữ liệu, chẳng hạn như giao diện mạng hoặc tệp pcap.

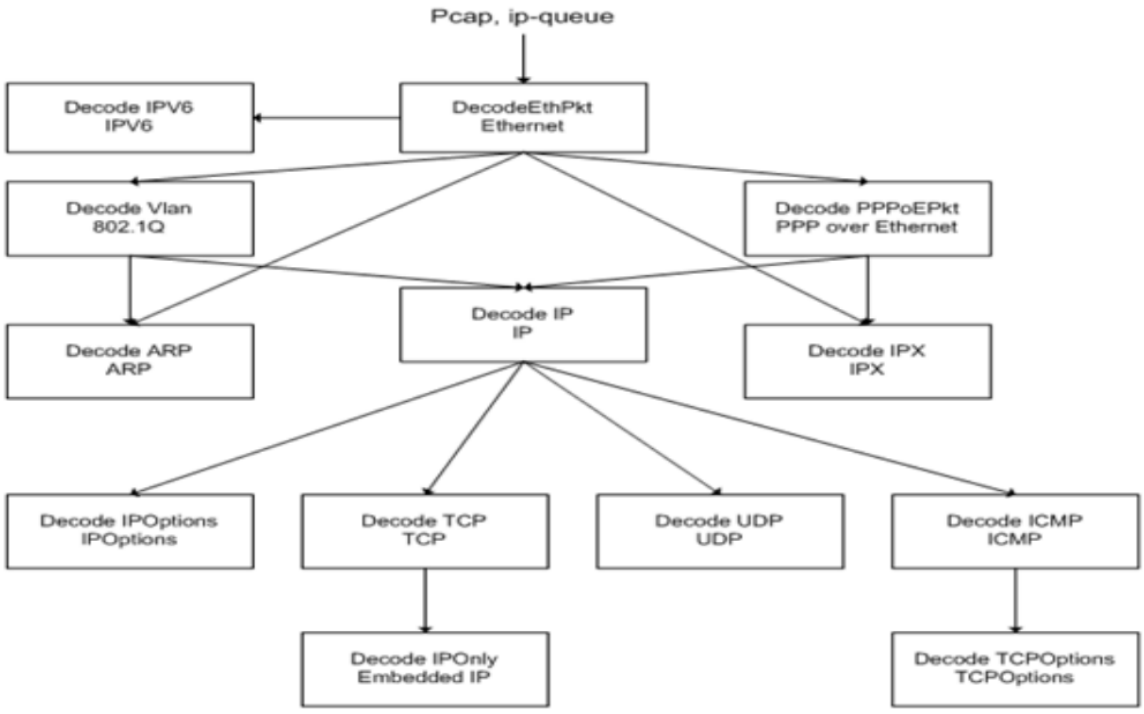
+ Xác định giao thức: Snort sẽ xác định giao thức của gói tin dựa trên các tiêu đề và dữ liệu gói tin. Các giao thức thông thường mà Snort hỗ trợ bao gồm TCP, UDP, ICMP, HTTP, FTP, SMTP, SSH, và nhiều giao thức khác.

+ Giải mã gói tin: Snort sử dụng các quy tắc và hàm giải mã tương ứng để trích xuất thông tin quan trọng từ gói tin. Các thông tin này bao gồm địa chỉ nguồn và đích, cổng nguồn và đích, cờ, số thứ tự, và các trường dữ liệu khác liên quan đến giao thức.

+ Xây dựng các đối tượng dữ liệu: Snort tạo ra các đối tượng dữ liệu từ thông tin giải mã, ví dụ như một luồng TCP hoặc một yêu cầu HTTP. Các đối tượng này sẽ được sử dụng bởi các module tiền xử lý và module phát hiện để kiểm tra các quy tắc và phát hiện các mẫu tấn công.

+ Chuyển tiếp dữ liệu: Sau khi quá trình giải mã hoàn tất, Snort sẽ chuyển tiếp dữ liệu đã giải mã cho các module tiền xử lý (Preprocessors) và module phát hiện (Detection Engine) để tiếp tục quá trình phát hiện xâm nhập.

Hình dưới đây mô tả quá trình giải mã gói tin Ethernet:



Hình 2.3 Quá trình giải mã gói tin Ethernet

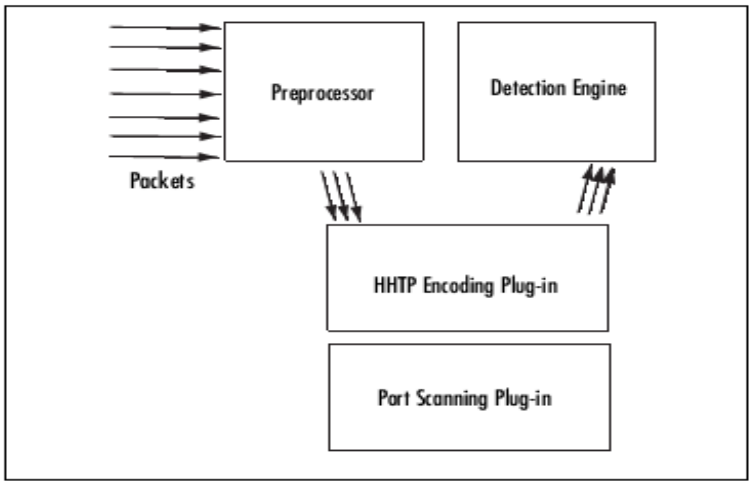
Quá trình giải mã gói tin (Packet Decoder) trong Snort là một phần quan trọng trong quá trình phát hiện xâm nhập, vì nó cho phép Snort nhìn thấy và hiểu được thông tin quan trọng từ gói tin mạng, từ đó tạo ra cơ sở để xác định các hành vi đáng ngờ và tấn công trong mạng.

Module này cũng có thể tạo ra các cảnh báo riêng dựa trên: tiêu đề giao thức bị thay đổi, gói tin quá dài, tùy chọn trong tiêu đề TCP bất thường hoặc không chính xác.

Gói tin sau khi giải mã sẽ được gửi đến Module Tiền xử lý.

2.2 Preprocessors

Module Tiền xử lý là một module rất quan trọng đối với bất kỳ một hệ thống IDS nào để có thể chuẩn bị gói dữ liệu đưa vào cho Module Phát hiện.



Hình 2.4. Quá trình xử lý ở Preprocessors

Ba chức năng chính của các module này là:

a) Kết hợp lại các gói tin.

Khi một lượng dữ liệu lớn được gửi đi, thông tin sẽ không đóng gói toàn bộ vào một gói tin mà phải thực hiện việc phân mảnh, chia gói tin ban đầu thành nhiều gói tin rồi mới gửi đi. Khi Snort nhận được các gói tin này nó phải thực hiện việc ghép nối lại để có được dữ liệu nguyên dạng ban đầu, từ đó mới thực hiện được các công việc xử lý tiếp.

Như đã biết khi một phiên làm việc của hệ thống diễn ra, sẽ có rất nhiều gói tin được trao đổi trong phiên đó. Một gói tin riêng lẻ sẽ không có trạng thái và nếu công việc phát hiện xâm nhập chỉ dựa hoàn toàn vào gói tin đó sẽ không đem lại hiệu quả cao. Module Tiền xử lý giúp Snort có thể hiểu được các phiên làm việc khác nhau

(nói cách khác đem lại tính có trạng thái cho các gói tin) từ đó giúp đạt được hiệu quả cao hơn trong việc phát hiện xâm nhập.

b) Giải mã và chuẩn hóa giao thức

Công việc phát hiện xâm nhập dựa trên dấu hiệu nhận dạng nhiều khi bị thất bại khi kiểm tra các giao thức dữ liệu có thể được thể hiện dưới nhiều dạng khác nhau. Ví dụ: một web server có thể chấp nhận nhiều dạng URL như URL được viết dưới dạng mã hexa/Unicode, URL chấp nhận cả dấu \ hay / hoặc nhiều ký tự này liên tiếp cùng lúc. Chẳng hạn có dấu hiệu nhận dạng "scripts/iisadmin", kẻ tấn công có thể vượt qua được bằng cách tùy biến các yêu cầu gửi đến web server như sau:

"scripts/./iisadmin"

"scripts/examples/../iisadmin"

"scripts\iisadmin"

"scripts/.\iisadmin"

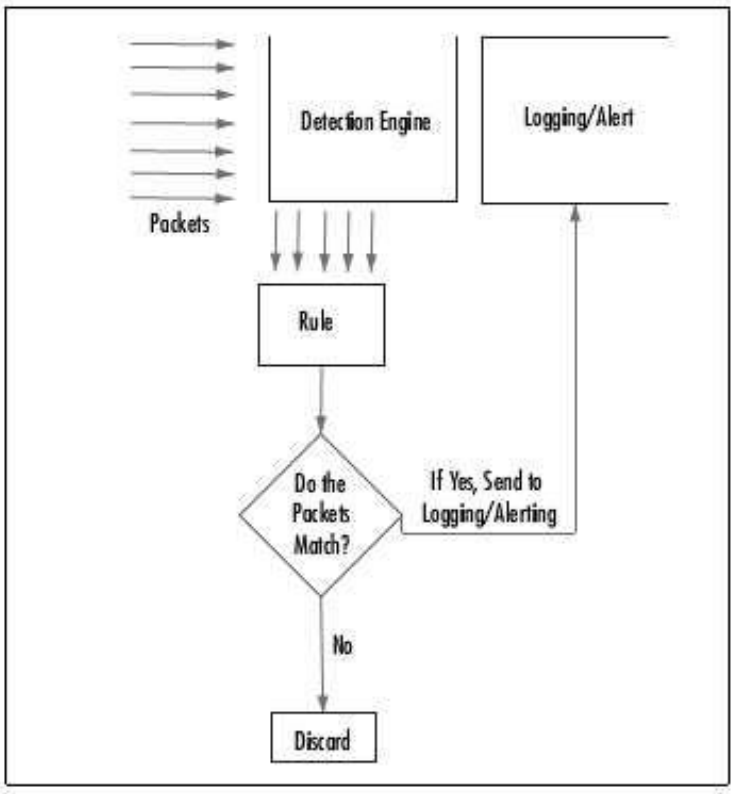
Hoặc thực hiện việc mã hóa các chuỗi này dưới dạng khác. Nếu Snort chỉ thực hiện đơn thuần việc so sánh dữ liệu với dấu hiệu nhận dạng sẽ xảy ra tình trạng bỏ sót các hành vi xâm nhập. Do vậy, một số Module Tiền xử lý của Snort phải có nhiệm vụ giải mã và chỉnh sửa, sắp xếp lại các thông tin đầu vào này để thông tin khi đưa đến Module Phát hiện có thể phát hiện được mà không bỏ sót. Hiện nay Snort đã hỗ trợ việc giải mã và chuẩn hóa cho các giao thức: TELNET, HTTP, RPC, ARP.

c) Phát hiện các xâm nhập bất thường

Các Module Tiền xử lý dạng này thường dùng để đối phó với các xâm nhập không thể hoặc rất khó phát hiện được bằng các luật thông thường hoặc các dấu hiệu bất thường trong giao thức.

2.3 Detection Engine

Đây là module quan trọng nhất của Snort. Nó chịu trách nhiệm phát hiện các dấu hiệu xâm nhập. Module Phát hiện sử dụng các luật được định nghĩa trước để so sánh với dữ liệu thu thập được từ đó xác định xem có xâm nhập xảy ra hay không. Rồi tiếp theo mới có thể thực hiện một số công việc như ghi log, tạo thông báo và kết xuất thông tin.



Hình 2.5.Gói tin được xử lý ở Detection Engine bằng các luật

Một vấn đề rất quan trọng trong Module Phát hiện là thời gian thực thi, xử lý các gói tin. Tùy thuộc vào hệ thống của bạn mạnh như thế nào mà sẽ tốn những khoảng thời gian khác nhau để xử lý, vì một IDS thường nhận được rất nhiều gói tin, tương ứng với việc cũng có rất nhiều các luật được thực thi. Nếu lưu lượng cần xử lý trên mạng là quá lớn khi Snort đang hoạt động trong chế độ NIDS, bạn có thể mất một vài gói tin hoặc thời gian đáp ứng không chính xác. Khả năng xử lý của Module Phát hiện phụ thuộc vào các yếu tố sau:

- Số lượng các luật.
- Sức mạnh của hệ thống máy mà Snort đang chạy.
- Tốc độ của bus hệ thống.
- Lưu lượng trên mạng.

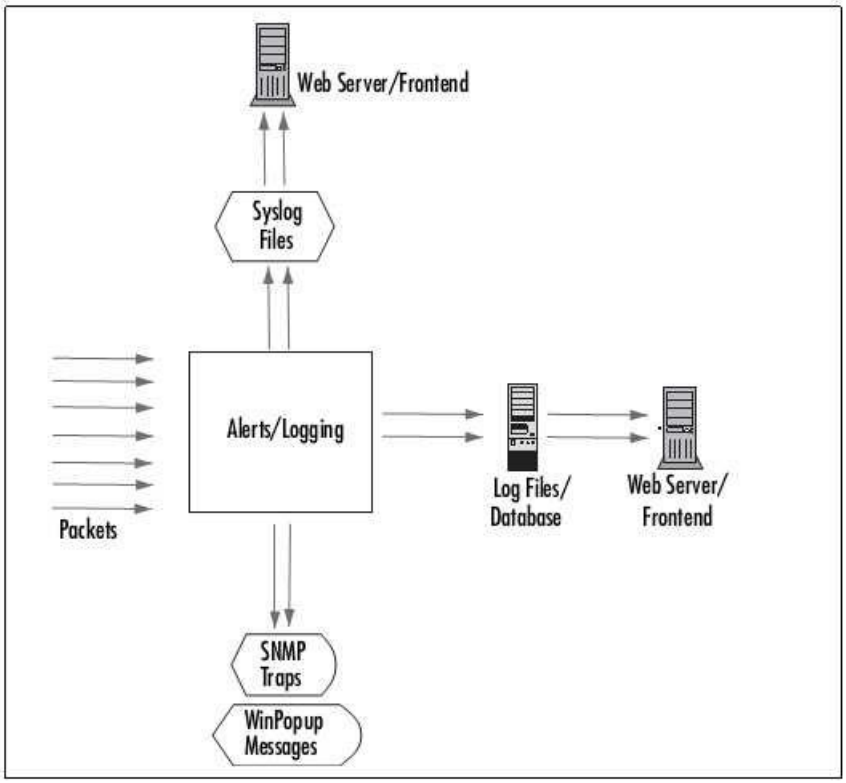
Một Module Phát hiện cũng có khả năng tách các phần của gói tin ra và áp dụng các luật lên từng phần nào của gói tin đó. Các phần đó có thể là:

- IP header.
- Header ở tầng giao vận: TCP, UDP.
- Header ở tầng ứng dụng: DNS header, HTTP header, FTP header,…
- Payload của gói tin.

Một vấn đề nữa trong Module Phát hiện đó là việc xử lý thế nào khi một gói tin bị phát hiện bởi nhiều luật. Do các luật trong Snort cũng được đánh thứ tự ưu tiên, nên một gói tin khi bị phát hiện bởi nhiều luật khác nhau, cảnh báo được đưa ra sẽ là cảnh báo ứng với luật có mức ưu tiên lớn nhất.

2.4 Logging and Alerting System.

Tùy thuộc việc Module Phát hiện có nhận dạng được xâm nhập hay không mà một gói tin có thể được ghi log hay đưa ra một cảnh báo. Các file log được lưu dưới các định dạng đơn giản như tcpdump hoặc một vài dạng khác, tất cả được lưu trữ trong folder mặc định /var/log/snort. Bạn có thể sử dụng tùy chọn –l trong command line để thay đổi vị trí tạo ra các logfile và cảnh báo.



Hình 2.6.Thành phần cảnh báo và logging

Cảnh báo có thể gửi thông qua pop-up, ghi vào logfile, SNMP traps, SMS, hoặc lưu trữ dưới dạng cơ sở dữ liệu SQL (chẳng hạn như MySQL).

2.5 Output Modules.

Module này có thể thực hiện các thao tác khác nhau tùy theo việc bạn muốn lưu kết quả xuất ra như thế nào. Tùy theo việc cấu hình hệ thống mà nó có thể thực hiện các công việc như là:

+ Ghi logfile.

+ Ghi syslog: syslog và một chuẩn lưu trữ các file log được sử dụng rất nhiều trên các hệ thống Unix, Linux.

+ Ghi cảnh báo vào cơ sở dữ liệu.

+ Tạo file log dạng XML: việc ghi logfile dạng XML rất thuận tiện cho việc trao đổi và chia sẻ dữ liệu.

+ Cấu hình lại router, firewall.

+ Gửi các cảnh báo được gói trong gói tin sử dụng giao thức SNMP. Các gói tin dạng SNMP này sẽ được gửi tới một SNMP server từ đó giúp cho việc quản lý các cảnh báo và hệ thống IDS một cách tập trung và thuận tiện hơn.

+ Gửi các thông điệp SMB (Server Message Block) tới các máy tính Windows.

Nếu không hài lòng với các cách xuất thông tin như trên, có thể viết các Module Kết xuất thông tin riêng tuỳ theo mục đích sử dụng.

3 Các chế độ hoạt động của Snort.

3.1 Sniffer mode.

Đây là một tính năng cơ bản dễ dàng sử dụng nhưng rất hiệu quả trong việc giám sát các luồng dữ liệu đang lưu thông trên hệ thống. Nó có khả năng tạo ra một bảng tóm tắt về các traffic trên mạng sau khi capture các gói tin, từ đó giúp người quản trị có cái nhìn tổng quan về hệ thống.

Bảng sau mô tả một vài từ khóa trong chế độ Sniffer mode

Khoá	Mô tả
-v	Đưa ra packet headers trong phần output
-d	Hiển thị packet payload
-a	Hiển thị ARP packets
-e	Hiển thị dữ liệu lớp data link

Bảng 2.1 Các tùy chọn trong chế độ Sniffer

+ Nếu bạn chỉ quan tâm đến thông tin của những gói tin TCP/IP packet headers, hãy bắt đầu với khóa -v: **./snort -v**

→ Lệnh này sẽ chạy Snort và chỉ hiển thị địa chỉ IP và các TCP/UDP/ICMP headers, ngoài ra không có gì khác.

+ Nếu bạn muốn theo dõi các dữ liệu application đang vận chuyển, hãy thêm khóa -d như sau: **./snort -vd**

→ Chỉ thị Snort trên sẽ hiển thị các gói dữ liệu cũng như các header.

+ Nếu bạn muốn hiển thị mô tả chi tiết hơn, như việc hiển thị cả header lớp liên kết dữ liệu, hãy sử dụng thêm khóa -e: **./snort -vde**

→ Các chuỗi thập lục phân hiển thị nhiều dữ liệu hơn, như địa chỉ MAC và địa chỉ IP. Khi thực hiện kiểm tra trên một mạng hoặc capture dữ liệu bằng Snort, việc bật -vde cung cấp nhiều thông tin nhất.

+ Để lưu lại trong logfile thay vì xuất ra console, sử dụng: **./snort -dve > sniffermode.log**

+ Ngoài ra, do một số thiết bị chuyển mạch có thể chia ra hay gộp chung các khóa trong một dòng lệnh cũng có thể nhập ra như: **./snort -d -v –e**

3.2 Packet Logger mode.

Nếu bạn muốn ghi lại các gói dữ liệu vào đĩa, bạn cần phải chỉ định một thư mục log và thêm khóa –l, Snort sẽ tự động biết để đi vào chế độ Packet logger:

./snort -dev -l file./log

Thư mục log mặc định trong Snort là /var/log/snort.

3.3 Network Intrusion Detection System (NIDS) mode.

Để kích hoạt chế độ NIDS dùng lệnh: **./snort -c ./snort.conf**

Trong đó snort.conf là tên của tập tin nơi mà bạn qui định các luật của Snort, tập tin này nằm trong /etc/snort. Tất cả các luật được qui định trong file snort.conf sẽ được áp đặt cho mỗi gói tin để quyết định một hành động sẽ được thực hiện hay không dựa trên các loại quy tắc này. Nếu không chỉ định một thư mục output cho chương trình, nó sẽ mặc định là /var/log/snort.

Lệnh này sẽ cấu hình Snort để chạy theo hình thức NIDS cơ bản nhất, ghi lại các log dữ liệu dựa theo các luật trong snort.conf theo định dạng ASCII vào đĩa lưu trữ bằng cách sử dụng một cấu trúc thư mục phân cấp (giống như chế độ Packet Logger).

**** NIDS Mode Output Options**

Có một số cách để cấu hình đầu ra output của Snort ở chế độ NIDS. Mặc định log sẽ được ghi theo định dạng ASCII và được đặt trong cơ chế cảnh báo toàn phần. Cơ chế cảnh báo toàn phần sẽ hiển thị thông điệp cảnh báo cùng với toàn bộ các

packet headers. Ngoài ra còn có các cơ chế cảnh báo bằng việc thực thi các dòng lệnh cũng như việc log các dữ liệu.

Cơ chế cảnh báo của Snort khá phức tạp với bảy chế độ hoạt động theo các dòng lệnh,đó là: full, fast, socket, syslog, console, cmg, và none. Sáu trong số các chế độ này thực hiện với khóa -a tại dòng lệnh.

Các gói tin có thể được lưu lại với định dạng ASCII hoặc định dạng mã nhị phân bằng khóa –b tại dòng lệnh command line. Để vô hiệu hóa toàn bộ các gói tin log, hãy sử dụng khóa –N.

Dưới đây là bảng các tùy chọn cảnh báo ở chế độ NIDS.

Option	Description
-a fast	Đưa ra một thông điệp cảnh báo với định dạng đơn giản trong thời gian ngắn cùng với mốc thời gian và địa chỉ IP nguồn/đích và số port kết nối.
-a full	Đưa ra một cảnh báo đầy đủ. Đây là chế độ mặc định nếu bạn không chỉ định chế độ hoạt động cụ thể cho hệ thống.
-a unsock	Gửi một thông báo đến một socket UNIX rằng có một chương trình khác có thể lắng nghe nó.
-a none	Tắt chế độ cảnh báo.
-a console	Gửi một cảnh báo "fast-style" ra màn hình console.
-a cmg	Tạo một cảnh báo "cmg style".

Bảng 2.2 Các tùy chọn cảnh báo ở chế độ NIDS

Để gửi cảnh báo đến syslog, sử dụng khóa -s. Cơ chế mặc định của syslog là LOG_AUTHPRIV và LOG_ALERT. Nếu bạn muốn cấu hình cơ chế khác cho syslog output, hãy sử dụng những chỉ thị output plugin trong file snort rules.

**** Chuẩn Alert Output Khi Snort tạo ra một thông điệp cảnh báo, nó thường dựa vào cấu trúc giống như sau:**

[**] [116:56:1] alert message [**]

+ Thông số đầu tiên "116" là Generator ID (số thứ tự cảnh báo), điều này cho phép user biết được những thành phần trong một thông điệp Snort. Các thông số GIDs được trình bày ở etc/generators trong source Snort.

+ Thông số thứ hai "56" là ID Snort (đôi khi được gọi là Signature ID). Quy tắc dựa trên SID được viết trực tiếp vào các quy định với tùy chọn sid. Trong trường hợp này, 56 đại diện cho rule có sid 56.

+ Thông số thứ ba "1" là số revision ID. Con số này chủ yếu được sử dụng viết ra các signatures, với mỗi hành động thực thi của các rule, nên tăng con số này để phân biệt các tùy chọn rev.

3.4 Inline mode.

Inline mode là một nhánh cơ chế hoạt động của Snort. Phiên bản Snort 2.3.0 RC1 tích hợp các hệ thống phòng chống xâm nhập (IPS) của Snort Inline vào hệ thống chính của Snort. Snort Inline kết hợp khả năng ngăn chặn của iptables vào bên trong Snort, sau đó đưa ra những bổ sung và thay đổi trong bộ qui tắc luật Snort để giúp iptables đưa ra những động cho phép hay hủy một gói tin.

Có ba loại quy tắc có thể sử dụng khi chạy Snort với Snort Inline:

+ DROP – Hành động DROP yêu cầu iptables loại bỏ gói tin và ghi lại thông tin như hành động LOG thông thường.

+ REJECT – Hành động REJECT yêu cầu iptables từ chối gói tin, ghi lại thông tin log và gửi một yêu cầu TCP reset nếu giao thức TCP hoặc gói tin ICMP không thể truy cập được port hoặc nếu giao thức là UDP, có nghĩa là iptables sẽ loại bỏ và gửi lại một thông báo cho nguồn gửi gói tin đó.

+ SDROP – Hành động SDROP cũng tương tự hành động DROP, điều khác biệt là ở chỗ Snort sẽ không ghi lại bất kì thông tin gì như hành động LOG.

4 Snort Rules.

4.1 Phương pháp phát hiện xâm nhập.

Sử dụng Snort rule khá đơn giản và linh hoạt nhưng nó chứa đựng một sức mạnh không nhỏ. Hầu hết các hoạt động tấn công hay xâm nhập vào hệ thống đều có dấu hiệu riêng. Căn cứ vào đó, người ta tạo ra các tập luật cho Snort. Các dấu hiệu có thể xuất hiện trong phần header của gói tin, phần nội dung hoặc nằm trên nhiều gói tin. Do vậy có một số nguyên tắc cần phải biết và nhớ để phát huy hết sức mạnh của rule trong Snort. Một luật có thể được sử dụng để tạo nên một thông điệp cảnh báo, log một thông điệp hay có thể bỏ qua một gói tin.

Snort sử dụng phương pháp phát hiện xâm nhập dựa trên luật (rule-based intrusion detection). Phương pháp này dựa trên việc sử dụng các luật (rules) để phát hiện các hoạt động xâm nhập trên mạng.

Dưới đây là mô tả cơ bản về phương pháp phát hiện xâm nhập trong Snort:

+ Luật (Rules): Snort sử dụng các luật để định nghĩa các mẫu xâm nhập hoặc các hoạt động đáng ngờ trên mạng. Mỗi luật bao gồm các thành phần chính như tiêu đề,

mẫu, tùy chọn và hành động. Các luật có thể được tạo ra bởi người dùng hoặc sử dụng từ các nguồn đáng tin cậy.

+ Kiểm tra và so khớp: Snort kiểm tra các gói tin mạng khi chúng đi qua hệ thống. Snort so khớp các gói tin với các mẫu xâm nhập được định nghĩa trong các luật.

+ Cảnh báo: Khi Snort phát hiện một hoạt động trùng khớp với một luật, nó thực hiện một hành động nhất định, chẳng hạn như ghi lại gói tin, gửi cảnh báo tới hệ thống quản trị hoặc thực hiện các biện pháp phòng ngừa. Hành động phát hiện có thể được tùy chỉnh để phù hợp với môi trường mạng cụ thể.

+ Học máy (Machine Learning) và Tự động hóa: Snort cũng hỗ trợ tích hợp các thuật toán học máy và công nghệ tự động hóa để phát hiện xâm nhập. Các kỹ thuật này cho phép Snort tự động học và phát hiện các mẫu xâm nhập mới dựa trên dữ liệu và kinh nghiệm tích lũy.

Phương pháp phát hiện xâm nhập dựa trên luật trong Snort cho phép người dùng tùy chỉnh và điều chỉnh các luật để phát hiện các mẫu xâm nhập cụ thể và thực hiện các hành động phòng ngừa khi cần thiết. Ngoài ra, Snort cũng hỗ trợ tích hợp các phương pháp phát hiện xâm nhập khác như phân tích đặc trưng, phân tích hành vi và phân tích thông tin độc hại để cung cấp sự linh hoạt và hiệu quả trong việc phát hiện xâm nhập trên mạng.

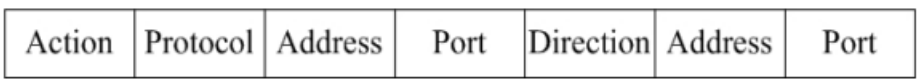
4.2 Cấu trúc luật của Snort.

Snort rules có 2 phần: rule header và rule options

Hình 2.7 Các thành phần của rules

a) Rule header: chứa thông tin về hành động mà luật đó sẽ thực hiện khi phát hiện ra có xâm nhập nằm trong gói tin và nó cũng chứa các tiêu chuẩn để áp dụng luật với gói tin đó.

Cấu trúc chung của một Rule header được thể hiện ở hình sau:



Hình 2.8 Cấu trúc của Rule header

- **Action**: là phần qui định loại hành động nào được thực thi khi các dấu hiệu của gói tin được nhận dạng chính xác bằng luật đó. Thông thường, các hành động tạo ra một cảnh báo hoặc log thông điệp hoặc kích hoạt một luật khác.
- **Protocol**: là phần qui định việc áp dụng luật cho các packet chỉ thuộc một giao thức cụ thể nào đó. Ví dụ như IP, TCP, UDP …
- **Address**: là phần địa chỉ nguồn và địa chỉ đích. Các địa chỉ có thể là một máy đơn, nhiều máy hoặc của một mạng nào đó. Trong hai phần địa chỉ trên thì một sẽ là địa chỉ nguồn, một sẽ là địa chỉ đích và địa chỉ nào thuộc loại nào sẽ do phần Direction "->" qui định.
- **Port**: xác định các cổng nguồn và đích của một gói tin mà trên đó luật được áp dụng.
- **Direction:** phần này sẽ chỉ ra đâu là địa chỉ nguồn, đâu là địa chỉ đích.

b) Rule Option: chứa một thông điệp cảnh báo và các thông tin về các phần của gói tin dùng để tạo nên cảnh báo, các tiêu chuẩn phụ để so sánh luật với gói tin.

Ví dụ:

alert icmp any any -> any any (msg: "Ping test";)

Giải thích:

- **Action**: "alert", một cảnh báo sẽ được tạo ra nếu như có hành động ping đến bất kỳ ở port nào (gói tin sẽ được ghi vào logfile mỗi khi cảnh báo được tạo ra).
- **Protoco**l: "icmp", luật chỉ áp dụng với các gói tin ICMP.
- **Source Address**: "any"; **Source port**: "any", luật áp dụng đối với tất cả gói tin có địa chỉ nguồn bất kỳ, đến từ port bất kỳ.
- **Destination Address**: "any"; **Destination port**: "any", luật áp dụng với các gói tin có địa chỉ đích bất kỳ, gửi đến port bất kỳ.
- **Option**: msg: "Ping test", một cảnh báo chứa dòng thông báo "Ping test" sẽ được tạo ra khi phát hiện có hành động ping trên hệ thống

4.2.1 Rule Header.

a) Action:

Là phần đầu tiên của luật, chỉ ra hành động nào được thực hiện khi mà các điều kiện của luật được thoả mãn. Một hành động được thực hiện khi và chỉ khi tất cả các điều kiện đều phù hợp.

Có 5 hành động đã được định nghĩa nhưng có thể tạo ra các hành động riêng tuỳ thuộc vào yêu cầu .

+ Pass: Hành động này hướng dẫn Snort bỏ qua gói tin này. Hành động này đóng vai trò quan trọng trong việc tăng cường tốc độ hoạt động của Snort khi mà ta không muốn áp dụng các kiểm tra trên các gói tin nhất định. Ví dụ ta sử dụng các bẫy (đặt trên một máy nào đó) để nhử các hacker tấn công vào thì ta phải cho tất cả

các gói tin đi đến được máy đó. Hoặc là dùng một máy quét để kiểm tra độ an toàn mạng của mình thì ta phải bỏ qua tất cả các gói tin đến từ máy kiểm tra đó.
+ Log: ghi nhật ký gói tin, có thể ghi vào file hay vào cơ sở dữ liệu tuỳ thuộc vào nhu cầu.
+ Alert: Gửi một thông điệp cảnh báo khi dấu hiệu xâm nhập được phát hiện, sau khi gửi thông điệp thì gói tin sẽ được ghi vào logfile.
+ Activate: sử dụng để tạo ra một cảnh báo và kích hoạt một luật khác kiểm tra thêm các điều kiện của gói tin.
+ Dynamic: chỉ ra đây là luật được gọi bởi các luật khác có hành động là Activate.
+ Các hành động do người dùng định nghĩa: một hành động mới được định nghĩa theo cấu trúc sau:

```
ruletype action_name
{
action definition
}
```

Trong đó, ruletype là từ khoá. Hành động được định nghĩa chính xác trong dấu ngoặc "{ }"

Ví dụ:

```
ruletype smb_db_alert
{
type alert
output alert_smb: workstation.list
output database: log, mysql, user=test password=test
dbname=snort host = localhost
}
```

Đây là hành động có tên là smb_db_alert dùng để gửi thông điệp cảnh báo dưới dạng cửa sổ pop-up SMB tới các máy có tên trong danh sách liệt kê trong file workstation.list và tới cơ sở dữ liệu MySQL tên là snort.

b) Protocol:

Là phần thứ hai của một luật có chức năng chỉ ra loại gói tin mà luật sẽ được áp dụng. Hiện tại Snort hiểu được các protocol sau: IP, ICMP, TCP, UDP.

c) Address:

Có hai phần địa chỉ trong một luật của Snort. Các địa chỉ này được dùng để kiểm tra nguồn sinh ra (Source Address) và đích đến (Destination Address) của gói tin. Địa chỉ có thể là địa chỉ của một IP đơn hoặc là địa chỉ của một mạng. Ta có thể dùng từ "any" để áp dụng luật cho tất cả các địa chỉ.

Trong hai địa chỉ của một luật Snort thì có một địa chỉ là địa chỉ nguồn và địa chỉ còn lại là địa chỉ đích. Việc xác định đâu là địa chỉ nguồn, đâu là địa chỉ đích thì phụ thuộc vào phần hướng (direction).

Ví dụ như luật:

alert tcp any any -> 192.168.1.10/32 80 (msg: “TTL=100”; ttl: 100;)

+ Luật trên sẽ tạo ra một cảnh báo đối với tất cả các gói tin từ bất kì nguồn nào có TTL = 100 đi đến web server 192.168.1.10 tại port 80.

+ Ta cũng có thể sử dụng dấu phủ định “!” để loại trừ một địa chỉ cụ thể hay một dãy địa chỉ nào đó. Ví dụ, luật sau sẽ áp dụng cho tất cả các gói tin ngoại trừ các gói có nguồn xuất phát từ mạng lớp C 192.168.2.0.

alert icmp ![192.168.2.0/24] any -> any any (msg: “Ping with TTL=100”; ttl: 100;)

d) Port:

Số hiệu port dùng để áp dụng luật cho các gói tin đến từ hoặc đi đến một port hay một phạm vi port cụ thể nào đó. Từ “any” cũng được dùng để đại diện cho tất cả các cổng.

Chú ý là số hiệu cổng chỉ có ý nghĩa trong các giao thức TCP và UDP thôi. Nếu protocol của luật là IP hay ICMP thì số hiệu cổng không đóng vai trò gì cả.

Ví dụ:

alert tcp 192.168.2.0/24 23 -> any any (content: “confidential”; msg: ”Detected confidential”;)

Ta có thể áp dụng luật cho dãy các port thay vì chỉ cho một port nào đó. Port bắt đầu và port kết thúc phân cách nhau bởi dấu hai chấm “:”.

Ví dụ :

alert udp any 1024:2048 -> any any (msg: “UDP ports”;)

e) Direction:

Chỉ ra đâu là nguồn đâu là đích, có thể là “->” hay “<-” hoặc “<>”. Trường hợp “<>” là khi ta muốn kiểm tra cả Client và Server.

4.2.2 Rule Option.

Sức mạnh, sự linh hoạt của Snort đều nằm ở Rule Option nên nó có một vai trò vô cùng quan trọng đối với Snort. Mỗi Rule Option sẽ cách nhau bằng “;”. Rule

Option và nội dung chính của option đó được cách bằng ":".

Rule options gồm có 4 phần lớn:

+ **General:** Cung cấp những thông tin về rule nhưng không có ảnh hưởng gì trong quá trình phát hiện sự bất ổn.

+ **Payload:** Tìm kiếm thông tin trong phần payload của packet.

+ **Non-Payload:** Tìm kiếm thông tin trong phần non-payload của packet.

+ **Post-Detection:** Xảy ra sau khi một rule được kích hoạt.

a) General.

+ **msg**

– Dùng để chứa chuỗi thông tin mà người quản trị đã khai báo, khi phát hiện sự bất thường trong gói tin nhận được thì Snort sẽ đưa chuỗi thông tin đó cho người quản trị.

– Cấu trúc: **msg: "message text";**

+ **reference**

– Dùng để tham chiếu đến hệ thống xác định định danh tấn công ở bên ngoài. Reference không có khả năng phát hiện tấn công. Các thông tin tham khảo có rất nhiều trên internet (CVE, Bugtraq,…), những hệ thống chứa thông tin tham khảo thì chứa các loại tấn công đã biết đến.

– Cấu trúc: **reference:<id system>,<id>;[reference: <id system>,<id>;]**

+ **sid**

– Dùng để xác định điều luật cụ thể của Snort, được sử dụng chung với từ khóa "rev".

- < 100: Dành cho tương lai
- 100 – 1 000 000: Thuộc định nghĩa của nhà phát triển Snort Sử dụng bởi nhà quản trị.
- > 1 000 000: Sử dụng bởi nhà quản trị.

– Cấu trúc: **sid: <Snort rules id>;**

+ **rev**

– Dùng để xác định phiên bản của luật. Nếu cập nhật luận mới thì từ khóa này sẽ giúp chúng ta phân biệt được các phiên bản.

– Cấu trúc: **rev: <revision integer>;**

+ **classtype**

– Dùng để phân loại một qui tắc như phát hiện một cuộc tấn công mà cuộc tấn công đó là một trong những loại tấn công đã được phân loại. Snort cung cấp một thiết lập các loại tấn công mặc định và được được sử dụng bởi các luật mặc định mà Snort cung cấp. Chúng ta tự xác định các loại rule của các classification sẽ tốt hơn các dữ liệu mà Snort đã cung cấp.

– Cấu trúc: **classtype: <class name>;**

– Phân loại tấn công được khai báo trong file classification.conf và có cấu trúc như sau: **Config classification: <class name>, <class description>, <default priority>**

– Độ ưu tiên 1 (cao) có sự ảnh hưởng lớn và độ ưu tiên 3 (thấp) thì ít bị ảnh hưởng (số ưu tiên càng nhỏ thì độ ưu tiên càng cao).

Ví dụ:

config classification: DoS, Denial of Service Attack, 2

alert udp any any -> 192.168.1.0/24 6838 (msg:"DoS"; content: "server"; classtype:DoS;)

+ **priority**

– Gán giá trị ưu tiên cho điều luật (độ ưu tiên trong classification là mặc định, còn từ khóa “priority” đóng vai trò ghi đè lên độ ưu tiên của classification)

– Cấu trúc: **priority: <priority integer>;**

b) Payload Detection Options.

+ **content**

– Đây là một trong những thành phần quan trọng của Snort. Nó cho phép người dùng thiết lập điều luật để tìm nội dung gói tin trong gói Payload và đưa ra hành động dựa trên dữ liệu đã được thiết lập.

– Content khá phức tạp, nội dung thông tin (content string) có thể viết dưới dạng ASCII hoặc nhị phân dưới dạng kí tự thập lục phân hoặc có thể trộn lẫn giữa ASCII hoặc nhị phân. Dữ liệu hệ nhị phân được đặt trong “|”.

– Những lưu ý khi sử dụng tùy chọn content:

- Lọc nội dung sẽ tốn nhiều tài nguyên và phải cẩn thận khi sử dụng quá nhiều luật để lọc nội dung.
- Nếu sử dụng bảng mã ASCII, chúng ta phải chú ý đến phím Caps Lock, dấu chấm.
- Có thể sử dụng nhiều từ khóa content ở một rule để tìm nhiều tín hiệu trong gói tin.
- Nội dung phù hợp là trường hợp nhạy cảm. Cấu trúc: content: [!] ""; Ví dụ: alert tcp any any -> any 80 (content:!"GET";)

– Cấu trúc: **content: [!] "<content string>";**

Ví dụ: alert tcp any any -> any 80 (content:!"GET";)

+ **nocase**

– Mục đích của từ khóa này là tìm kiếm mẫu cụ thể, mẫu này không phân biệt chữ hoa, thường. nocase có phạm vi ảnh hưởng tới các content trước đó.

– Cấu trúc: nocase;

Ví dụ:

alert tcp any any -> any 21 (msg:"FTP ROOT"; content:"USER root"; nocase;)

+ **rawbytes**

– Cho phép tìm kiếm trong gói dữ liệu thô (raw), nó bỏ qua bất kỳ một mã decoding nào mà được thực hiện bởi các preprocessor rawbytes nằm phía sau tùy chọn content.

– Cấu trúc: rawbytes;

Ví dụ:

alert tcp any any -> any 21 (msg: "Telnet NOP"; content: "|FF F1|"; rawbytes;)

+ **depth**

– Xác định số byte đầu tiên của gói tin phải kiểm tra. Từ khóa “depth” nằm phía sau tùy chọn “content”.

– Cấu trúc:**depth: <number>;**

+ **offset**

– Khai báo cho Snort biết rằng sẽ bắt đầu kiểm tra gói tin ở byte thứ mấy trong gói tin. Từ khóa "offset" nằm phía sau tùy chọn "content".

– Cấu trúc: **offset: <number>;**

+ **distance**

– Tùy chọn này tương tự tùy chọn "offset", điểm khác biết đó là "offset" chỉ cho bộ tìm kiếm biết là bắt đầu tìm kiếm từ vị trí thứ mấy tính từ đầu Payload, trong khi đó "distance" sẽ tính từ vị trí kết thúc của mẫu trước đó.

– Cấu trúc: **distance: <byte count>;**

+ **within**

– Tương tự như "depth" và nó được sử dụng kết hợp với "distance". Điểm khác biệt giữa "within" và "depth" là: "depth" tìm kiếm N byte tính từ đầu Payload, trong khi đó "within" tìm kiếm mẫu trong N byte tính từ vị trí kết thúc của mẫu trước đó.

– Cấu trúc: **within: <byte count>;**

+ **urilen**

– Dùng để xác định độ dài chính xác, ngắn nhất, dài nhất hay phạm vi của URI.

– Cấu trúc: **urilen: int<>int; urilen: [<,>] <int>;**

Ví dụ: urilen: 5 urilen: < 5 urilen: 5<>10

+ **isdataat**

– Kiểm tra Payload có dữ liệu ở một nơi được xác định trong Payload, tùy chọn dùng để tìm kiếm cho tới hết sự phù hợp nội dung trước đó. Sử dụng đối số là "relative" để tìm kiếm dữ liệu liên quan tới mẫu kết thúc của tùy chọn "content" trước đó.

– Cấu trúc: **isdataat:<int>[,relative];**

Ví dụ:

alert tcp any any -> any 111 (content:"PASS"; isdataat:50,relative; content:!"|0a|"; distance:0;)

+ **pcre**

Cho phép các rule được viết tương thích với dạng biểu thức của ngôn ngữ perl.

Cấu trúc: **pcre:[!]"(/<regex>/|m<delim><regex><delim>)[ismxAEGRUB]";**

+ **byte_test**

– Kiểm tra một trường byte đối với một giá trị cụ thể (sử dụng các toán tử). Khả năng kiểm tra giá trị phân hay chuyển đổi của chuỗi byte thành giá trị nhị phân tương đương và kiểm tra chúng.

– Cấu trúc: **byte_test: <bytes to convert>, [!]<operator>, <value>, <offset> [,relative] [,<endian>] [,<number type>, string];**

+ **byte_jum**

– Tùy chọn này thực hiện lấy một số byte, biến đổi chúng thành dạng số, thiết lập con trỏ trỏ tới vị trí là giá trị số của các byte này, để sử dụng cho khả năng phát hiện xâm nhập sau đó.

– Cấu trúc: **byte_jump: <bytes_to_convert>, <offset>** \

[,relative] [,multiplier <multiplier value>] [,big] [,little][,string] \

[,hex] [,dec] [,oct] [,align] [,from_beginning];

c) Non-Payload Detection Options.

+ **fragoffset**

– Tùy chọn này cho phép so sánh trường IP fragment offset trong phần đầu gói tin IP với một giá trị ở dạng cơ số 10.

– Cấu trúc: **fragoffset:[<|>]<number>;**

Ví dụ:

alert ip any any -> any any (msg: "First Fragment"; fragbits: M; fragoffset: 0;)

+ **ttl**

– Để kiểm tra thời gian sống của IP.

– Cấu trúc: **ttl:[[<number>-]><=]<number>;**

+ **tos**

– Dùng để kiểm tra trường TOS trong phần đầu của gói tin IP với một giá trị cụ thể.

– Cấu trúc: **tos:[!]<number>;**

+ **id**

– Dùng để kiểm tra trường ID trong phần đầu của gói tin IP với một giá trị cụ thể.

– Cấu trúc: **id:<number>;**

Ví dụ: id:31337;

(31337 là ID hacker thường sử dụng)

+ **ipopts**

– Kiểm tra IP cụ thể với các tùy chọn. Các tùy chọn có thể được kiểm tra:

- 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

– Cấu trúc: **popts:<rr|eol|nop|ts|sec|esec|lsrr|ssrr|satid|any>;**

Ví dụ: ipopts:lsrr;

+ **fragbits**

– Tùy chọn này được sử dụng để kiểm tra các bit (cờ) phân mảnh (fragmentation) và dự phòng (reserved) có được thiết lập trong phần đầu của gói tin IP không.

- M – Phân mảnh nhiều
- D – Không phân mảnh
- R – Bit dành riêng

– Các tùy chỉnh có thể thiết lập để thay đổi phù hợp tiêu chuẩn.

- +: Phù hợp với một hoặc nhiều bit đã quy định.
- *: phù hợp nếu các bit đã quy định được thiết lập
- !: Phù hợp nếu các bit đã quy định không được thiết lập.

– Cấu trúc: **fragbits:[+*!]<[MDR]>;**

Ví dụ: fragbits:MD+;

+ **dsize**

– Dùng để kiểm tra kích thước Payload, “dsize” được sử dụng để kiểm tra sự bất thường về kích thước của gói tin, thường dùng để kiểm tra buffer overflows.

– Cấu trúc: **dsize: [<>]<number>[<><number>];**

Ví dụ: dsize:300<>400;

+ **flags**

– Tùy chọn này được sử dụng để kiểm tra các cờ (flag bit) cụ thể của giao thức TCP. Các bit sau đó có thể được kiểm tra:

- F – FIN (LSB in TCP Flags byte)
- S – SYN
- R – RST
- P – PSH
- A – ACK
- U – URG
- 1 – Reserved bit 1 (MSB trong TCP Flags)
- 2 – Reserved bit 2
- 0 – No TCP Flags Set

– Các tùy chỉnh có thể thiết lập để thay đổi phù hợp tiêu chuẩn:

- +: Phù hợp với một hoặc nhiều bit đã quy định.
- *: phù hợp nếu các bit đã quy định được thiết lập.
- !: Phù hợp nếu các bit đã quy định không được thiết lập.

– Cấu trúc: **[!|*|+]<FSRPAU120>[,<FSRPAU120>];**

Ví dụ: alert tcp any any -> any any (flags:SF,12;)

+ **flow**

– Sử dụng kết hợp với TCP stream reassembly, chỉ có thể áp dụng lên một số hướng của luồng dữ liệu. Luật này chỉ có thể áp dụng cho clients và servers. “flow” cho phép gói tin liên qua tới các máy client trong mạng đang duyệt web để phân biệt từ các servers đang chạy trong mạng

– Cấu_trúc:

flow:[(established|stateless)][,(to_client|to_server|from_client|from_server)] \ [,(no_stream|only_stream)];

+ **flowbits**

– Dùng để theo dõi các máy tính trong suốt quá trình thực hiện phiên giao thức trao đổi. Tùy chọn flowbits hữu ít nhất cho TCP sessions, nó cho phép luật theo dõi trạng thái của một ứng dụng giao thức.

– Cấu trúc:

flowbits:[set|unset|toggle|isset|reset|noalert][,<STATE_NAME>];

– Có 7 từ khóa liên quan tới flowbits:

- set: Thiết lập trạng thái cụ thể của luồng đang hoạt động.
- unset: Không thiết lập trạng thái cụ thể của luồng đang hoạt động.
- toggle: Thiết lập trạng thái cụ thể nếu trạng thái chưa thiết lập và ngược lại.
- isset: Kiểm tra nếu trạng thái cụ thể đã thiết lập.
- isnotset: Kiểm tra nếu trạng thái cụ thể chưa thiết lập.
- noalert: Bởi vì luật không tạo ra cảnh báo, bất chấp các tùy chọn phát hiện còn lại.

+ **seq**

– Dùng để kiểm tra số thứ tự (sequence number) của một kết nối TCP.

– Cấu trúc: **seq:<number>;**

+ **ack**

– Dùng để kiểm tra số TCP acknowledge.

– Cấu trúc: **ack: <number>;**

+ **window**

– Dùng để kiểm tra TCP windows size.

– Cấu trúc: **window:[!]<number>;**

+ **icmp_id**

– Dùng để kiểm tra giá trị trường ID của thông báo ICMP, nó rất hữu dụng bởi vì một số kênh chương trình chuyển đổi sử dụng ICMP khi giao tiếp, được phát triển để phát hiện sự tấn công DDoS.

– Cấu trúc: **icmp_id:<number>;**

+ **icmp_seq**

– Dùng để kiểm tra số thứ tự (sequence value) cụ thể của gia thức ICMP, nó rất hữu dụng bởi vì một số kênh chương trình chuyển đổi sử dụng ICMP khi giao tiếp, được phát triển để phát hiện sự tấn công DDoS.

– Cấu trúc: **icmp_seq:<number>;**

+ **ip_proto**

– Dùng để kiểm tra IP protocol header. Danh sách protocol được quy định bằng tên.

– Cấu trúc:**ip_proto:[!|>|<] <name or number>;**

+ **stream_size**

– Dùng để kiểm tra sự phù hợp số byte chạy trên luồng, như xác định số TCP sequence. Stream_size hoạt động khi Stream5 preprocessor được bật.

– Cấu trúc: **stream_size:<server|client|both|either>,<operator>,<number>**

- < operator>
- < – less than
- – greater than
- = – equal
- != – not
- <= – less than or equal
- >= – greater than or equal

d) Post Detection Options.

+ **logto**

– Ghi nhận tất cả các gói và xuất ra file.

– Cấu trúc: **logto:"filename";**

+ **session**

– Dùng để lấy dữ liệu của user từ TCP Sessions (telnet, rlogin, ftp,…).

– Cấu trúc: **session: [printable|all];**

+ **resp**

– Dùng để đóng phiên làm việc khi cảnh báo được bật. Trong Snort resp được mệnh danh là sự đáp ứng linh hoạt.

– resp_mechanism:

- rst_snd: Gửi gói TCP-RST tới socket bên gửi.
- rst_rcv: Gửi gói TCP-RST tới socket bên nhận.
- rst_all: Gửi gói TCP-RST tới socket của 2 bên.
- icmp_net: Gửi một ICMP NET UNREACH tới bên nhận.
- icmp_host: Gửi một ICMP HOST UNREACH tới bên nhận.
- icmp_port: Gửi một ICMP PORT UNREACH tới bên nhận.
- icmp_all: Gửi tất cả gói ICMP ở trên tới bên nhận.

– Cấu trúc:
resp:<resp_mechanism>[,<resp_mechanism>[,<resp_mechanism>]];

+ **react**

– Mặc định thì Snort sẽ đóng các kết nối, khóa vài vị trí hoặc dịch vụ và gửi cảnh báo đến trình duyện web của user (nếu như truy cập web).

- block – Đóng kết nối và gửi thông báo.
- warn – Gửi thông báo.
- msg – Bao gồm từ khóa msg để block.
- proxy – Dùng port porxy để gửi thông báo.

– Cấu trúc: **react: block[, <react_additional_modifier>];**

Ví dụ: alert tcp any any <> 192.168.1.0/24 80 (content: "bad.htm"; \

msg: "Not for children!"; react: block, msg, proxy 8000;)

+ Ngoài ra còn một số tùy chọn khác như: **tag, activates, activated_by, count, replace, detection_filter.**

4.3 Đánh giá tập luật

Để đánh giá một tập luật trong Snort hoặc bất kỳ hệ thống phát hiện xâm nhập nào, có một số yếu tố quan trọng cần chú ý. Dưới đây là những yếu tố quan trọng trong việc đánh giá tập luật:

+ Hiệu suất: Đánh giá hiệu suất của tập luật bao gồm khả năng phát hiện các mẫu xâm nhập, độ chính xác (sự nhận dạng chính xác của hoạt động xâm nhập), độ nhạy (khả năng phát hiện xâm nhập mà không tạo ra quá nhiều cảnh báo sai), và tốc độ xử lý của hệ thống phát hiện.

+ Khả năng phản ứng: Xem xét khả năng phản ứng của tập luật khi phát hiện một hoạt động xâm nhập. Hành động phản ứng có thể bao gồm ghi lại thông tin, gửi cảnh báo cho quản trị viên, chặn kết nối hoặc thực hiện các biện pháp phòng ngừa khác.

+ Độ tin cậy: Đánh giá độ tin cậy của tập luật, bao gồm nguồn gốc và độ tin cậy của các luật được sử dụng. Luật phải được tạo ra bởi các nguồn đáng tin cậy, được kiểm tra và đảm bảo tính chính xác.

+ Khả năng tùy chỉnh: Xem xét khả năng tùy chỉnh và mở rộng tập luật. Hệ thống phát hiện xâm nhập phải hỗ trợ việc thêm, chỉnh sửa hoặc xóa các luật để phù hợp với môi trường mạng cụ thể.

+ Cập nhật và duy trì: Đánh giá khả năng cập nhật và duy trì của tập luật. Hệ thống phát hiện xâm nhập cần có cơ chế để cập nhật các luật mới nhất để đối phó với các mối đe dọa mới.

+ Tiêu chuẩn và tuân thủ: Xem xét liệu tập luật tuân thủ các tiêu chuẩn và hướng dẫn bảo mật quan trọng. Việc tuân thủ các tiêu chuẩn giúp đảm bảo tính an toàn và hiệu quả của tập luật.

+ Độ phù hợp với môi trường mạng: Đánh giá độ phù hợp của tập luật với môi trường mạng cụ thể. Mỗi mạng có các đặc điểm riêng, vì vậy tập luật cần được tùy chỉnh và cấu hình phù hợp để đáp ứng yêu cầu và môi trường cụ thể.

Quan trọng nhất, đánh giá tập luật trong Snort hoặc bất kỳ hệ thống phát hiện xâm nhập nào đòi hỏi kiến thức sâu về mạng, an ninh thông tin và kiến thức về các mối đe dọa cụ thể mà bạn muốn phát hiện.

4.4 Dư thừa nội dung trong tập luật

Trong Snort, dư thừa nội dung trong tập luật (redundant content in rules) là một vấn đề mà người dùng cần chú ý để tối ưu hóa hiệu suất của hệ thống phát hiện xâm nhập. Dư

thừa nội dung xảy ra khi một số luật chứa các mẫu trùng lặp hoặc trùng lặp các quy tắc xác định.

Dư thừa nội dung có thể xảy ra trong một số trường hợp, ví dụ:

+ Quá nhiều luật chứa cùng một mẫu: Một số luật có thể chứa cùng một mẫu xâm nhập. Việc sử dụng quá nhiều luật với cùng một mẫu có thể dẫn đến tình trạng dư thừa nội dung và làm tăng tải hệ thống mà không cần thiết.

+ Luật chứa các mẫu trùng lặp: Trong một số trường hợp, các luật có thể chứa các mẫu trùng lặp hoặc tương tự nhau. Điều này dẫn đến việc xử lý trùng lặp và không hiệu quả trong quá trình phát hiện xâm nhập.

Để giảm dư thừa nội dung trong tập luật Snort, bạn có thể thực hiện các biện pháp sau:

+ Kiểm tra và loại bỏ các luật trùng lặp: Xem xét tập luật và xác định những luật có cùng mẫu hoặc cùng mục tiêu. Loại bỏ các luật trùng lặp không cần thiết để giảm dư thừa nội dung.

+ Tối ưu hóa luật: Đánh giá các luật hiện có và tối ưu hóa chúng để giảm sự trùng lặp nội dung. Bạn có thể sử dụng các công cụ và kỹ thuật tối ưu hóa luật để loại bỏ dư thừa và tối ưu hóa hiệu suất.

+ Sử dụng biểu thức chính quy: Thay vì chỉ định cùng một mẫu trong nhiều luật, bạn có thể sử dụng biểu thức chính quy để tạo ra một mẫu chung và áp dụng nó cho nhiều luật. Điều này giúp giảm sự trùng lặp và tăng hiệu suất.

+ Xem xét việc sử dụng các quy tắc hợp nhất (suppression rules): Các quy tắc hợp nhất là các quy tắc được sử dụng để ngăn chặn cảnh báo của một luật cụ thể khi một luật khác phát hiện cùng một hoạt động. Sử dụng quy tắc hợp nhất có thể giúp giảm sự trùng lặp nội dung và giảm số lượng cảnh báo không cần thiết.

Tổng quát, để giảm dư thừa nội dung trong tập luật Snort, bạn cần xem xét, loại bỏ hoặc tối ưu hóa các luật trùng lặp và sử dụng các kỹ thuật tối ưu hóa để tăng hiệu suất và giảm tải trên hệ thống phát hiện xâm nhập.

CHƯƠNG 3: THỰC NGHIỆM

3.1. Mô hình thực nghiệm:

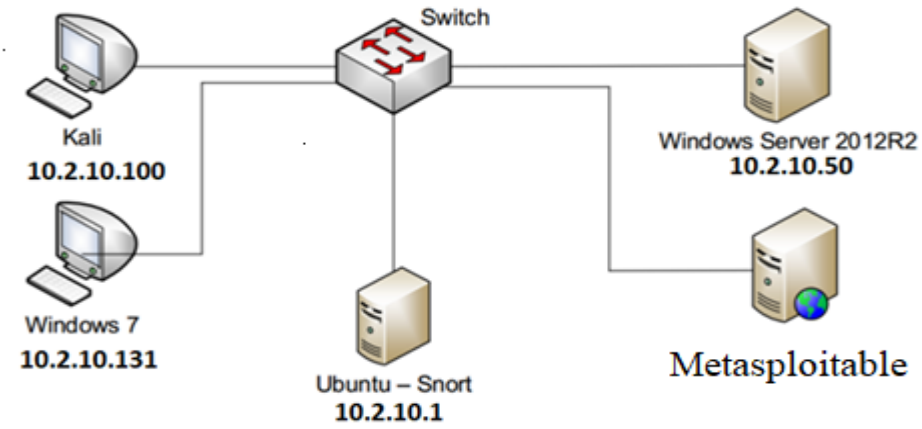
+ Để đảm bảo an toàn cho mạng máy tính nhằm phát hiện các cuộc tấn công vào mạng nội bộ, cần triển khai hệ thống phát hiện xâm nhập. Hệ thống phát hiện xâm nhập có thể là thiết bị chuyên dụng hoặc dưới dạng phần mềm. Trong mô hình mạng thử nghiệm nghiên cứu và học tập thì phần mềm miễn phí Snort là phù hợp.

+ Chuẩn bị

- 1 máy ảo chạy Ubuntu 22.04 (có cài đặt snort)
- 1 máy ảo chạy hệ điều hành Window 7
- 1 máy ảo chạy hệ điều hành Window Server 2012
- 1 máy ảo hệ điều hành Kalo linux
- 1 máy Metasploitable-linux-2.0.0

Metasploitable là một hệ thống ảo được phát triển nhằm mục đích cho các chuyên gia bảo mật và nhà phát triển phần mềm kiểm thử tấn công và phát hiện lỗ hổng bảo mật. Hệ thống này được cấu hình để có nhiều lỗ hổng bảo mật và dễ dàng bị tấn công, giúp cho người dùng có thể tìm hiểu và thực hành kỹ năng tấn công, cũng như học cách phát hiện và khắc phục các lỗ hổng bảo mật trên các hệ thống thực tế.

+ Mô hình cài đặt



+ Cài đặt phần mềm phát hiện Snort trên Ubuntu

- Cài đặt snort

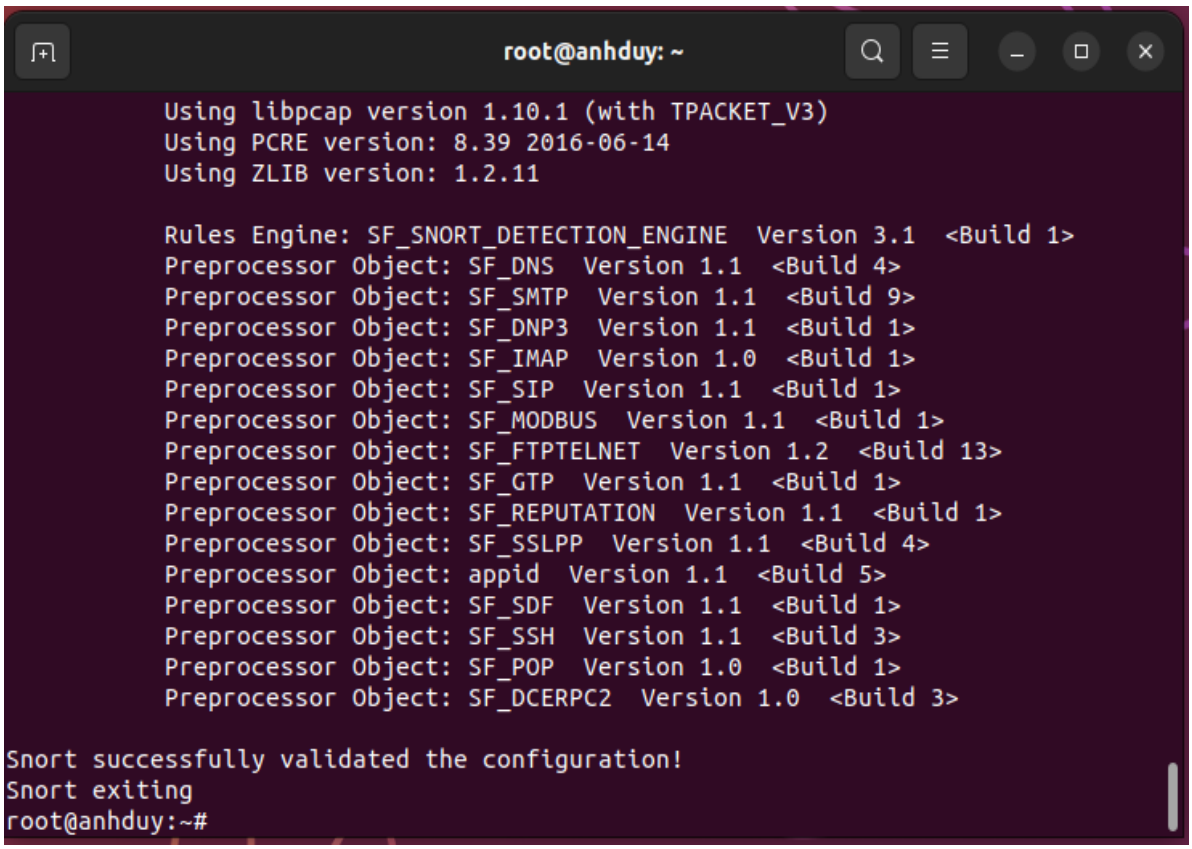
```
sudo apt update
sudo apt install snort -y
```

- Tải và thêm các rules của snort

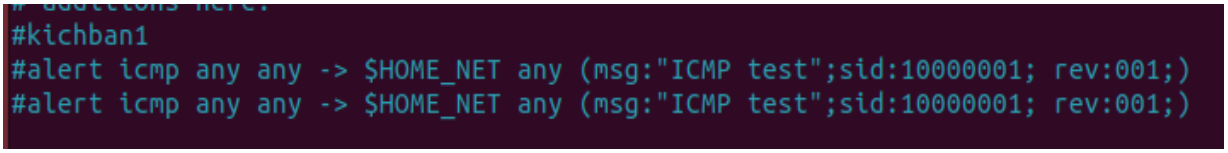
wget https://www.snort.org/rules/community -O ~/community.tar.gz
=> Lưu về dưới tập tin nén

– Giải nén
sudo tar -xvf ~/community.tar.gz -C ~/
sudo cp ~/community-rules/* /etc/snort/rules

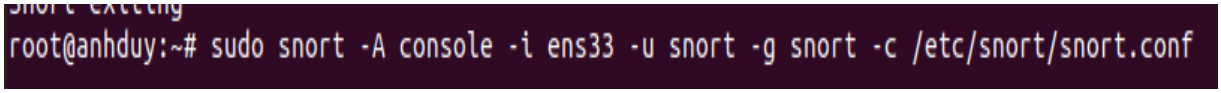
– Chạy thử snort
sudo snort -T -c /etc/snort/snort.conf (thành công)



– Thêm rules phát hiện tấn công bằng lênh ping:
sudo nano /etc/snort/rules/local.rules
alert icmp any any -> $HOME_NET any (msg:"ICMP test";sid:10000001; rev:001;)



– Thực thi snort :



– Trên máy kali thực hiện lệnh ping tới máy ubuntu

```
┌──(pad㉿kali)-[~]
└─$ ping 10.2.10.1
PING 10.2.10.1 (10.2.10.1) 56(84) bytes of data.
64 bytes from 10.2.10.1: icmp_seq=1 ttl=64 time=0.304 ms
64 bytes from 10.2.10.1: icmp_seq=2 ttl=64 time=0.806 ms
64 bytes from 10.2.10.1: icmp_seq=3 ttl=64 time=0.945 ms
64 bytes from 10.2.10.1: icmp_seq=4 ttl=64 time=0.271 ms
^C
── 10.2.10.1 ping statistics ──
4 packets transmitted, 4 received, 0% packet loss, time 3038ms
rtt min/avg/max/mdev = 0.271/0.581/0.945/0.298 ms
```

– Kết quả snort đã phát hiện lệnh Ping

```
Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
Preprocessor Object: appid  Version 1.1  <Build 5>
Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
Preprocessor Object: SF_POP  Version 1.0  <Build 1>
Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Commencing packet processing (pid=3102)
06/23-11:26:52.197061  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 10.2.10.100 -> 10.2.10.1
06/23-11:26:52.197090  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 10.2.10.1 -> 10.2.10.100
06/23-11:26:53.210366  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 10.2.10.100 -> 10.2.10.1
06/23-11:26:53.210395  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 10.2.10.1 -> 10.2.10.100
06/23-11:26:54.233667  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 10.2.10.100 -> 10.2.10.1
06/23-11:26:54.233703  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 10.2.10.1 -> 10.2.10.100
06/23-11:26:55.234991  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 10.2.10.100 -> 10.2.10.1
06/23-11:26:55.235035  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {ICMP} 10.2.10.1 -> 10.2.10.100
^C
*** Caught Int-Signal
===============================================================================
Run time for packet processing was 19.12043 seconds
Snort processed 13 packets.
```

– Trên máy snort phải xuất hiện cảnh báo về lệnh ping và trong thư mục /var/log/snort phải có file log về sự kiện trên

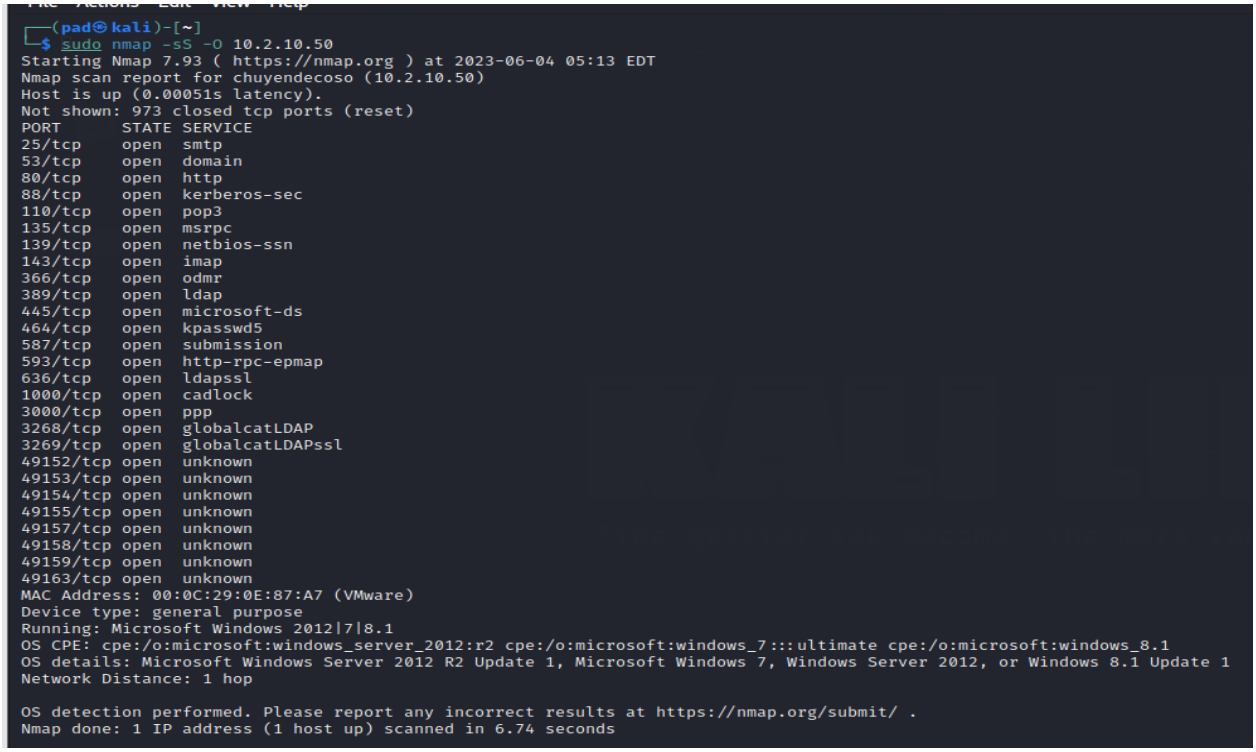
– Qua thực nghiệm trên , snort đã được cài đặt và hoạt động một các tốt nhất

3.2. Các kịch bản thực hiện tấn công và phát hiện

3.2.1 Phát hiện tấn công dò quét dịch vụ và cổng

Bước 1. Thực hiện tấn công

Tại bước này, sử dụng Nmap tại máy Kali để dò quét các dịch, cổng đăng mở và hệ điều hành đang chạy đối với mỗi địa chỉ IP . Tấn công dò quét tới máy chủ Windows Server 2012. (IP 10.2.10.50)



Bước 2. Thiết lập luật phát hiện tấn công dò quét cho Snort

+ Với hình thức tấn công trên Nmap gửi rất nhiều gói tin với cờ SYN với máy chủ Windows Server 2012 và Ubuntu web.

+ Sử dụng trình soạn thảo nano hoặc vi chỉnh sửa tệp tin local.rules và thêm vào luật phát hiện như sau:

alert tcp any any -> $HOME_NET any (msg:"SYN scan attack"; detection_filter:track by_src, count 100, seconds 2; flags:S; classtype:network-scan; sid:10000002;rev:1;)

+ Tập luật trên là một tập luật để phát hiện tấn công "SYN scan" (quét SYN) trên giao thức TCP. Dưới đây là giải thích từng phần trong tập luật:

− alert: Loại tập luật, thông báo khi có sự khớp.

− tcp any any -> $HOME_NET any: Quy định các gói tin TCP đi từ bất kỳ địa chỉ nguồn và bất kỳ cổng nguồn nào đến địa chỉ mạng nội bộ (được xác định bởi biến $HOME_NET) trên bất kỳ cổng đích nào.

− msg:"SYN scan attack": Một thông điệp mô tả tấn công được phát hiện.

− detection_filter:track by_src, count 100, seconds 2: Một bộ lọc phát hiện để giới hạn số lượng cảnh báo. Nếu có 100 hoặc nhiều hơn cảnh báo từ cùng một nguồn trong vòng 2 giây, chỉ cảnh báo đầu tiên sẽ được gửi.

− flags:S: Điều kiện cho phép tập luật phù hợp với các gói tin TCP có cờ (flags) là S, tương ứng với cờ SYN. Cờ SYN được sử dụng trong quá trình bắt tay (handshake) TCP để thiết lập kết nối.

− classtype:network-scan: Loại phân loại tấn công, chỉ định rằng tấn công liên quan đến quét mạng.

− sid:10000002: Số ID duy nhất cho tập luật, sử dụng để xác định một cách duy nhất tập luật trong các cơ sở dữ liệu và nhật ký.

− rev:1: Số phiên bản của tập luật.

+ Tập luật trên sẽ phát hiện các gói tin TCP có cờ SYN đi từ cùng một nguồn và đến mạng nội bộ với một tốc độ quét nhanh. Nếu có quá nhiều cảnh báo từ cùng một nguồn trong một khoảng thời gian ngắn, chỉ cảnh báo đầu tiên sẽ được gửi. Tập luật này hữu ích trong việc phát hiện và ngăn chặn các hoạt động quét mạng bằng cách theo dõi các gói tin có cờ SYN.

+ Với luật trên Snort sẽ theo dõi các gói tin với cờ SYN cùng xuất phát từ một nguồn gửi, đếm 100 gói trong 2 giây thì sẽ cảnh báo.

+ Tại máy Snort sử dụng lệnh khởi tạo chương trình:

```
root@anhduy:~# sudo snort -A console -i ens33 -u snort -g snort -c /etc/snort/snort.conf
```

Bước 3. Kết quả phát hiện tấn công

Sử dụng lệnh tail như kịch bản 1 xem trực tiếp sự kiện phát hiện tại máy Snort:

```
        Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
        Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
        Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
        Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
        Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
        Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
        Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
        Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
        Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
        Preprocessor Object: appid  Version 1.1  <Build 5>
        Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
        Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
        Preprocessor Object: SF_POP  Version 1.0  <Build 1>
        Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Commencing packet processing (pid=3241)
06/23-11:45:33.495114  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.2.10.100:37339 -> 10.2.10.50:705
06/23-11:45:33.505032  [**] [1:10000002:1] SYN scan attack [**] [Classification: Detection of a Network Scan] [Priority: 3] {TCP} 10.2.10.100:37339 -> 10.2.10.50:6129
06/23-11:45:33.505034  [**] [1:10000002:1] SYN scan attack [**] [Classification: Detection of a Network Scan] [Priority: 3] {TCP} 10.2.10.100:37339 -> 10.2.10.50:5405
06/23-11:45:33.505035  [**] [1:10000002:1] SYN scan attack [**] [Classification: Detection of a Network Scan] [Priority: 3] {TCP} 10.2.10.100:37339 -> 10.2.10.50:32768
06/23-11:45:33.505204  [**] [1:10000002:1] SYN scan attack [**] [Classification: Detection of a Network Scan] [Priority: 3] {TCP} 10.2.10.100:37339 -> 10.2.10.50:1001
06/23-11:45:33.505205  [**] [1:10000002:1] SYN scan attack [**] [Classification: Detection of a Network Scan] [Priority: 3] {TCP} 10.2.10.100:37339 -> 10.2.10.50:512
06/23-11:45:33.505349  [**] [1:10000002:1] SYN scan attack [**] [Classification: Detection of a Network Scan] [Priority: 3] {TCP} 10.2.10.100:37339 -> 10.2.10.50:3013
06/23-11:45:33.505350  [**] [1:10000002:1] SYN scan attack [**] [Classification: Detection of a Network Scan] [Priority: 3] {TCP} 10.2.10.100:37339 -> 10.2.10.50:902
06/23-11:45:33.505351  [**] [1:10000002:1] SYN scan attack [**] [Classification: Detection of a Network Scan] [Priority: 3] {TCP} 10.2.10.100:37339 -> 10.2.10.50:5903
06/23-11:45:33.505579  [**] [1:10000002:1] SYN scan attack [**] [Classification: Detection of a Network Scan] [Priority: 3] {TCP} 10.2.10.100:37339 -> 10.2.10.50:3880
06/23-11:45:33.505580  [**] [1:10000002:1] SYN scan attack [**] [Classification: Detection of a Network Scan] [Priority: 3] {TCP} 10.2.10.100:37339 -> 10.2.10.50:3914
06/23-11:45:33.505580  [**] [1:10000002:1] SYN scan attack [**] [Classification: Detection of a Network Scan] [Priority: 3] {TCP} 10.2.10.100:37339 -> 10.2.10.50:1076
06/23-11:45:33.505581  [**] [1:10000002:1] SYN scan attack [**] [Classification: Detection of a Network Scan] [Priority: 3] {TCP} 10.2.10.100:37339 -> 10.2.10.50:50389
```

Với hiển thị này ta biết được một máy nào đó có địa chỉ IP 10.2.10.100 (IP của máy kali) đang gửi gói SYN tới 10.0.0.50 (IP máy window server 2012) với cổng đích là 53. Ngoài ra giao diện còn hiển thị rất nhiều gói tin với các cổng khác. Do vậy sử dụng Snort ta phát hiện được đang có cuộc tấn công dò quét.

Bước 4 Đánh giá :

+ Tập luật này được sử dụng để phát hiện các tấn công dò quét mạng bằng cách kiểm tra gói tin TCP với cờ SYN và theo dõi số lượng gói tin từ cùng một nguồn trong khoảng thời gian 2 giây. Nếu số lượng gói tin vượt quá 100 gói, tập luật sẽ phát hiện và kích hoạt cảnh báo "SYN scan attack".

+ Ưu điểm: Tập luật này sử dụng một bộ lọc phát hiện với ngưỡng (threshold) để theo dõi số lượng gói tin từ cùng một nguồn trong 2 giây. Nếu vượt quá 100 gói tin, tập luật sẽ kích hoạt. Điều này có thể giúp giảm thiểu số lượng cảnh báo giả.

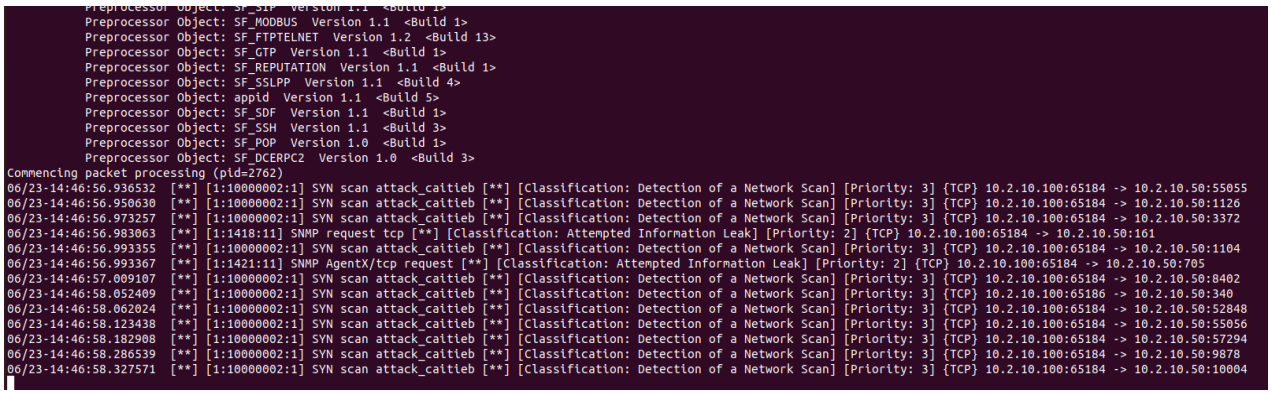
+ Nhược điểm: Tập luật này không chỉ định rõ thông điệp cảnh báo, không cho phép điều chỉnh ngưỡng phát hiện cụ thể.

Bước 5. Cải tiến tập luật

Để cải tiến tập luật và tối ưu thời gian và hiệu suất, có thể sử dụng tùy chọn "threshold" thay vì "detection_filter"

alert tcp any any -> $HOME_NET any (msg:"SYN scan attack"; flags:S; threshold: type threshold, track by_src, count 100, seconds 2; classtype:network-scan; sid:10000002; rev:1;)

Tập luật cải tiến sử dụng "threshold" có thể được coi là tốt hơn trong việc kiểm soát thời gian và hiệu suất. Nó giảm thiểu cảnh báo giả bằng cách áp dụng ngưỡng cho số lượng gói tin từ cùng một nguồn trong một khoảng thời gian nhất định. Điều này giúp tăng hiệu suất hệ thống IDS/IPS của bạn và giảm thời gian xử lý cảnh báo không cần thiết.

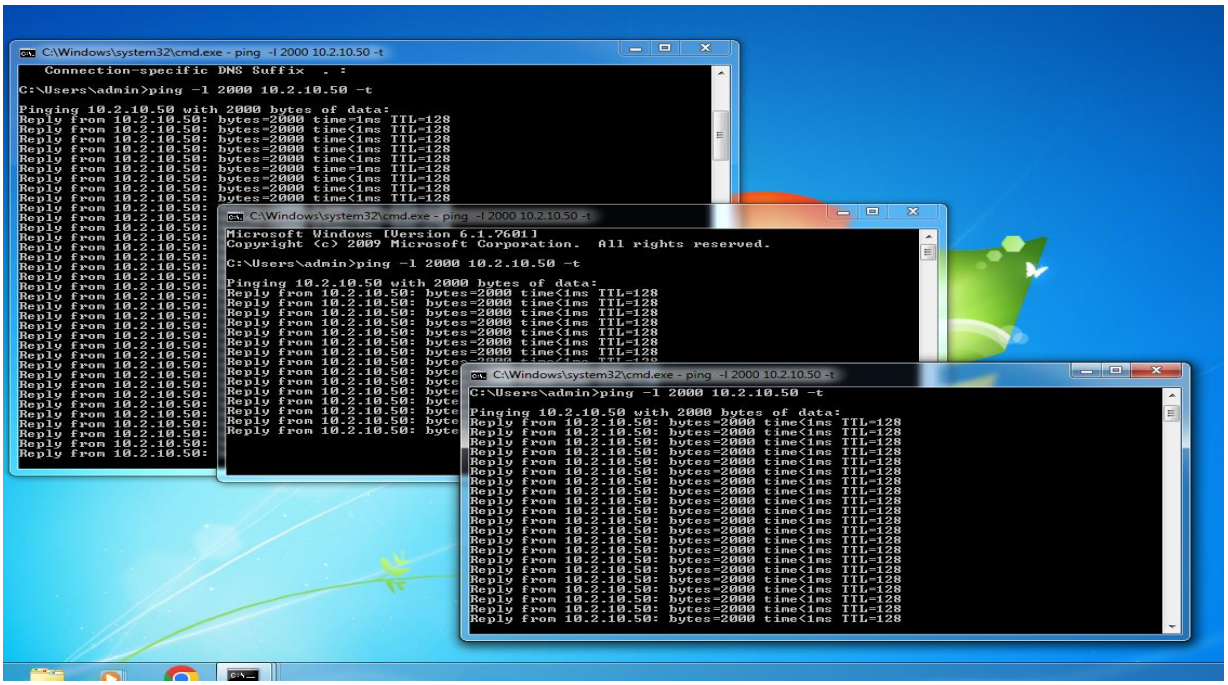


3.2.2 Phát hiện tấn công từ chối dịch vụ DoS

Bước 1. Tấn công

Tại máy tính Windows 7, sử dụng chương trình dòng lệnh CMD để ping với số lượng lớn các gói tin ICMP có kích thước lớn tới máy chủ Windows Server. Sử dụng lệnh

```
C:\Users\admin>ping -l 2000 10.2.10.50 -t
```



Chú ý: Đây là cách tấn công thử nghiệm để kiểm tra luật phát hiện của Snort, gây ảnh hưởng rất ít tới máy chủ.

Bước 2. Thiết lập tập luật phát hiện cho Snort

+ Sử dụng trình soạn thảo văn bản nano hoặc vi thêm vào luật phát hiện cho Snort trong tệp tin: /etc/snort/rules/local.rules

alert icmp any any -> $HOME_NET any (msg:"Dos Attack"; itype:8; dsize:>1000; detection_filter:track by_src, count 1000, seconds 20; classtype:denial-of-service; sid:10000003; rev:003;)

− alert: Loại tập luật, thông báo khi có sự khớp.

− icmp any any -> $HOME_NET any: Quy định các thông điệp ICMP đi từ bất kỳ địa chỉ nguồn và bất kỳ cổng đến địa chỉ mạng nội bộ (được xác định bởi biến

$HOME_NET) trên bất kỳ cổng đích nào.

− msg:"DoS Attack": Một thông điệp mô tả tấn công được phát hiện.

− itype:8: Điều kiện cho phép tập luật phù hợp với các gói tin ICMP với loại (type) là 8. Type 8 trong giao thức ICMP tương ứng với gói tin ICMP Echo Request, còn được biết đến là gói tin Ping.

− dsize:>1000: Điều kiện để gói tin ICMP phù hợp là có kích thước dữ liệu (dsize) lớn hơn 1000 byte. Điều này nhằm phát hiện các gói tin ICMP với kích thước không hợp lệ hoặc quá lớn.

− detection_filter:track by_src, count 10, seconds 20: Một bộ lọc phát hiện để giới hạn số lượng cảnh báo. Nếu có 10 hoặc nhiều hơn cảnh báo trong vòng 20 giây từ cùng một nguồn, chỉ cảnh báo đầu tiên sẽ được gửi.

− classtype:denial-of-service: Loại phân loại tấn công, chỉ định rằng tấn công liên quan đến chối dịch vụ.

− sid:10000003: Số ID duy nhất cho tập luật, sử dụng để xác định một cách duy nhất tập luật trong các cơ sở dữ liệu và nhật ký.

− rev:1: Số phiên bản của tập luật.

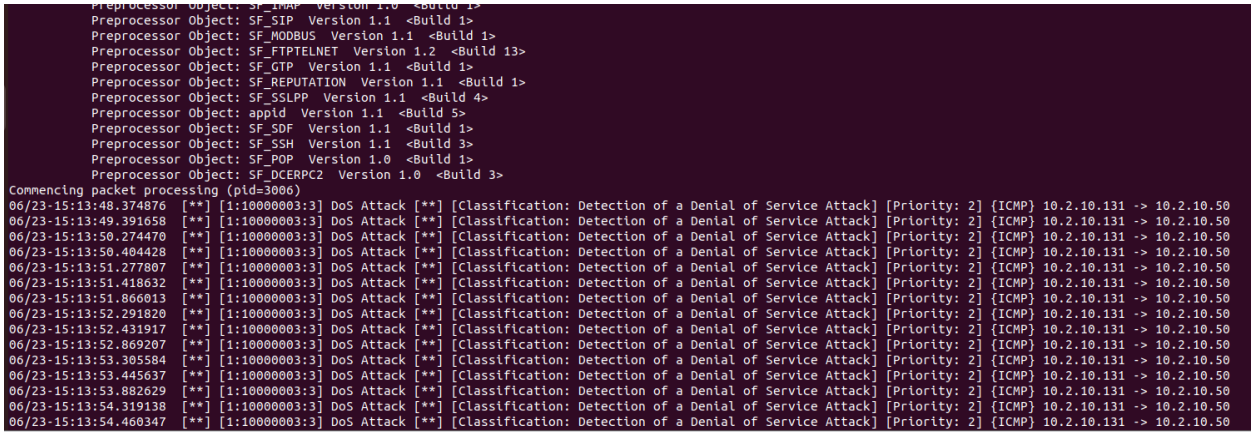
+ Tập luật trên sẽ phát hiện các gói tin ICMP Echo Request (Ping) có kích thước dữ liệu lớn hơn 1000 byte, có nguồn từ cùng một địa chỉ IP trong vòng 20 giây. Nếu điều kiện này được đáp ứng, Snort sẽ tạo ra một cảnh báo cho tấn công "DoS Attack".

+ Tại máy Snort sử dụng lệnh khởi tạo chương trình lắng nghe:

```
root@anhduy:~# sudo snort -A console -i ens33 -u snort -g snort -c /etc/snort/snort.conf
```

Bước 3. Phát hiện tấn công

Tại máy Snort sử dụng cửa sổ dòng lệnh và chạy lệnh tail như trên: Kết quả:



Với cảnh báo này người quản trị biết được rằng đang có cuộc tấn công dạng từ chối dịch vụ sử dụng giao thức ICMP có nguồn xuất phát từ máy có địa chỉ IP 10.2.10.131(win10) tới máy đích 10.2.10.50.(winserver 2012)

Bước 4 . Cải tiến tập luật

+ Tập luật mới sau khi cải tiến là:

alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"DoS Attack_CaiTien"; ttl: >64; itype:8; icode:0; dsize:>1000; detection_filter:track by_src, count 10, seconds 60; reference:arachnids,246; classtype:denial-of-service; sid:499; rev:5;)

+ Đánh giá : Tập luật mới đã được cải tiến so với tập luật cũ trong một số khía cạnh quan trọng

– Chính xác: Tập luật mới sử dụng các điều kiện kiểm tra bổ sung như TTL, type/code của gói tin ICMP để loại bỏ các gói tin không liên quan. Điều này giúp tăng độ chính xác của tập luật và giảm khả năng báo động giả mạo.

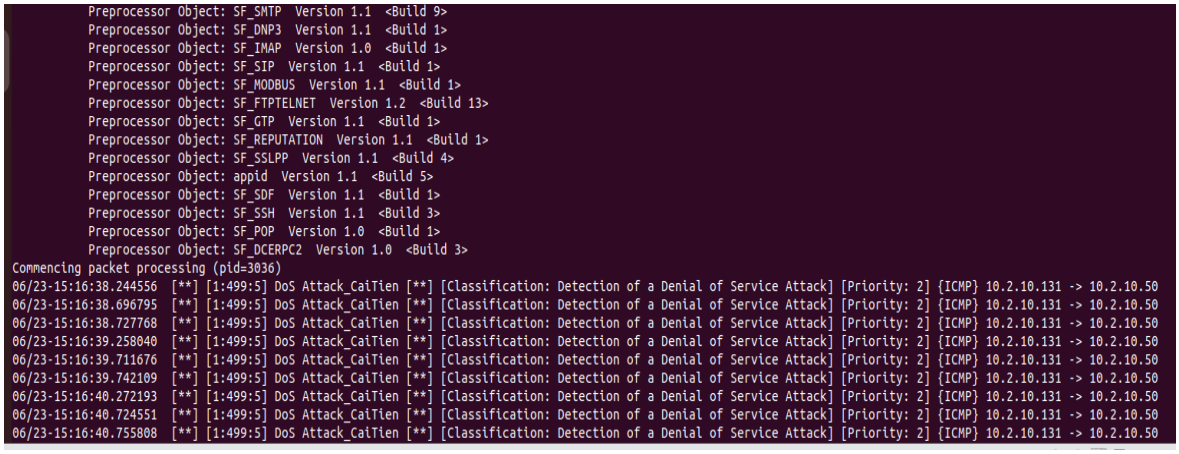
– Hiệu suất: Tập luật mới sử dụng detection_filter với một ngưỡng đếm gói tin thấp hơn và thời gian theo dõi lâu hơn. Điều này giúp giảm khả năng quá tải hệ thống và tăng hiệu suất chung của tập luật.

– Tối ưu hóa: Tập luật mới sử dụng các điều kiện kiểm tra bổ sung để loại bỏ các gói tin không liên quan ngay từ đầu. Điều này giúp giảm khối lượng dữ liệu phải xử lý và tăng tốc độ thực hiện tập luật.

Tập luật mới này bổ sung một số điều kiện bổ sung như TTL, mã và tham chiếu, trong khi giữ nguyên các điều kiện như kích thước dữ liệu và bộ lọc phát hiện. Tập luật này có thể cải thiện khả năng phát hiện tấn công PING OF DEATH bằng cách áp dụng các điều kiện nâng cao để loại bỏ các gói tin không liên quan. Tuy nhiên,

cần lưu ý rằng hiệu quả của tập luật phụ thuộc vào môi trường mạng cụ thể và các yếu tố khác như kiến trúc mạng và loại tấn công được mục tiêu.

Tóm lại, tập luật mới đã cải tiến đáng kể so với tập luật cũ trong việc tăng độ chính xác và hiệu suất

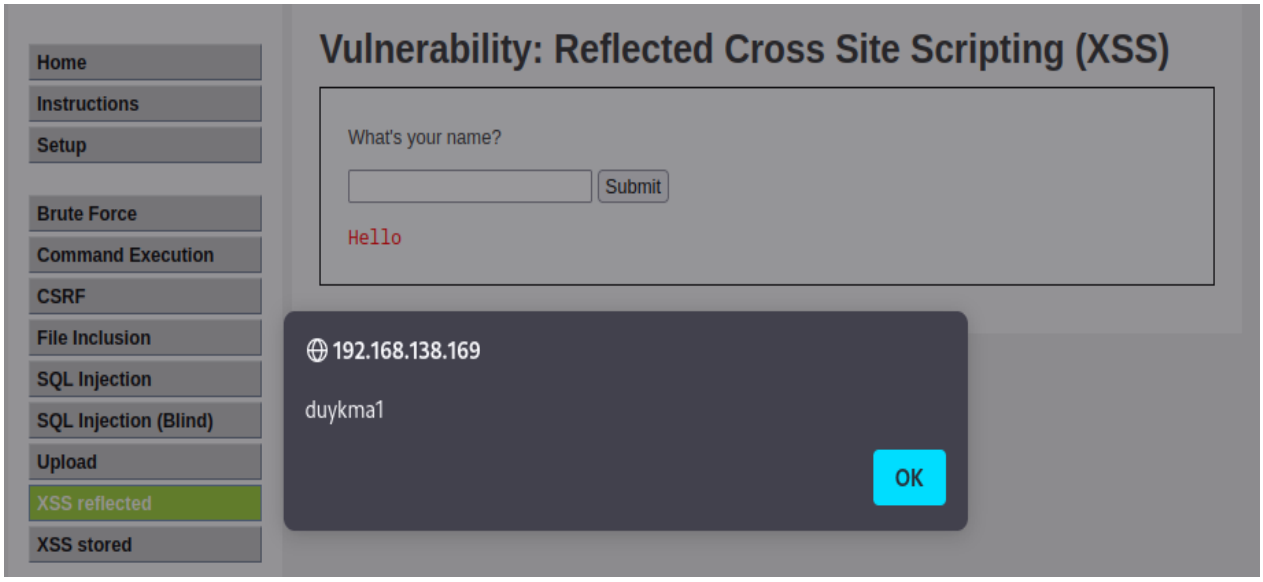


3.2.3. Phát hiện tấn công XSS

Bước 1. Tấn công

+ Vô trình duyệt ,truy cập vào trang DVWA ,tài khoản login mặc định là: (admin ,password)

+ Chèn dữ liệu “<script>alert("duykma")</script>” vào Vulnerability: Reflected Cross Site Scripting (XSS). Thu được kết quả :



+ Với kết quả trên ,thì đồng nghĩa với việc ,thực hiện XSS thành công

Bước 2 : Thiết lập tập luật phát hiện cho snort

+ Sử dụng trình soạn thảo văn bản nano hoặc vi thêm vào luật phát hiện cho Snort trong tệp tin: /etc/nsm/rules/local.rules

alert tcp $HOME_NET any -> $HOME_NET $HTTP_PORTS (msg:"Dectected XSS attack !."; flow:established,to_server; content:"GET"; http_method; content:"script"; http_uri; classtype:attempted-recon; sid:10000009; rev:2;)

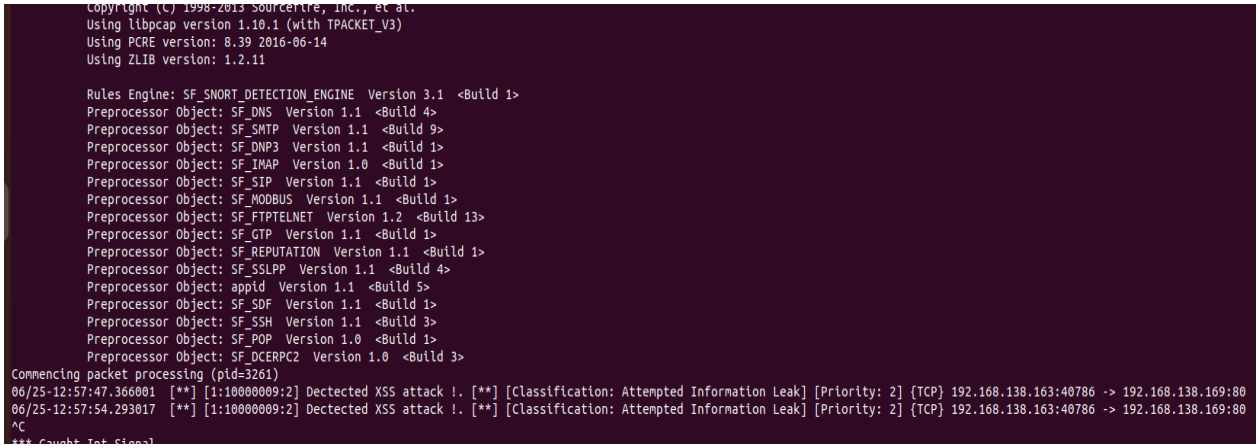
- alert: Thông báo khi có sự kiện xảy ra
- tcp: Kiểu giao thức, trong trường hợp này là TCP
- $HOME_NET any: Địa chỉ mạng của máy chạy Snort làm nơi lắng nghe
- $HOME_NET $HTTP_PORTS: Địa chỉ mạng nội bộ và cổng HTTP để kiểm tra nội dung
- msg:"Dectected XSS attack !.": Nội dung thông báo hiển thị khi phát hiện tấn công XSS
- flow:established,to_server: Chỉ phát hiện các luồng TCP đã được thiết lập và có gói tin được gửi đến server
- content:"GET"; http_method: Kiểm tra xem nội dung có chứa phương thức HTTP GET không
- content:"script"; http_uri: Kiểm tra xem URI HTTP có chứa chuỗi "script" không
- classtype:attempted-recon: Phân loại tấn công là attempted-recon
- sid:10000009: Số ID của luật
- rev:2: Số phiên bản của luật

+ Tập luật này được thiết kế để phát hiện cuộc tấn công XSS (Cross-Site Scripting) bằng cách kiểm tra phương thức HTTP và URI của gói tin. Nó có thể phát hiện những yêu cầu GET có chứa từ "script" trong URI, cho biết có khả năng xảy ra một cuộc tấn công XSS.

+ Tuy nhiên, tập luật này có một số hạn chế và có thể không phát hiện được tất cả các cuộc tấn công XSS. Nó chỉ tập trung vào phương thức GET và chứa từ "script" trong URI, trong khi các cuộc tấn công XSS có thể sử dụng các phương thức và cấu trúc khác.

Bước 3 : Phát hiện tấn công

Vào Snort giám sát:



Dựa vào luật Snort đã thiết lập , thì đã phát hiện tấn công XSS

Bước 4 . Cải tiến tập luật

+ Tập luật mới sau khi cải tiến :

alert tcp $HOME_NET any -> $HOME_NET $HTTP_PORTS (msg:"Detected XSS attack"; flow:established,to_server; content:"GET"; http_method; pcre:"/\<script\>/i"; content:"script"; http_uri; classtype:attempted-recon; sid:10000009; rev:3;)

+ Đánh giá : Tập luật mới sử dụng pcre:"/\<script\>/i" để kiểm tra sự xuất hiện của từ "<script>" trong nội dung HTTP. Biểu thức chính quy /\<script\>/i sẽ so khớp không phân biệt chữ hoa/chữ thường với từ "<script>".

– Tập luật mới sử dụng biểu thức chính quy để so khớp nhanh chóng và chính xác với từ "<script>" trong nội dung HTTP.

– Sử dụng biểu thức chính quy có thể giúp tăng tính linh hoạt và khả năng phát hiện cuộc tấn công XSS.

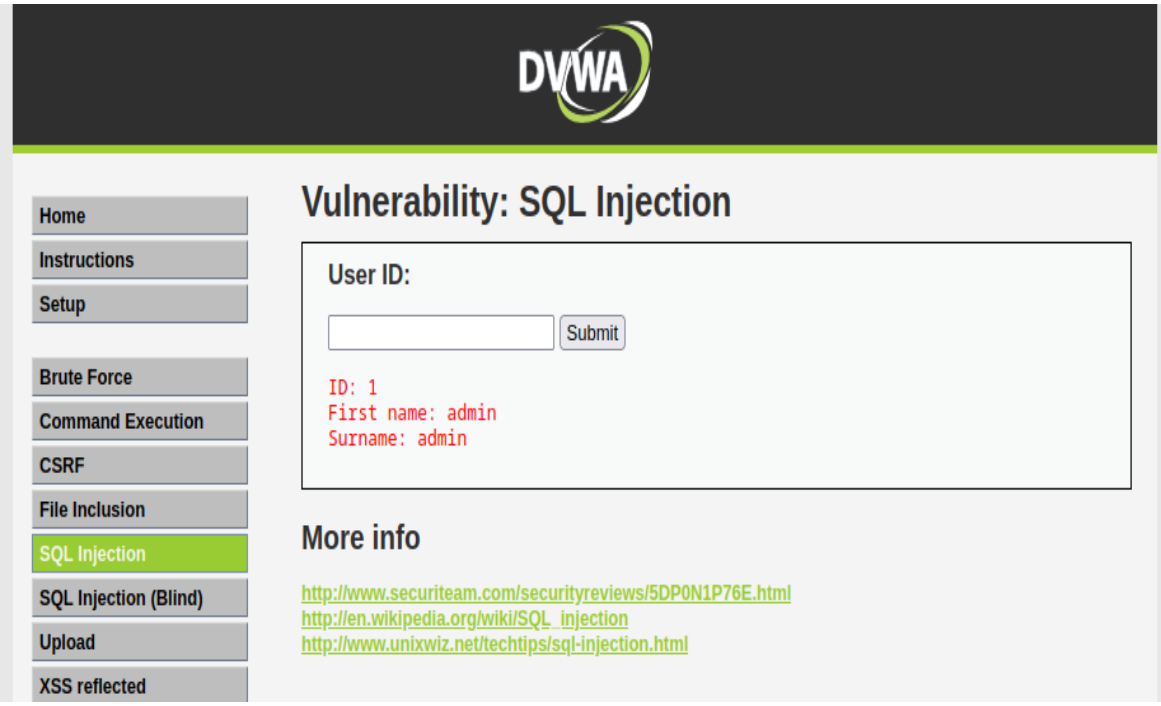
– Hiệu suất và tính năng của tập luật mới sẽ phụ thuộc vào môi trường hệ thống cụ thể và cách tấn công XSS được thực hiện.

3.2.4. Phát hiện tấn công SQL Injection

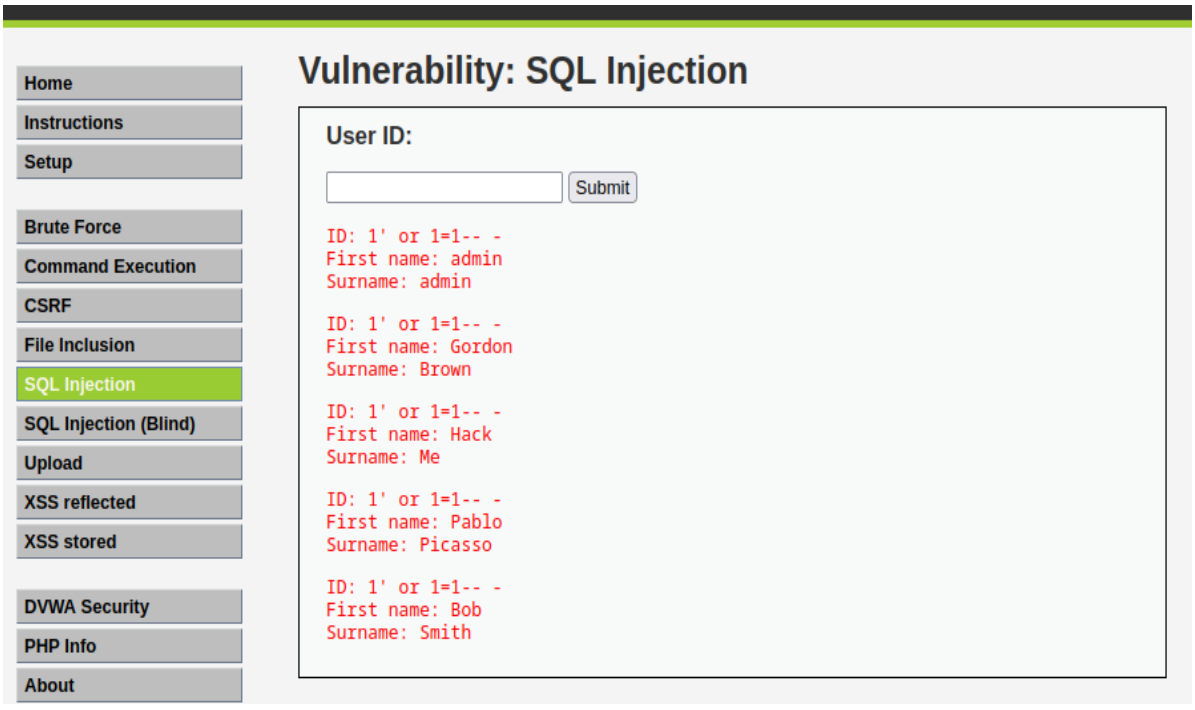
Bước 1. Tấn công

+ Vào trình duy và truy cập vào DVWA ,Chọn kiểm thử SQL Injection

+ Tại trường User ID : nhập “1” thì có kết quả như sau :



+ Bây giờ chèn câu lệnh SQL : " ' or 1=1 -- - " vô trường User Id ,kết quả trả về



Bước 2: Thiết lập tập luật phát hiện cho snort

Sử dụng trình soạn thảo văn bản nano hoặc vi thêm vào luật phát hiện cho Snort trong tệp tin: /etc/nsm/rules/local.rules

alert tcp $HOME_NET any -> $HOME_NET $HTTP_PORTS (msg:"Dectected SQL Injection Attack ."; flow:established,to_server;

content:"GET"; http_method; content:"or"; http_uri; classtype:attempted-recon; sid:10000010; rev:1;)

alert tcp $HOME_NET any -> $HOME_NET $HTTP_PORTS (msg:"Dectected SQL Injection Attack ."; flow:established,to_server; content:"GET"; http_method; content:"select"; http_uri; classtype:attempted-recon; sid:10000011; rev:1;)

+ Cả hai tập luật đều có mục tiêu phát hiện các cuộc tấn công SQL Injection thông qua phương thức HTTP GET.

+ Các nội dung kiểm tra (content rules) trong tập luật đều tìm kiếm chuỗi cụ thể trong phương thức HTTP và URI.

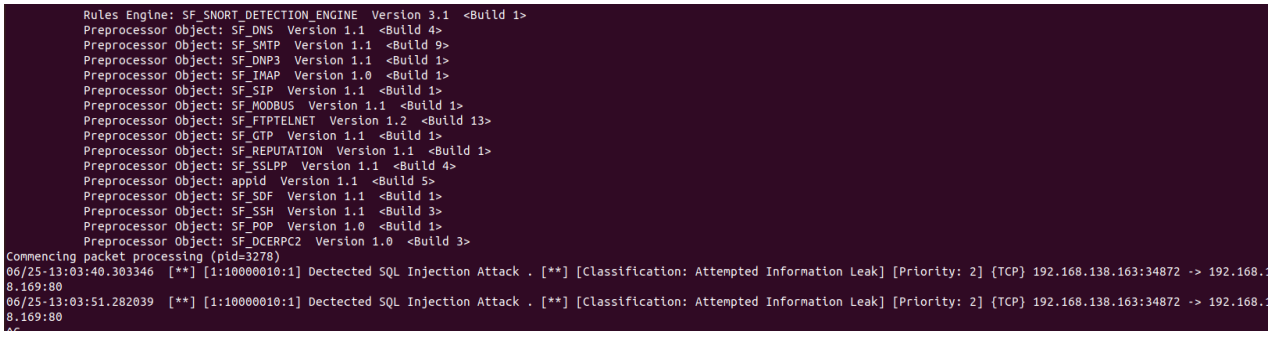
+ Tuy nhiên, các tập luật này có thể bị hạn chế bởi việc chỉ kiểm tra các chuỗi cụ thể như "or" và "select", trong khi các cuộc tấn công SQL Injection có thể sử dụng nhiều kỹ thuật và từ khóa khác nhau.

+ Để cải thiện tập luật, bạn có thể xem xét sử dụng các kỹ thuật phức tạp hơn như sử dụng biểu thức chính quy (PCRE) để tìm kiếm các mẫu SQL Injection phổ biến.

+ Việc đánh giá tốc độ và hiệu suất của tập luật sẽ phụ thuộc vào môi trường cụ thể và tài nguyên hệ thống. Thử nghiệm và tinh chỉnh trong môi trường thực tế là cần thiết để đảm bảo tối ưu hóa hiệu suất và tính khả dụng của tập luật.

Bước 3 . Phát hiện tấn công

Vào Snort để giám sát



Dựa vào luật Snort đã thiết lập , thì đã phát hiện tấn công SQL Injection

Bước 4 . Cải tiến tập luật

Tập luật sau khi cải tiến:

alert tcp $HOME_NET any -> $HOME_NET $HTTP_PORTS (msg:"Detected SQL Injection Attack_Caitien"; flow:established,to_server; content:"GET"; http_method;content:"or";pcre:"/\b(union|select|insert|delete|update|drop|alter)\b/i"; http_uri;classtype:attempted-recon; sid:10000009; rev:2;)

alert tcp $HOME_NET any -> $HOME_NET $HTTP_PORTS (msg:"Detected SQL Injection Attack_Caitien"; flow:established,to_server; content:"GET"; http_method;

content:"select";pcre:"/\b(union|select|insert|delete|update|drop|alter)\b/i";http_uri;classtype:attempted-recon; sid:10000009; rev:2;)

+ Tập luật mới đã được cải tiến bằng cách sử dụng biểu thức chính quy để tìm kiếm nhiều từ khóa SQL Injection phổ biến hơn. Điều này cải thiện khả năng phát hiện các cuộc tấn công SQL Injection so với tập luật cũ chỉ tìm kiếm một từ khóa duy nhất.

+ Tuy nhiên, tập luật cải tiến vẫn có một số hạn chế:

- Tập luật này chỉ xác định cuộc tấn công SQL Injection dựa trên việc tìm kiếm các từ khóa trong URI (đường dẫn) HTTP. Điều này có nghĩa là nó có thể bỏ qua các cuộc tấn công SQL Injection sử dụng các phương thức HTTP khác hoặc không sử dụng URI để truyền tham số.

- Tập luật này cũng có thể tạo ra các thông báo giả-positives (cảnh báo sai) nếu các từ khóa SQL Injection xuất hiện trong một ngữ cảnh không liên quan hoặc là một phần của dữ liệu hợp lệ.

3.3. Đánh giá kết quả thực nghiệm

Đánh giá thực nghiệm là một bước quan trọng trong quá trình triển khai giải pháp phát hiện xâm nhập Snort, và để đảm bảo rằng giải pháp có khả năng phát hiện các tấn công mạng và đáp ứng được các yêu cầu bảo mật của một hệ thống mạng thực tế. Qua các kịch bản thực nghiệm trên , cho ta thấy được sức mạng của snort là vô cùng lớn , nếu biết tận dụng nó . Việc cải tiến các tập luật sẽ giúp hệ thống phát hiện kịp thời các cuộc tấn công hay lỗ hổng mà các hacker hay người dùng tạo ra , nhanh chóng được xử lý và giảm thiểu khả năng rủi ro của hệ thông .

Tổng kết lại, các kịch bản phát hiện tấn công đã được triển khai và kiểm tra trên môi trường thực tế với kết quả tốt. Snort đã phát hiện và cảnh báo cho người quản trị hệ thống về các hoạt động tấn công từ các địa chỉ IP khác nhau. Điều này cho

thấy khả năng của Snort trong việc phát hiện các hoạt động tấn công và đưa ra giải pháp kịp thời, giúp người quản trị hệ thống bảo vệ mạng máy tính trước những mối đe dọa từ bên ngoài.

KẾT LUẬN

Sau thời gian nghiên cứu đề tài "XÂY DỰNG VÀ TRIỂN KHAI HỆ THỐNG PHÒNG CHỐNG TẤN CÔNG IDS/IPS DỰA TRÊN CÔNG CỤ MÃ NGUỒN MỞ" và đi sâu hơn về phần cải tiến luật giúp tăng tốc độ xử lý cũng như hiệu năng của nó mang lại nhóm chúng em đã tìm hiểu được một số vấn đề sau đây:

+ Bản chất và cách thức hoạt động của IDS, cách thức triển khai IDS trong hệ thống mạng

+ Bản chất và cách thức hoạt động của IPS, cách thức triển khai IPS trong hệ thống mạng.

+ Nguyên lý hoạt động của các thành phần trong Snort cũng như hoạt động của toàn bộ hệ thống Snort.

+ Tập rule của Snort, cách xây dựng một rule trong Snort, sử dụng và viết thêm hay cải tiến một số rule. Qua đó người dùng Snort có thể xây dựng các rule phù hợp với nhu cầu của hệ thống mạng hiện tại

Hạn chế còn trong đề tài

+ Trong quá trình nghiên cứu, nhóm chúng em đã gặp không ít khó khăn trong việc triển khai hệ thống Snort IDS-IPS

+ Không có thiết bị và phải demo trên máy ảo. Do đó chúng em chỉ trình bày các demo đơn giản để hiểu rõ về cách thức hoạt động của Snort IDS-IPS

+ Không có đủ tài nguyên để triển khai các cuộc tấn công xâm nhập thực tế và kiểm tra hiệu quả của Snort.

+ Nhiều tính năng của Snort phức tạp và cần có kiến thức chuyên sâu để cấu hình và sử dụng.

+ Vẫn còn một số thách thức trong việc phát hiện các cuộc tấn công xâm nhập mới và phức tạp.

Các hướng mở rộng, phát triển cho đề tài này:

+ Có thể xây dựng hệ thống honeypot đánh lạc hướng các cuộc tấn công, đây là một giải pháp hay nhưng khó khăn để thực thi, đòi hỏi nhiều kiến thức và kĩ năng.

+ Kết hợp xây dựng hệ thống IDS/IPS dựa trên sự kết hợp giữa phần cứng (các thiết bị IDS/IPS) và phần mềm.

+ Tìm hiểu và nghiên cứu sâu hơn về các giải pháp phát hiện xâm nhập khác như Suricata, Zeek, OSSEC, Wazuh để so sánh và đánh giá.

TÀI LIỆU THAM KHẢO

[1]. https://fptcloud.com/ids-la-gi/

[2].https://viblo.asia/p/network-tim-hieu-co-che-cach-hoat-dong-cua-ids-phan-1-ojaqG0NeREKw

[3].https://viblo.asia/p/network-tim-hieu-co-che-cach-hoat-dong-cua-ids-phan-1-ojaqG0NeREKw

[4].https://itnavi.com.vn/blog/he-thong-phat-hien-xam-nhap-ids

[5]https://quantrimang.com/cong-nghe/he-thong-phat-hien-xam-pham-ids-phan-1-37334

[6]https://viblo.asia/p/network-tim-hieu-co-che-cach-hoat-dong-cua-ids-phan-2-pDljMbe5RVZn

[7] https://antoanthongtin.gov.vn/gp-attm/xac-dinh-tap-luat-cho-ids-dua-tren-cac-dau-hieu-100146

[8]https://www.forum.vnpro.org/forum/ccnp-security-%C2%AE-ccsp/firewall-vpn-snaf-snaa/13037-c%C3%A1c-lu%E1%BA%ADt-c%E1%BB%A7a-snort

[9]https://quantrimang.com/cong-nghe/su-dung-snort-phat-hien-mot-so-kieu-tan-cong-pho-bien-hien-nay-vao-cac-ung-dung-web-45727/

[10]https://whitehat.vn/threads/cach-viet-rule-phat-hien-shellcode-cua-snort.9637/

[11] https://github.com/maj0rmil4d/snort-ddos-mitigation/blob/main/dos.rules

[12]https://www.snort.org/rule-docs/1-355

[13]https://github.com/eldondev/Snort/blob/master/rules/ftp.rules