MỤC LỤC

LỜI NÓI ĐẦU

An ninh thông tin nói chung và an ninh mạng nói riêng đang là vấn đề được quan tâm không chỉ ở Việt Nam mà trên toàn thế giới. Cùng với sự phát triển nhanh chóng của mạng Internet, việc đảm bảo an ninh cho các hệ thống thông tin càng trở nên cấp thiết hơn bao giờ hết.

Trong lĩnh vực an ninh mạng, phát hiện và phòng chống tấn công xâm nhập cho các mạng máy tính là một đề tài hay, thu hút được sự chú ý của nhiều nhà nghiên cứu với nhiều hướng nghiên cứu khác nhau. Trong xu hướng đó, đồ án thực tập chuyên ngành này chúng em mong muốn có thể tìm hiểu, nghiên cứu về phát hiện và phòng chống xâm nhập mạng với mục đích nắm bắt được các giải pháp, các kỹ thuật tiên tiến để chuẩn bị tốt cho hành trang của mình sau khi ra trường. Mặc dù đã cố gắng hết sức nhưng do kiến thức và khã năng nhìn nhận vấn đề còn hạn chế nên bài làm không tránh khỏi thiếu sót, rất mong được sự quan tâm và góp ý thêm của thầy cô và tất cả các bạn.

Để có thể hoàn thành đươc đồ án này , chúng em xin gửi lời cảm ơn sâu sắc nhất tới thầy Nguyễn Đào Trường đã nhiệt tình hướng dẫn, chỉ bảo và cung cấp cho chúng em nhiều kiến thức rất bổ ích trong suốt quá trình làm đồ án. Nhờ sự giúp đỡ tận tâm của thầy, chúng em mới có thể hoàn thành được đồ án này.

Một lần nữa xin cảm ơn thầy rất nhiều !

CHƯƠNG I : TỔNG QUAN VỀ IDS/IPS

1.1 Giới thiệu về IDS/IPS

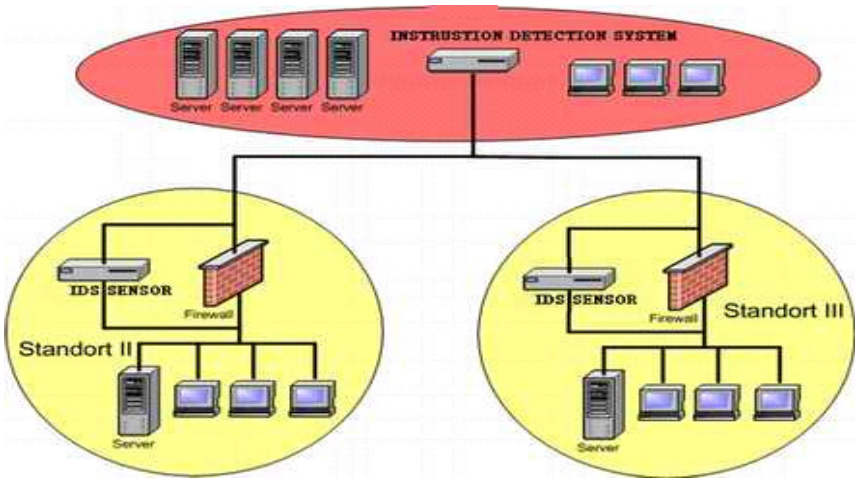
1.1.1 Định nghĩa

Hệ thống phát hiện xâm nhập (IDS) là hệ thống có nhiệm vụ theo dõi, phát hiện và (có thể) ngăn cản sự xâm nhập, cũng như các hành vi khai thác trái phép tài nguyên của hệ thống được bảo vệ mà có thể dẫn đến việc làm tổn hại đến tính bảo mật, tính toàn vẹn và tính sẵn sàng của hệ thống.

Hệ thống IDS sẽ thu thập thông tin từ rất nhiều nguồn trong hệ thống được bảo vệ sau đó tiến hành phân tích những thông tin đó theo các cách khác nhau để phát hiện những xâm nhập trái phép.

Khi một hệ thống IDS có khả năng ngăn chặn các nguy cơ xâm nhập mà nó phát hiện được thì nó được gọi là một hệ thống phòng chống xâm nhập hay IPS.

Hình sau minh hoạ các vị trí thường cài đặt IDS trong mạng :



Hình : Các vị trí đặt IDS trong mạng

1.1.2 Sự khác nhau giữa IDS và IPS

Có thể nhận thấy sự khác biệt giữa hai khái niệm ngay ở tên gọi: "phát hiện" và "ngăn chặn". Các hệ thống IDS được thiết kế với mục đích chủ yếu là phát hiện và cảnh báo các nguy cơ xâm nhập đối với mạng máy tính nó đang bảo vệ trong khi đó, một hệ thống IPS ngoài khả năng phát hiện còn có thể tự hành động chống lại các nguy cơ theo các quy định được người quản trị thiết lập sẵn.

Tuy vậy, sự khác biệt này trên thực tế không thật sự rõ ràng. Một số hệ thống IDS được thiết kế với khả năng ngăn chặn như một chức năng tùy chọn. Trong khi đó một số hệ thống IPS lại không mang đầy đủ chức năng của một hệ thống phòng chống theo đúng nghĩa.

Một câu hỏi được đặt ra là lựa chọn giải pháp nào, IDS hay IPS? Câu trả lời tùy thuộc vào quy mô, tính chất của từng mạng máy tính cụ thể cũng như chính sách an ninh của những người quản trị mạng. Trong trường hợp các mạng có quy mô nhỏ, với một máy chủ an ninh, thì giải pháp IPS thường được cân nhắc

nhiều hơn do tính chất kết hợp giữa phát hiện, cảnh báo và ngăn chặn của nó. Tuy nhiên với các mạng lơn hơn thì chức năng ngăn chặn thường được giao phó cho một sản phẩm chuyên dụng như một firewall chẳng hạn. Khi đó, hệ thống cảnh báo sẽ chỉ cần theo dõi, phát hiện và gửi các cảnh báo đến một hệ thống ngăn chặn khác. Sự phân chia trách nhiệm này sẽ làm cho việc đảm bảo an ninh cho mạng trở nên linh động và hiệu quả hơn.

1.2 phân loại IDS/IPS

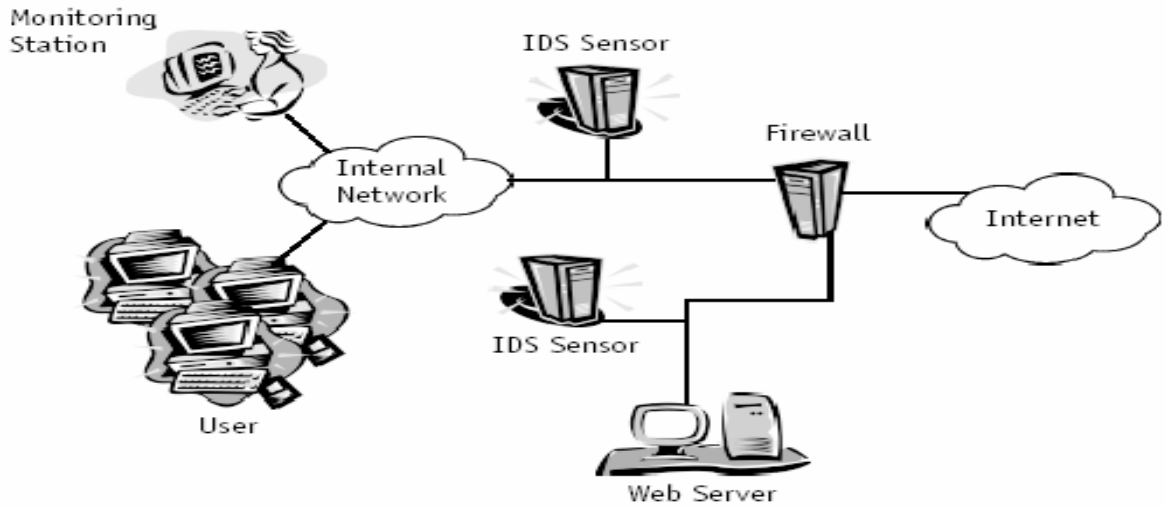
Cách thông thường nhất để phân loại các hệ thống IDS (cũng như IPS) là dựa vào đặc điểm của nguồn dữ liệu thu thập được. Trong trường hợp này, các hệ thống IDS được chia thành các loại sau:

- Host-based IDS (HIDS): Sử dụng dữ liệu kiểm tra từ một máy trạm đơn để phát hiện xâm nhập.
- Network-based IDS (NIDS): Sử dụng dữ liệu trên toàn bộ lưu thông mạng, cùng với dữ liệu kiểm tra từ một hoặc một vài máy trạm để phát hiện xâm nhập.

1.2.1 Network based IDS – NIDS

NIDS thường bao gồm có hai thành phần logic :

- Bộ cảm biến – Sensor : đặt tại một đoạn mạng, kiểm soát các cuộc lưu thông nghi ngờ trên đoạn mạng đó.
- Trạm quản lý : nhận các tín hiệu cảnh báo từ bộ cảm biến và thông báo cho một điều hành viên.



Hình I : Mô hình NIDS

Một NIDS truyền thống với hai bộ cảm biến trên các đoạn mạng khác nhau cùng giao tiếp với một trạm kiểm soát.

Ưu điểm

- Chi phí thấp : Do chỉ cần cài đặt NIDS ở những vị trí trọng yếu là có thể giám sát lưu lượng toàn mạng nên hệ thống không cần phải nạp các phần mềm và quản lý trên các máy toàn mạng.
- Phát hiện được các cuộc tấn công mà HIDS bỏ qua: Khác với HIDS, NIDS kiểm tra header của tất cả các gói tin vì thế nó không bỏ sót các dấu hiệu xuất phát từ đây. Ví dụ: nhiều cuộc tấn công DoS, TearDrop (phân nhỏ) chỉ bị phát hiện khi xem header của các gói tin lưu chuyển trên mạng.
- Khó xoá bỏ dấu vết (evidence): Các thông tin lưu trong log file có thể bị kẻ đột nhập sửa đổi để che dấu các hoạt động xâm nhập, trong tình huống này HIDS khó có đủ thông tin để hoạt động. NIDS sử dụng lưu thông hiện hành trên mạng để phát hiện xâm nhập. Vì thế, kẻ đột nhập không thể xoá bỏ được các dấu vết tấn công. Các thông tin bắt được không chỉ chứa cách thức tấn công mà cả thông tin hỗ trợ cho việc xác minh và buộc tội kẻ đột nhập.
- Phát hiện và đối phó kịp thời : NIDS phát hiện các cuộc tấn công ngay khi xảy ra, vì thế việc cảnh báo và đối phó có thể thực hiện được nhanh hơn. VD : Một hacker thực hiện tấn công DoS dựa trên TCP có thể bị NIDS phát hiện và ngăn chặn ngay bằng việc gửi yêu cầu TCP reset nhằm chấm dứt cuộc tấn công trước khi nó xâm nhập và phá vỡ máy bị hại.
- Có tính độc lập cao: Lỗi hệ thống không có ảnh hưởng đáng kể nào đối với công việc của các máy trên mạng. Chúng chạy trên một hệ thống chuyên dụng dễ dàng cài đặt; đơn thuần chỉ mở thiết bị ra, thực hiện một vài sự thay đổi cấu hình và cắm chúng vào trong mạng tại một vị trí cho phép nó kiểm soát các cuộc lưu thông nhạy cảm.

Nhược điểm

- Bị hạn chế với Switch: Nhiều lợi điểm của NIDS không phát huy được trong các mạng chuyển mạch hiện đại. Thiết bị switch chia mạng thành nhiều phần độc lập vì thế NIDS khó thu thập được thông tin trong toàn mạng. Do chỉ kiểm tra mạng trên đoạn mà nó trực tiếp kết nối tới, nó không thể phát hiện một cuộc tấn công xảy ra trên các đoạn mạng khác.

Vấn đề này dẫn tới yêu cầu tổ chức cần phải mua một lượng lớn các bộ cảm biến để có thể bao phủ hết toàn mạng gây tốn kém về chi phí cài đặt.

- Hạn chế về hiệu năng: NIDS sẽ gặp khó khăn khi phải xử lý tất cả các gói tin trên mạng rộng hoặc có mật độ lưu thông cao, dẫn đến không thể phát hiện các cuộc tấn công thực hiện vào lúc "cao điểm". Một số nhà sản xuất đã khắc phục bằng cách cứng hoá hoàn toàn IDS nhằm tăng cường tốc độ cho nó. Tuy nhiên, do phải đảm bảo về mặt tốc độ nên một số gói tin được bỏ qua có thể gây lỗ hổng cho tấn công xâm nhập.
- Tăng thông lượng mạng: Một hệ thống phát hiện xâm nhập có thể cần truyền một dung lượng dữ liệu lớn trở về hệ thống phân tích trung tâm, có nghĩa là một gói tin được kiểm soát sẽ sinh ra một lượng lớn tải phân tích. Để khắc phục người ta thường sử dụng các tiến trình giảm dữ liệu linh hoạt để giảm bớt số lượng các lưu thông được truyền tải. Họ cũng thường thêm các chu trình tự ra các quyết định vào các bộ cảm biến và sử dụng các trạm trung tâm như một thiết bị hiển thị trạng thái hoặc trung tâm truyền thông hơn là thực hiện các phân tích thực tế. Điểm bất lợi là nó sẽ cung cấp rất ít thông tin liên quan cho các bộ cảm biến; bất kỳ bộ cảm biến nào sẽ không biết được việc một bộ cảm biến khác dò được một cuộc tấn công. Một hệ thống như vậy sẽ không thể dò được các cuộc tấn công hiệp đồng hoặc phức tạp.
- Một hệ thống NIDS thường gặp khó khăn trong việc xử lý các cuộc tấn công trong một phiên được mã hoá. Lỗi này càng trở nên trầm trọng khi nhiều công ty và tổ chức đang áp dụng mạng riêng ảo VPN.
- - Một số hệ thống NIDS cũng gặp khó khăn khi phát hiện các cuộc tấn công mạng từ các gói tin phân mảnh. Các gói tin định dạng sai này có thể làm cho NIDS hoạt động sai và đổ vỡ.
- 1.2.2 Host based IDS – HIDS

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-select user, như vậy những cố gắng liên tục để login vào account root có thể được coi là một cuộc tấn công.

Ưu điểm

- Xác định được kết quả của cuộc tấn công: Do HIDS sử dụng dữ liệu log lưu các sự kiện xảy ra, nó có thể biết được cuộc tấn công là thành công hay thất bại với độ chính xác cao hơn NIDS. Vì thế, HIDS có thể bổ sung thông tin tiếp theo khi cuộc tấn công được sớm phát hiện với NIDS.
- Giám sát được các hoạt động cụ thể của hệ thống: HIDS có thể giám sát các hoạt động mà NIDS không thể như: truy nhập file, thay đổi quyền, các hành động thực thi, truy nhập dịch vụ được phân quyền. Đồng thời nó cũng giám sát các hoạt động chỉ được thực hiện bởi người quản trị. Vì thế, hệ thống host-based IDS có thể là một công cụ cực mạnh để phân tích các cuộc tấn công có thể xảy ra do nó thường cung cấp nhiều thông tin chi tiết và chính xác hơn một hệ network-based IDS.
- Phát hiện các xâm nhập mà NIDS bỏ qua: chẳng hạn kẻ đột nhập sử dụng bàn phím xâm nhập vào một server sẽ không bị NIDS phát hiện.
- Thích nghi tốt với môi trường chuyển mạch, mã hoá: Việc chuyển mạch và mã hoá thực hiện trên mạng và do HIDS cài đặt trên máy nên nó không bị ảnh hưởng bởi hai kỹ thuật trên.
- Không yêu cầu thêm phần cứng: Được cài đặt trực tiếp lên hạ tầng mạng có sẵn (FTP Server, WebServer) nên HIDS không yêu cầu phải cài đặt thêm các phần cứng khác.

Nhược điểm

- Khó quản trị : các hệ thống host-based yêu cầu phải được cài đặt trên tất cả các thiết bị đặc biệt mà bạn muốn bảo vệ. Đây là một khối lượng công việc lớn để cấu hình, quản lí, cập nhật.
- Thông tin nguồn không an toàn: một vấn đề khác kết hợp với các hệ thống host-based là nó hướng đến việc tin vào nhật ký mặc định và năng lực kiểm soát của server. Các thông tin này có thể bị tấn công và đột nhập dẫn đến hệ thống hoạt đông sai, không phát hiện được xâm nhập.
- Hệ thống host-based tương đối đắt : nhiều tổ chức không có đủ nguồn tài chính để bảo vệ toàn bộ các đoạn mạng của mình sử dụng các hệ thống host-based. Những tổ chức đó phải rất thận trọng trong việc chọn các hệ thống nào để bảo vệ. Nó có thể để lại các lỗ hổng lớn trong mức độ bao phủ phát hiện xâm nhập. Ví dụ như một kẻ tấn công trên một hệ thống láng giềng không được bảo vệ có thể đánh hơi thấy các thông tin xác thực hoặc các tài liệu dễ bị xâm phạm khác trên mạng.

- Chiếm tài nguyên hệ thống : Do cài đặt trên các máy cần bảo vệ nên HIDS phải sử dụng các tài nguyên của hệ thống để hoạt động như: bộ vi xử lí, RAM, bộ nhớ ngoài.
-
- 1.3 Cơ chế hoạt động của hệ thống IDS/IPS

Có hai cách tiếp cận cơ bản đối với việc phát hiện và phòng chống xâm nhập là :

phát hiện sự lạm dụng (Misuse Detection Model): Hệ thống sẽ phát hiện các xâm nhập bằng cách tìm kiếm các hành động tương ứng với các kĩ thuật xâm nhập đã được biết đến (dựa trên các dấu hiệu - signatures) hoặc các điểm dễ bị tấn công của hệ thống.

phát hiện sự bất thường (Anomaly Detection Model): Hệ thống sẽ phát hiện các xâm nhập bằng cách tìm kiếm các hành động khác với hành vi thông thường của người dùng hay hệ thống.

- 1.3.1 phát hiện sự lạm dụng

Phát hiện sự lạm dụng là phát hiện những kẻ xâm nhập đang cố gắng đột nhập vào hệ thống mà sử dụng một số kỹ thuật đã biết. Nó liên quan đến việc mô tả đặc điểm các cách thức xâm nhập vào hệ thống đã được biết đến, mỗi cách thức này được mô tả như một mẫu. Hệ thống phát hiện sự lạm dụng chỉ thực hiện kiểm soát đối với các mẫu đã rõ ràng. Mẫu có thể là một xâu bit cố định (ví dụ như một virus đặc tả việc chèn xâu),…dùng để mô tả một tập hay một chuỗi các hành động đáng nghi ngờ.

Ở đây, ta sử dụng thuật ngữ *kịch bản xâm nhập* (intrusion scenario). Một hệ thống phát hiện sự lạm dụng điển hình sẽ liên tục so sánh hành động của hệ thống hiện tại với một tập các kịch bản xâm nhập để cố gắng dò ra kịch bản đang được tiến hành. Hệ thống này có thể xem xét hành động hiện tại của hệ thống được bảo vệ trong thời gian thực hoặc có thể là các bản ghi kiểm tra được ghi lại bởi hệ điều hành.

Các kỹ thuật để phát hiện sự lạm dụng khác nhau ở cách thức mà chúng mô hình hoá các hành vi chỉ định một sự xâm nhập. Các hệ thống phát hiện sự lạm dụng thế hệ đầu tiên sử dụng các luật (rules) để mô tả những gì mà các nhà quản trị an ninh tìm kiếm trong hệ thống. Một lượng lớn tập luật được tích luỹ dẫn đến khó có thể hiểu và sửa đổi bởi vì chúng không được tạo thành từng nhóm một cách hợp lý trong một kịch bản xâm nhập.

Để giải quyết khó khăn này, các hệ thống thế hệ thứ hai đưa ra các biểu diễn kịch bản xen kẽ, bao gồm các tổ chức luật dựa trên mô hình và các biểu diễn về phép biến đổi trạng thái. Điều này sẽ mang tính hiệu quả hơn đối với người dùng hệ thống cần đến sự biểu diễn và hiểu rõ ràng về các kịch bản. Hệ thống phải thường xuyên duy trì và cập nhật để đương đầu với những kịch bản xâm nhập mới được phát hiện.

Do các kịch bản xâm nhập có thể được đặc tả một cách chính xác, các hệ thống phát hiện sự lạm dụng sẽ dựa theo đó để theo vết hành động xâm nhập. Trong một chuỗi hành động, hệ thống phát hiện có thể đoán trước được bước tiếp theo của hành động xâm nhập. Bộ dò tìm phân tích thông tin hệ thống để kiểm tra bước tiếp theo, và khi cần sẽ can thiệp để làm giảm bởi tác hại có thể.

- 1.3.2 phát hiện sự bất thường

Dựa trên việc định nghĩa và mô tả đặc điểm của các hành vi có thể chấp nhận của hệ thống để phân biệt chúng với các hành vi không mong muốn hoặc bất thường, tìm ra các thay đổi, các hành vi bất hợp pháp.

Như vậy, bộ phát hiện sự không bình thường phải có khả năng phân biệt giữa những hiện tượng thông thường và hiện tượng bất thường.

Ranh giới giữa dạng thức chấp nhận được và dạng thức bất thường của đoạn mã và dữ liệu lưu trữ được định nghĩa rõ ràng (chỉ cần một bit khác nhau), còn ranh giới giữa hành vi hợp lệ và hành vi bất thýờng thì khó xác định hơn.

Phát hiện sự không bình thường được chia thành hai loại tĩnh và động

1.3.2.1 Phát hiện tĩnh

Dựa trên giả thiết ban đầu là phần hệ thống được kiểm soát phải luôn luôn không đổi. Ở đây, ta chỉ quan tâm đến phần mềm của vùng hệ thống đó (với giả sử là phần cứng không cần phải kiểm tra). Phần tĩnh của một hệ thống bao gồm 2 phần con: mã hệ thống và dữ liệu của phần hệ thống đó. Hai thông tin này đều được biểu diễn dưới dạng một xâu bit nhị phân hoặc một tập các xâu. Nếu biểu diễn này có sự sai khác so với dạng thức gốc thì hoặc có lỗi xảy ra hoặc một kẻ xâm nhập nào đó đã thay đổi nó. Lúc này, bộ phát hiện tĩnh sẽ được thông báo để kiểm tra tính toàn vẹn dữ liệu.

Cụ thể là: bộ phát hiện tĩnh đưa ra một hoặc một vài xâu bit cố định để định nghĩa trạng thái mong muốn của hệ thống. Các xâu này giúp ta thu được một biểu diễn về trạng thái đó, có thể ở dạng nén. Sau đó, nó so sánh biểu diễn trạng thái thu được với biểu diễn tương tự được tính toán dựa trên trạng thái hiện tại

của cùng xâu bit cố định. Bất kỳ sự khác nhau nào đều là thể hiện lỗi như hỏng phần cứng hoặc có xâm nhập.

Biểu diễn trạng thái tĩnh có thể là các xâu bit thực tế được chọn để định nghĩa cho trạng thái hệ thống, tuy nhiên điều đó khá tốn kém về lưu trữ cũng như về các phép toán so sánh. Do vấn đề cần quan tâm là việc tìm ra được sự sai khác để cảnh báo xâm nhập chứ không phải chỉ ra sai khác ở đâu nên ta có thể sử dụng dạng biểu diễn được nén để giảm chi phí. Nó là giá trị tóm tắt tính được từ một xâu bit cơ sở. Phép tính toán này phải đảm bảo sao cho giá trị tính được từ các xâu bit cơ sở khác nhau là khác nhau. Có thể sử dụng các thuật toán checksums, message-digest (phân loại thông điệp), các hàm băm.

Một số bộ phát hiện xâm nhập kết hợp chặt chẽ với meta-data (dữ liệu mô tả các đối tượng dữ liệu) hoặc thông tin về cấu trúc của đối tượng được kiểm tra. Ví dụ, meta-data cho một log file bao gồm kích cỡ của nó. Nếu kích cỡ của log file tăng thì có thể là một dấu hiệu xâm nhập.

1.3.2.2 Phát hiện động

Trước hết ta đưa ra khái niệm *hành vi* của hệ thống (*behavior*). Hành vi của hệ thống được định nghĩa là một chuỗi các sự kiện phân biệt, ví dụ như rất nhiều hệ thống phát hiện xâm nhập sử dụng các *bản ghi kiểm tra* (*audit record*), sinh ra bởi hệ điều hành để định nghĩa các sự kiện liên quan, trong trường hợp này chỉ những hành vi mà kết quả của nó là việc tạo ra các bản ghi kiểm tra của hệ điều hành mới được xem xét.

Các sự kiện có thể xảy ra theo trật tự nghiêm ngặt hoặc không và thông tin phải được tích luỹ. *Các ngưỡng* được định nghĩa để phân biệt ranh giới giữa việc sử dụng tài nguyên hợp lý hay bất thường.

Nếu không chắc chắn hành vi là bất thường hay không, hệ thống có thể dựa vào các tham số được thiết lập trong suốt quá trình khởi tạo liên quan đến hành vi. Ranh giới trong trường hợp này là không rõ ràng do đó có thể dẫn đến những cảnh báo sai.

Cách thức thông thường nhất để xác định ranh giới là sử dụng các phân loại thống kê và các độ lệch chuẩn. Khi một phân loại được thiết lập, ranh giới có thể được vạch ra nhờ sử dụng một số độ lệch chuẩn. Nếu hành vi nằm bên ngoài thì sẽ cảnh báo là có xâm nhập.

Cụ thể là: các hệ thống phát hiện động thường tạo ra một *profile* (dữ liệu) cơ sở để mô tả đặc điểm các hành vi bình thường, chấp nhận được. Một dữ liệu bao

gồm tập các đo lường được xem xét về hành vi, mỗi đại lượng đo lường gồm nhiều chiều:

- Liên quan đến các lựa chọn: thời gian đăng nhập, vị trí đăng nhập,…
- Các tài nguyên được sử dụng trong cả quá trình hoặc trên một đơn vị thời gian: chiều dài phiên giao dịch, số các thông điệp gửi ra mạng trong một đơn vị thời gian,…
- Chuỗi biểu diễn các hành động.

Sau khi khởi tạo dữ liệu cơ sở, quá trình phát hiện xâm nhập có thể được bắt đầu. Phát hiện động lúc này cũng giống như phát hiện tĩnh ở đó chúng kiểm soát hành vi bằng cách so sánh mô tả đặc điểm hiện tại về hành vi với mô tả ban đầu của hành vi được mong đợi (chính là dữ liệu cơ sở), để tìm ra sự khác nhau. Khi hệ thống phát hiện xâm nhập thực hiện, nó xem xét các sự kiện liên quan đến thực thể hoặc các hành động là thuộc tính của thực thể. Chúng xây dựng thêm một dữ liệu hiện tại.

Các hệ thống phát hiện xâm nhập thế hệ trước phải phụ thuộc vào các bản ghi kiểm tra (*audit record*) để bắt giữ các sự kiện hoặc các hành động liên quan. Các hệ thống sau này thì ghi lại một cơ sở dữ liệu đặc tả cho phát hiện xâm nhập. Một số hệ thống hoạt động với thời gian thực, hoặc gần thời gian thực, quan sát trực tiếp sự kiện trong khi chúng xảy ra hơn là đợi hệ điều hành tạo ra bản ghi mô tả sự kiện.

Khó khăn chính đối với các hệ thống phát hiện động là chúng phải xây dựng các dữ liệu cơ sở một cách chính xác, và sau đó nhận dạng hành vi sai trái nhờ các dữ liệu.

Các dữ liệu cơ sở có thể xây dựng nhờ việc giả chạy hệ thống hoặc quan sát hành vi người dùng thông thường qua một thời gian dài.

- 1.3.3 So sánh giữa hai mô hình

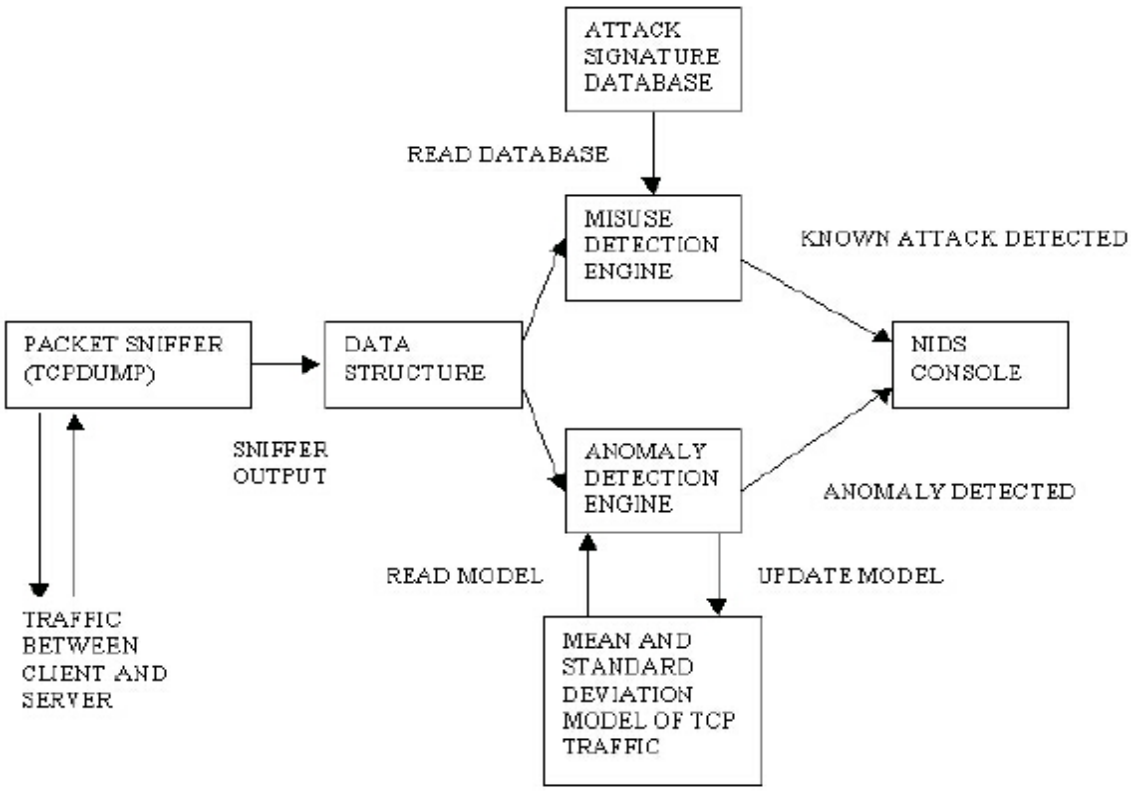
Phát hiện sự lạm dụng	Phát hiện sự bất thường
Bao gồm: • Cơ sở dữ liệu các dấu hiệu tấn công. • Tìm kiếm các so khớp mẫu đúng.	Bao gồm: • Cơ sở dữ liệu các hành động thông thường. • Tìm kiếm độ lệch của hành động thực tế so với hành động thông thường. •
Hiệu quả trong việc phát hiện các	Hiệu quả trong việc phát hiện các

dạng tấn công đã biết, hay các biến thể (thay đổi nhỏ) của các dạng tấn công đã biết. Không phát hiện được các dạng tấn công mới.	dạng tấn công mới mà một hệ thống phát hiện sự lạm dụng bỏ qua.
Dễ cấu hình hơn do đòi hỏi ít hơn về thu thập dữ liệu, phân tích và cập nhật	Khó cấu hình hơn vì đưa ra nhiều dữ liệu hơn, phải có được một khái niệm toàn diện về hành vi đã biết hay hành vi được mong đợi của hệ thống
Đưa ra kết luận dựa vào phép so khớp mẫu (*pattern matching*).	Đưa ra kết quả dựa vào tương quan bằng thống kê giữa hành vi thực tế và hành vi được mong đợi của hệ thống (hay chính là dựa vào độ lệch giữa thông tin thực tế và ngưỡng cho phép).
Có thể kích hoạt một thông điệp cảnh báo nhờ một dấu hiệu chắc chắn, hoặc cung cấp dữ liệu hỗ trợ cho các dấu hiệu khác.	Có thể hỗ trợ việc tự sinh thông tin hệ thống một cách tự động nhưng cần có thời gian và dữ liệu thu thập được phải rõ ràng.

Bảng So sánh 2 mô hình phát hiện

Để có được một hệ thống phát hiện xâm nhập tốt nhất ta tiến hành kết hợp cả hai phương pháp trên trong cùng một hệ thống. Hệ thống kết hợp này sẽ cung cấp khả năng phát hiện nhiều loại tấn công hơn và hiệu quả hơn.

Sơ đồ hệ thống kết hợp như sau:



Hình I : Hệ thống kết hợp 2 mô hình phát hiện

- 1.4 Một số sản phẩm của IDS/IPS

Phần này giới thiệu một số sản phẩm IDS, IPS thương mại cũng như miễn phí phổ biến, những sản phẩm điển hình trong lĩnh vực phát hiện và phòng chống xâm nhập.

Cisco IDS-4235

Cisco IDS (còn có tên là NetRanger) là một hệ thống NIDS, có khả năng theo dõi toàn bộ lưu thông mạng và đối sánh từng gói tin để phát hiện các dấu hiệu xâm nhập.

Cisco IDS là một giải pháp riêng biệt, được Cisco cung cấp đồng bộ phần cứng và phần mềm trong một thiết bị chuyên dụng.

Giải pháp kỹ thuật của Cisco IDS là một dạng lai giữa giải mã (decode) và đối sánh (grep). Cisco IDS hoạt động trên một hệ thống Unix được tối ưu hóa về cấu hình và có giao diện tương tác CLI (Cisco Command Line Interface) quen thuộc của Cisco.

ISS Proventia A201

Proventia A201 là sản phẩm của hãng Internet Security Systems. Về mặt bản chất, Proventia không chỉ là một hệ thống phần mềm hay phần cứng mà nó là một hệ thống các thiết bị được triển khai phân tán trong mạng được bảo vệ. Một hệ thống Proventia bao gồm các thiết bị sau:

- Intrusion Protection Appliance: Là trung tâm của toàn bộ hệ thống Proventia. Nó lưu trữ các cấu hình mạng, các dữ liệu đối sánh cũng như các quy định về chính sách của hệ thống. Về bản chất, nó là một phiên bản Linux với các driver thiết bị mạng được xây dựng tối ưu cũng như các gói dịch vụ được tối thiểu hóa.
- Proventia Network Agent: Đóng vai trò như các bộ cảm biến (sensor). Nó được bố trí tại những vị trí nhạy cảm trong mạng nhằm theo dõi toàn bộ lưu thông trong mạng và phát hiện những nguy cơ xâm nhập tiềm ẩn.
- SiteProtector: Là trung tâm điều khiển của hệ thống Proventia. Đây là nơi người quản trị mạng điều khiển toàn bộ cấu hình cũng như hoạt động của hệ thống.

Với giải pháp của Proventia, các thiết bị sẽ được triển khai sao cho phù hợp với cấu hình của từng mạng cụ thể để có thể đạt được hiệu quả cao nhất.

NFR NID-310

NFR là sản phẩm của NFR Security Inc. Cũng giống như Proventia, NFR NID là một hệ thống hướng thiết bị (appliance-based). Điểm đặc biệt trong kiến trúc của NFR NID là họ các bộ cảm biến có khả năng thích ứng với rất nhiều mạng khác nhau từ mạng 10Mbps đến các mạng gigabits với thông lượng rất lớn.

Một điểm đặc sắc của NFR NID là mô hình điều khiển ba lớp. Thay vì các thiết bị trong hệ thống được điều khiển trực tiếp bởi một giao diện quản trị (Administration Interface – AI) riêng biệt, NFR cung cấp một cơ chế điều khiển tập trung với các middle-ware làm nhiệm vụ điều khiển trực tiếp các thiết bị.

SNORT

Snort là phần mềm IDS mã nguồn mở, được phát triển bởi Martin Roesh. Snort đầu tiên được xây dựng trên nền Unix sau đó phát triển sang các nền tảng khác. Snort được đánh giá là IDS mã nguồn mở đáng chú ý nhất với những tính năng rất mạnh. Chi tiết về Snort sẽ được trình bày trong phần chương II của đề tài .

CHƯƠNG II : NGHIÊN CỨU ỨNG DỤNG SNORT TRONG IDS/IPS

- 2.1 Giới thiệu về snort

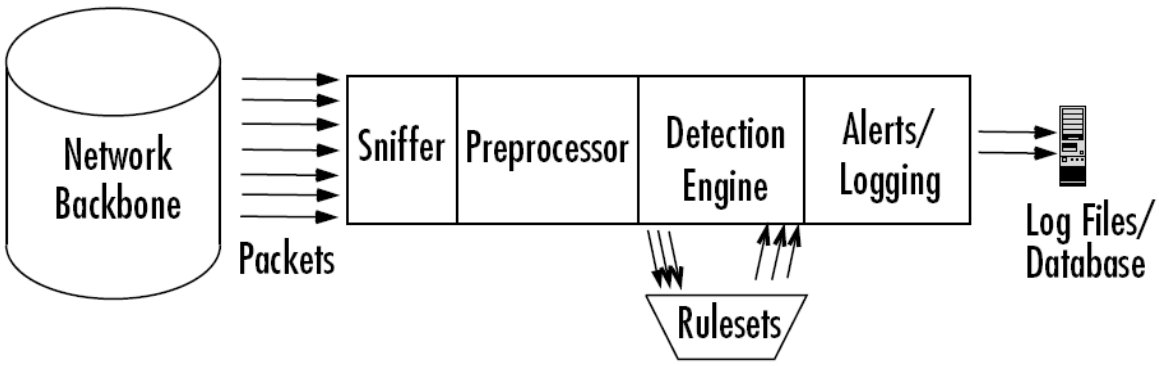
Snort là một NIDS được Martin Roesh phát triển dưới mô hình mã nguồn mở. Tuy Snort miễn phí nhưng nó lại có rất nhiều tính năng tuyệt vời mà không phải sản phẩm thương mại nào cũng có thể có được. Với kiến trúc thiết kế theo kiểu module, người dùng có thể tự tăng cường tính năng cho hệ thống Snort của mình bằng việc cài đặt hay viết thêm mới các module. Cơ sở dữ liệu luật của Snort đã lên tới 2930 luật và được cập nhật thường xuyên bởi một cộng đồng người sử dụng. Snort có thể chạy trên nhiều hệ thống nền như Windows, Linux, OpenBSD, FreeBSD, NetBSD, Solaris, HP-UX, AIX, IRIX, MacOS.

Bên cạnh việc có thể hoạt động như một ứng dụng thu bắt gói tin thông thường, Snort còn có thể được cấu hình để chạy như một NIDS. Snort hỗ trợ khả năng hoạt động trên các giao thức sau: Ethernet, 802.11,Token Ring, FDDI, Cisco HDLC, SLIP, PPP, và PF của OpenBSD.

2.2 Kiến trúc của snort

Snort bao gồm nhiều thành phần, với mỗi phần có một chức năng riêng. Các phần chính đó là:

- Môđun giải mã gói tin (Packet Decoder)
- Môđun tiền xử lý (Preprocessors)
- Môđun phát hiện (Detection Engine)
- Môđun log và cảnh báo (Logging and Alerting System)
- Môđun kết xuất thông tin (Output Module)
- Kiến trúc của Snort được mô tả trong hình sau:
-

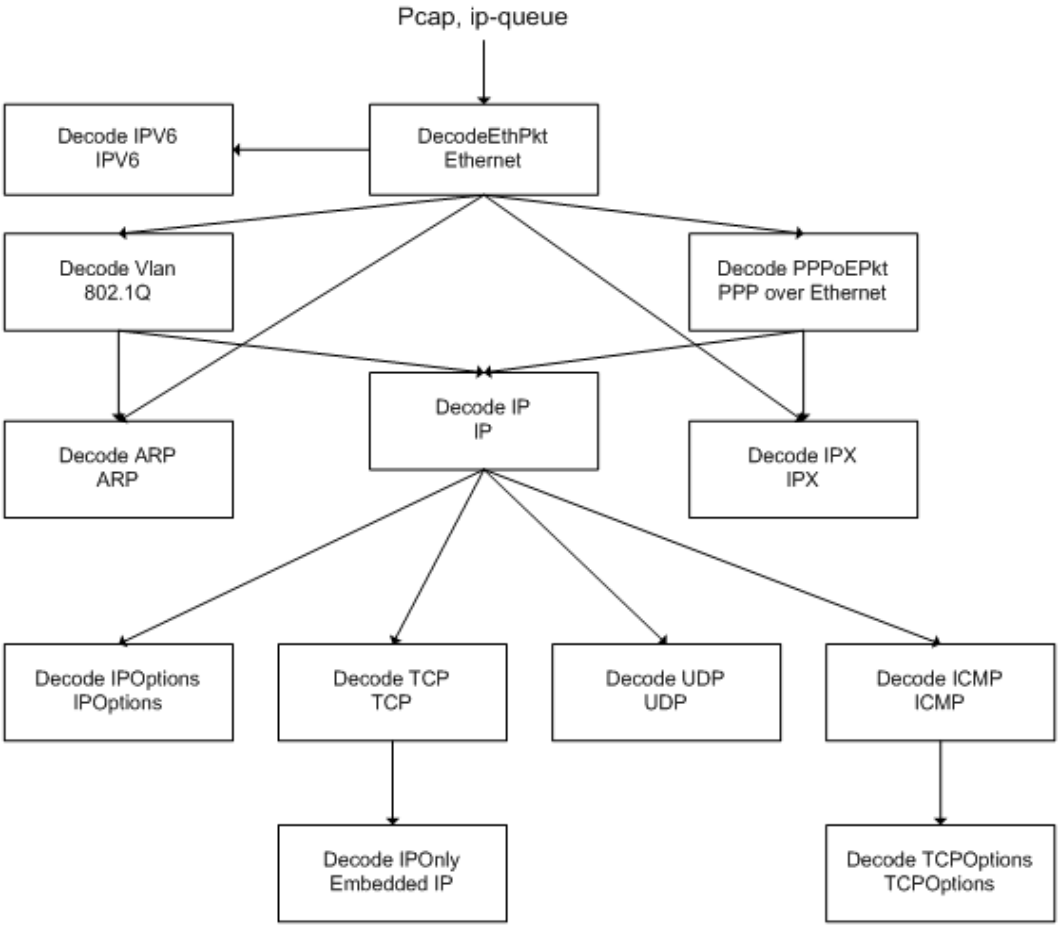


Hình IV : Mô hình kiến trúc hệ thống Snort

Khi Snort hoạt động nó sẽ thực hiện việc lắng nghe và thu bắt tất cả các gói tin nào di chuyển qua nó. Các gói tin sau khi bị bắt được đưa vào Môđun Giải mã gói tin. Tiếp theo gói tin sẽ được đưa vào môđun Tiền xử lý, rồi môđun Phát hiện. Tại đây tùy theo việc có phát hiện được xâm nhập hay không mà gói tin có thể được bỏ qua để lưu thông tiếp hoặc được đưa vào môđun Log và cảnh báo để xử lý. Khi các cảnh báo được xác định môđun Kết xuất thông tin sẽ thực hiện việc đưa cảnh báo ra theo đúng định dạng mong muốn. Sau đây ta sẽ đi sâu vào chi tiết hơn về cơ chế hoạt động và chức năng của từng thành phần.

2.2.1 Modun giải mã gói tin

Snort sử dụng thư viện pcap để bắt mọi gói tin trên mạng lưu thông qua hệ thống. Hình sau mô tả việc một gói tin Ethernet sẽ được giải mã thế nào:



Hình V: Xử lý một gói tin Ethernet

Một gói tin sau khi được giải mã sẽ được đưa tiếp vào môđun tiền xử lý.

2.2.2 Mô đun tiền xử lý

Môđun tiền xử lý là một môđun rất quan trọng đối với bất kỳ một hệ thống IDS nào để có thể chuẩn bị gói dữ liệu đưa và cho môđun Phát hiện phân tích. Ba nhiệm vụ chính của các môđun loại này là:

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 ta 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 stream 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.

Giải mã và chuẩn hóa giao thức (decode/normalize): công việc phát hiện xâm nhập dựa trên dấu hiệu nhận dạng nhiều khi bị thất bại khi kiểm tra các giao thức có 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 ta có dấu hiệu nhận dạng “scripts/iisadmin”, kẻ tấn công có thể vượt qua được bằng cách tùy biến các yêu cấu 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 môđun 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 môđun 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.

Phát hiện các xâm nhập bất thường (nonrule /anormal): các plugin 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. Các môđun tiền xử lý dạng này có thể thực hiện việc phát hiện xâm nhập theo bất cứ cách nào mà ta nghĩ ra từ đó tăng cường thêm tính năng cho Snort. Ví dụ, một plugin tiền xử lý có nhiệm vụ thống kê thông lượng mạng tại thời điểm bình thường để rồi khi có thông lượng mạng bất thường xảy ra nó có thể tính toán, phát hiện và đưa ra cảnh báo (phát hiện xâm nhập theo mô hình thống kê). Phiên bản hiện tại của Snort có đi kèm hai plugin giúp phát hiện các xâm nhập bất thường đó là portscan và bo (backoffice). Portcan dùng để đưa ra cảnh báo khi kẻ tấn công thực hiện việc quét các cổng của hệ thống để tìm lỗ hổng. Bo dùng để đưa ra cảnh báo khi hệ thống đã bị nhiễm trojan backoffice và kẻ tấn công từ xa kết nối tới backoffice thực hiện các lệnh từ xa.

2.2.3 Môđun phát hiện

Đây là môđun quan trọng nhất của Snort. Nó chịu trách nhiệm phát hiện các dấu hiệu xâm nhập. Môđun 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.

Một vấn đề rất quan trọng trong môđun phát hiện là vấn đề thời gian xử lý các gói tin: một IDS thường nhận được rất nhiều gói tin và bản thân nó cũng có rất nhiều các luật xử lý. Có thể mất những khoảng thời gian khác nhau cho việc xử lý các gói tin khác nhau. Và khi thông lượng mạng quá lớn có thể xảy ra việc bỏ sót hoặc không phản hồi được đúng lúc. Khả năng xử lý của môđun phát hiện dựa trên một số yếu tố như: số lượng các luật, tốc độ của hệ thống đang chạy Snort, tải trên mạng. Một số thử nghiệm cho biết, phiên bản hiện tại của Snort khi được tối ưu hóa chạy trên hệ thống có nhiều bộ vi xử lý và cấu hình máy tính tương đối mạnh thì có thể hoạt động tốt trên cả các mạng cỡ Giga.

Một môđun 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, …
- Phần tải của gói tin (bạn cũng có thể áp dụng các luật lên các phần dữ liệu được truyền đi của gói tin)

Một vấn đề nữa trong Môđun 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 ðýợc ðánh thứ tự ýu tiên, nên một gói tin khi bị phát hiện bởi nhiều luật khác nhau, cảnh báo ðýợc ðýa ra sẽ là cảnh báo ứng với luật có mức ýu tiên lớn nhất.

2.2.4 Môđun log và cảnh báo

Tùy thuộc vào việc môđun Phát hiện có nhận dạng đuợc xâm nhập hay không mà gói tin có thể bị ghi log hoặc đưa ra cảnh báo. Các file log là các file text dữ liệu trong đó có thể được ghi dưới nhiều định dạng khác nhau chẳng hạn tcpdump.

2.2.5 Mô đun kết xuất thong tin

Môđun 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 log file
- 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 log file 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, ta có thể viết các môđun kết xuất thông tin riêng tuỳ theo mục đích sử dụng.

2.3 Bộ luật của snort

2.3.1 Giới thiệu

Cũng giống như virus, hầu hết các hoạt động tấn công hay xâm nhập đều có các dấu hiệu riêng. Các thông tin về các dấu hiệu này sẽ được sử dụng để tạo nên các luật cho Snort. Thông thường, các bẫy (honey pots) được tạo ra để tìm hiểu xem các kẻ tấn công làm gì cũng như các thông tin về công cụ và công

nghệ chúng sử dụng. Và ngược lại, cũng có các cơ sở dữ liệu về các lỗ hổng bảo mật mà những kẻ tấn công muốn khai thác. Các dạng tấn công đã biết này được dùng như các dấu hiệu để phát hiện tấn công xâm nhập. Các dấu hiệu đó có thể xuất hiện trong phần header của các gói tin hoặc nằm trong phần nội dung của chúng. Hệ thống phát hiện của Snort hoạt động dựa trên các luật (rules) và các luật này lại được dựa trên các dấu hiệu nhận dạng tấn công. Các luật có thể được áp dụng cho tất cả các phần khác nhau của một gói tin dữ liệu .

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.

2.3.2 Cấu trúc luật của Snort

Hãy xem xét một ví dụ đơn giản :

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

Ta thấy cấu trúc của một luật có dạng như sau:

Hình VI : Cấu trúc luật của Snort

Diễn giải:

Tất cả các Luật của Snort về logic đều gồm 2 phần: Phần header và phần Option.

- Phần 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.
- Phần Option chứa một thông điệp cảnh báo và các thông tin về các phần của gói tin dùng để tạo nên cảnh báo. Phần Option chứa các tiêu chuẩn phụ thêm để đối sánh luật với gói tin. Một luật có thể phát hiện được một hay nhiều hoạt động thăm dò hay tấn công. Các luật thông minh có khả năng áp dụng cho nhiều dấu hiệu xâm nhập.

Dưới đây là cấu trúc chung của phần Header của một luật Snort:

Hình VII : Header luật của Snort

- 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.

Ví dụ:

alert icmp any any -> any any (msg: "Ping with TTL=100";ttl: 100;)

Phần đứng trước dấu mở ngoặc là phần Header của luật còn phần còn lại là phần Option. Chi tiết của phần Header như sau:

- Hành động của luật ở đây là "alert" : một cảnh báo sẽ được tạo ra nếu như các điều kiện của gói tin là phù hợp với luật(gói tin luôn được log lại mỗi khi cảnh báo được tạo ra).
- Protocol của luật ở đây là ICMP tức là luật chỉ áp dụng cho các gói tin thuộc loại ICMP. Bởi vậy, nếu như một gói tin không thuộc loại ICMP thì phần còn lại của luật sẽ không cần đối chiếu.
- Địa chỉ nguồn ở đây là "any": tức là luật sẽ áp dụng cho tất cả các gói tin đến từ mọi nguồn còn cổng thì cũng là "any" vì đối với loại gói tin ICMP thì cổng không có ý nghĩa. Số hiệu cổng chỉ có ý nghĩa với các gói tin thuộc loại TCP hoặc UDP thôi.
- Còn phần Option trong dấu đóng ngoặc chỉ ra một cảnh báo chứa dòng "Ping with TTL=100" sẽ được tạo khi tìm thấy điều kiện TTL=100. TTL là Time To Live là một trường trong Header IP.
-

2.3.2.1 Phần tiêu đề

Như phần trên đã trình bày, Header của luật bao gồm nhiều phần. Sau đây, là chi tiết cụ thể của từng phần một.

Hành động của luật (Rule Action)

Là phần đầu tiên của luật, chỉ ra hành động nào được thực hiện khi mà các điều kiện của luật được thoã 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 ta

có thể tạo ra các hành động riêng tuỳ thuộc vào yêu cầu của mình. Đối với các phiên bản trước của Snort thì khi nhiều luật là phù hợp với một gói tin nào đó thì chỉ một luật được áp dụng. Sau khi áp dụng luật đầu tiên thì các luật tiếp theo sẽ không áp dụng cho gói tin ấy nữa. Nhưng đối với các phiên bản sau của Snort thì tất cả các luật sẽ được áp dụng gói tin đó.

- 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: Hành động này dùng để log gói tin. Có thể log vào file hay vào cơ sở dữ liệu tuỳ thuộc vào nhu cầu của mình.
- Alert: Gửi một thông điệp cảnh báo khi dấu hiệu xâm nhập được phát hiện. Có nhiều cách để gửi thông điệp như gửi ra file hoặc ra một Console. Tất nhiên là sau khi gửi thông điệp cảnh báo thì gói tin sẽ được log lại.
- 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
}
```

ruletype là từ khoá.

Hành động được định nghĩa chính xác trong dấu ngoặc nhọn: có thể là một hàm viết bằng ngôn ngữ C chẳng hạn.

Ví dụ như:

```
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.

Protocols

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

Nếu là IP thì Snort sẽ kiểm tra header của lớp liên kết để xác định loại gói tin. Nếu bất kì giao thức nào khác được sử dụng thì Snort sử dụng header IP để xác định loại protocol. Protocol chỉ đóng vai trò trong việc chỉ rõ tiêu chuẩn trong phần header của luật. Phần option của luật có thể có các điều kiện không liên quan gì đến protocol.

Address

Có hai phần địa chỉ trong một luật của Snort. Các địa chỉ này được dùng để kiểm tra nguồn sinh ra và đích đến 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.

Địa chỉ được viết ngay theo sau một dấu gạch chéo và số bít trong subnet mask. Ví dụ như địa chỉ 192.168.2.0/24 thể hiện mạng lớp C 192.168.2.0 với 24 bít của subnet mask. Subnet mask 24 bít chính là 255.255.255.0. Ta biết rằng :

- Nếu subnet mask là 24 bít thì đó là mạng lớp C
- Nếu subnet mask là 16 bít thì đó là mạng lớp B
- Nếu subnet mask là 8 bít thì đó là mạng lớp A
- Nếu subnet mask là 32 bít thì đó là địa chỉ IP đơn.

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 cổng 80.

Ngăn chặn địa chỉ hay loại trừ địa chỉ

Snort cung cấp cho ta kĩ thuật để loại trừ địa chỉ bằng cách sử dụng dấu phủ định (dấu !). Dấu phủ định này đứng trước địa chỉ sẽ chỉ cho Snort không kiểm tra các gói tin đến từ hay đi tới địa chỉ 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;)

Danh sách địa chỉ

Ta có thể định rõ ra danh sách các địa chỉ trong một luật của Snort. Ví dụ nếu bạn muốn áp dụng luật cho tất cả các gói tin trừ các gói xuất phát từ hai mạng lớp C 192.168.2.0 và 192.168.8.0 thì luật được viết như sau:

alert icmp ![192.168.2.0/24, 192.168.8.0/24] any -> any any (msg: “Ping with TTL=100”; ttl: 100;)

Hai dấu [] chỉ cần dùng khi có dấu ! đứng trước.

Cổng (Port Number)

Số hiệu cổng dùng để áp dụng luật cho các gói tin đến từ hoặc đi đến một cổng hay một phạm vi cổng cụ thể nào đó. Ví dụ ta có thể sử dụng số cổng nguồn là 23 để áp dụng luật cho tất cả các gói tin đến từ một server Telnet. 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”;)

Số hiệu cổng chỉ hữu dụng khi ta muốn áp dụng một luật chỉ cho một loại gói tin dữ liệu cụ thể nào đó. Ví dụ như là một luật để chống hack cho web thì ta chỉ cần sử dụng cổng 80 để phát hiện tấn công.

Dãy cổng hay phạm vi cổng:

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

Ví dụ :

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

Ta cũng có thể dùn cổng theo kiểu cận trên và cận dưới, tức là chỉ sử dụng cổng bắt đầu hoặc cổng kết thúc mà thôi. Ví dụ như là "1024:" hoặc là ":2048"

Dấu phủ định cũng được áp dụng trong việc sử dụng cổng. Ví dụ sau sẽ log tất cả các gói tin ngoại trừ các gói tin xuất phát từ cổng 53.

log udp any !53 -> any any log udp

Sau đây là một số cổng thông dụng hay là các cổng của các dịch vụ thông dụng nhất:

- 20 FTP data
- 21 FTP
- 22 SSH
- 23 Telnet
- 24 SMTP
- 53 DNS Server
- 80 HTTP
- 110 POP3
- 161 SNMP
- 443 HTTPS
- 3360 MySQL

Hướng – 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.

2.3.2.2 Các tùy chọn

Phần Rule Option nằm ngay sau phần Rule Header và được bao bọc trong dấu ngoặc đơn. Nếu có nhiều option thì các option sẽ được phân cách với nhau bằng dấu chấm phẩy ",".Nếu nhiều option được sử dụng thì các option này phải đồng thời được thoã mãn tức là theo logic các option này liên kết với nhau bằng AND.

Mọi option được định nghĩa bằng các từ khoá. Một số các option còn chứa các tham số. Nói chung một option gồm 2 phần: một từ khoá và một tham số, hai phần này phân cách nhau bằng dấu hai chấm. Ví dụ đã dùng :

msg: "Detected confidented";

msg là từ khoá còn "Detected confidented" là tham số.

Sau đây là chi tiết một số các option của luật Snort.

Từ khoá ack

Trong header TCP có chứa trường Acknowledgement Number với độ dài 32 bit. Trường này có ý nghĩa là chỉ ra số thứ tự tiếp theo gói tin TCP của bên gửi đang được chờ để nhận. Trường này chỉ có ý nghĩa khi mà cờ ACK được thiết lập.

Các công cụ như Nmap sử dụng đặc điểm này ping một máy. Ví dụ, nó có thể gửi một gói tin TCP tới cổng 80 với cờ ACK được bật và số thứ tự là 0. Bởi vậy, bên nhận sẽ thấy gói tin không hợp lệ và sẽ gửi trở lại gói tin RST. Khi mà Nmap nhận được gói tin RST thì tức là địa chỉ đích đang "sống". Phương pháp này vẫn làm việc tốt đối với các máy không trả lời gói tin thuộc dạng ping ICMP ECHO REQUEST.

Vậy để kiểm tra loại ping TCP này thì ta có thể dùng luật như sau:

alert tcp any any -> 192.168.1.0/24 any (flags: A; ack: 0; msg: "TCP ping detected")

Từ khoá classtype

Các luật có thể được phân loại và gán cho một số chỉ độ ưu tiên nào đó để nhóm và phân biệt chúng với nhau. Để hiểu rõ hơn về từ khoá này ta đầu tiên phải hiểu được file *classification.config* (được bao gồm trong file *snort.conf* sử dụng từ khoá include). Mỗi dòng trong file classification.config có cú pháp như sau:

config classification: name, description, priority

trong đó:

- name: là tên dùng để phân loại, tên này sẽ được dùng với từ khoá classtype trong các luật Snort.
- description: mô tả về loại lớp này
- priority: là một số chỉ độ ưu tiên mặc định của lớp này. Độ ưu tiên này có thể được điều chỉnh trong từ khoá priority của phần option trong luật của Snort.

Ví dụ :

config classification: DoS , Denial of Service Attack, 2

và trong luật:

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

alert udp any any -> 192.168.1.0/24 6838 (msg:"DoS"; content: "server"; classtype: DoS; priority: 1;)

Trong câu lệnh thứ 2 thì ta đã ghi đè lên giá trị priority mặc định của lớp đã định nghĩa.

Từ khoá content

Một đặc tính quan trọng của Snort là nó có khả năng tìm một mẫu dữ liệu bên trong một gói tin. Mẫu này có thể dưới dạng chuỗi ASCII hoặc là một chuỗi nhị phân dưới dạng các kí tự hệ 16. Giống như virus, các tấn công cũng có các dấu hiệu nhận dạng và từ khoá content này dùng để tìm các dấu hiệu đó bên trong gói tin. Ví dụ:

alert tcp 192.168.1.0/24 any -> ![192.168.1.0/24] any (content: "GET"; msg: "GET match";)

Luật trên tìm mẫu "GET" trong phần dữ liệu của tất cả các gói tin TCP có nguồn đi từ mạng 192.168.1.0/24 và đi đến các địa chỉ không thuộc mạng đó. Từ "GET" này rất hay được dùng trong các tấn công HTTP.

Một luật khác cũng thực hiện đúng nhiệm vụ giống như lệnh trên nhưng mẫu dữ liệu lại dưới dạng hệ 16 là:

alert tcp 192.168.1.0/24 any -> ![192.168.1.0/24] any (content: "|47 45 54|"; msg: "GET match";)

Để ý rằng số 47 ở hệ 16 chính là bằng kí tự ASCII : G và tương tự 45 là E và 54 là T. Ta có thể dùng cả hai dạng trên trong cùng một luật nhưng nhớ là phải để dạng thập lục phân giữa cặp kí tự ||.

Tuy nhiên khi sử dụng từ khoá content ta cần nhớ rằng:

Đối sánh nội dung sẽ phải xử lý tính toán rất lớn và ta phải hết sức cân nhắc khi sử dụng nhiều luật có đối sánh nội dung.

Ta có thể sử dụng nhiều từ khoá content trong cùng một luật để tìm nhiều dấu hiệu trong cùng một gói tin.

Đối sánh nội dung là công việc rất nhạy cảm.

Có 3 từ khoá khác hay được dùng cùng với từ khoá content dùng để bổ sung thêm các điều kiện để tìm kiếm là :

- offset: dùng để xác định vị trí bắt đầu tìm kiếm (chuỗi chứa trong từ khoá content) là offset tính từ đầu phần dữ liệu của gói tin. Ví dụ sau sẽ tìm chuỗi "HTTP" bắt đầu từ vị trí cách đầu đoạn dữ liệu của gói tin là 4 byte:

alert tcp 192.168.1.0/24 any -> any any (content: “HTTP”; offset: 4; msg: “HTTP matched”;)

- dept : dùng để xác định vị trí mà từ đó Snort sẽ dừng việc tìm kiếm.Từ khoá này cũng thường được dùng chung với từ khoá offset vừa nêu trên.
- Ví dụ:

alert tcp 192.168.1.0/24 any -> any any (content: “HTTP”; offset: 4; dept: 40; msg: “HTTP matched”;).

- Từ khoá này sẽ giúp cho việc tiêu tốn thời gian tìm kiếm khi mà đoạn dữ liệu trong gói tin là khá lớn.
- content-list: được sử dụng cùng với một file. Tên file (được chỉ ra trong phần tham số của từ khoá này) là một file text chứa danh sách các chuỗi cần tìm trong phần dữ liệu của gói tin. Mỗi chuỗi nằm trên một dòng riêng biệt. Ví dụ như file test có dạng như sau:
- “test”

“Snort”

“NIDS”

và ta có luật sau:

alert tcp 192.168.1.0/24 any -> any any (content-list: “test”;msg: “This is my Test”;).

Ta cũng có thể dùng kí tự phủ định ! trước tên file để cảnh báo đối với các gói tin không tìm thấy một chuỗi nào trong file đó.

Từ khoá dsize

Dùng để đối sánh theo chiều dài của phần dữ liệu. Rất nhiều tấn công sử dụng lỗi tràn bộ đệm bằng cách gửi các gói tin có kích thước rất lớn. Sử dụng từ khoá này, ta có thể so sánh độ lớn của phần dữ liệu của gói tin với một số nào đó.

alert ip any any -> 192.168.1.0/24 any (dsize: > 6000; msg: “Goi tin co kich thuoc lon”;)

Từ khoá flags

Từ khoá này được dùng để phát hiện xem những bit cờ flag nào được bật (thiết lập) trong phần TCP header của gói tin. Mỗi cờ có thể được sử dụng như một tham số trong từ khoá flags. Sau đây là một số các cờ sử dụng trong từ khoá flags:

Flag	Kí tự tham số dùng trong luật của Snort
FIN (Finish Flag)	F
SYN – Sync Flag	S
RST – Reset Flag	R
PSH – Push Flag	P
ACK – Acknowledge Flag	A
URG – Urgent Flag	U
Reserved Bit 1	1
Reserved Bit 2	2
No Flag set	0

Bảng Các cờ sử dụng với từ khoá flags

Ta có thể sử dụng các dấu +, * và ! để thực hiện các phép toán logic AND, OR và NOT trên các bit cờ muốn kiểm tra. Ví dụ luật sau đây sẽ phát hiện một hành động quét dùng gói tin TCP SYN-FIN:

alert tcp any any -> 192.168.1.0/24 any (flags: SF; msg: "SYNC-FIN packet detected";)

Từ khoá fragbits

Phần IP header của gói tin chứa 3 bit dùng để chống phân mảnh và tổng hợp các gói tin IP. Các bit đó là:

- Reserved Bit (RB) dùng để dành cho tương lai.
- Don't Fragment Bit (DF): nếu bit này được thiết lập thì tức là gói tin đó không bị phân mảnh.
- More Fragments Bit (MF): nếu được thiết lập thì tức là các phần khác (gói tin bị phân mảnh) của gói tin vẫn đang còn trên đường đi mà chưa tới đích. Nếu bit này không được thiết lập thì có nghĩa là đây là phần cuối cùng của gói tin (hoặc là gói duy nhất). Điều này xuất phát từ nguyên nhân: Nơi gửi đi phải chia gói tin IP thành nhiều đoạn nhỏ do phụ thuộc vào Đơn vị truyền dữ liệu lớn nhất cho phép (Maximum Transfer Units - MTU) trên đường truyền. Kích thước của gói tin không được phép vượt quá kích thước lớn nhất này. Do vậy, bit MF này giúp bên đích có thể tổng hợp lại các phần khác nhau thành một gói tin hoàn chỉnh.

Đôi khi các bit này bị các hacker sử dụng để tấn công và khai thác thông tin trên mạng của ta. Ví dụ, bit DF có thể được dùng để tìm MTU lớn nhất và nhỏ nhất trên đường đi từ nguồn xuất phát đến đích đến.

Sử dụng fragbits, ta có thể kiểm tra xem các bit trên có được thiết lập hay không. Ví dụ luật sau sẽ phát hiện xem bit DF trong gói tin ICMP có được bật hay không:

alert icmp any any -> 192.168.1.0/24 any (fragbits: D; msg: “Dont Fragment bit set”;)

Trong luật này , D dùng cho bit DF, R cho bit dự trữ và M cho bit MF. Ta cũng có thể dùng dấu phủ định ! trong luật này để kiểm tra khi bit không được bật:

alert icmp any any -> 192.168.1.0/24 any (fragbits: !D; msg: “Dont Fragment bit not set”;)

2.4 Chế độ ngăn chặn của Snort : Snort – Inline

2.4.1 Tích hợp khả năng ngăn chặn vào Snort

Snort-inline là một nhánh phát triển của Snort do William Metcalf khởi xướng và lãnh đạo. Đến phiên bản 2.3.0 RC1 của Snort, inline-mode đã được tích hợp vào bản chính thức do snort.org phát hành. Sự kiện này đã biến Snort từ một IDS thuần túy trở thành một hệ thống có các khả năng của một IPS, mặc dù chế độ này vẫn chỉ là tùy chọn chứ không phải mặc định.

Ý tưởng chính của inline-mode là kết hợp khả năng ngăn chặn của iptables vào bên trong snort. Điều này được thực hiện bằng cách thay đổi môđun phát hiện và môđun xử lý cho phép snort tương tác với iptables. Cụ thể, việc chặn bắt các gói tin trong Snort được thực hiện thông qua Netfilter và thư viện libpcap sẽ được thay thế bằng việc sử dụng ipqueue và thư viện libipq. Hành động ngăn chặn của snort-inline sẽ được thực hiện bằng devel-mode của iptables.

2.4.2 Những bổ sung cho cấu trúc luật của Snort hỗ trợ Inline mode

Để hỗ trợ tính năng ngăn chặn của Snort-inline, một số thay đổi và bổ sung đã được đưa vào bộ luật Snort. Đó là đưa thêm 3 hành động DROP, SDROP, INJECT và thay đổi trình tự ưu tiên của các luật trong Snort.

DROP

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

SDROP

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

REJECT

Hành động REJECT yêu cầu iptables từ chối gói tin, có nghĩa là iptables sẽ loại bỏ và gửi lại một thông báo cho nguồn gửi gói tin đó. Hành động REJECT không ghi lại bất cứ thông tin gì.

Trình tự ưu tiên của các luật

Trong các phiên bản gốc, trình tự ưu tiên của các hành động trong Snort là : activation->dynamic-> alert->pass->log

Trong inline-mode, trình tự ưu tiên này được thay đổi như sau : activation->dynamic->pass->drop->sdrop->reject->alert->log

CHƯƠNG III: CÀI ĐẶT VÀ CẤU HÌNH SNORT TRÊN NỀN CENTOS, THỬ NGHIỆM KHẢ NĂNG PHẢN ỨNG CỦA SNORT IDS/IPS

3.1 SƠ LƯỢC VỀ QUÁ TRÌNH CÀI ĐẶT

3.1.1 Cài các gói yêu cầu sau

- Lần lượt cài các gói phụ thuộc:

(mysql, mysql-bench, mysql-server, mysql-devel, yum-utils, php-mysql, httpd, gcc, pcre-devel, php-gd, gd, distcache-devel, mod_ssl, glib2-devel, gcc-c++, libpcap-devel, php, php-pear)

- dùng lệnh (yum install package) để cài đặt cho các gói tin.
- một số gói cần thiế cho snort cần phải biên dịch từ soure

(libnet, libdnet, daq, pcre, Snortinline, BASE, adodb)

cd /tmp

wget http://www.filewatcher.com/m/libnet-.0.2a.tar.gz.140191.0.0.html

wget http://code.google.com/p/libdnet/downloads/detail?name=libdnet-1.12.tgz&can=2&q=

wget http://sourceforge.net/projects/adodb/files/adodb-php-4-and-5/adodb-4991-for-php/adodb4991.tgz/download

wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-7.9.tar.gz
download snort_inline http://snort-inline.sourceforge.net/download.html
download base từ nguồn http://sourceforge.net/projects/secureideas/files/
- sau khi download các gói về tiến hành biên dịch cho các gói
+ biên dịch gói libnet
cd /tmp (di chuyển vào thư mục tmp)
tar xvzf libnet-1.0.2a.tar.gz (giả nén libnet)
cd Libnet-1.0.2a (di chuyển vào thư mục Libnet-1.0.2a vừa giải nén)
./configure && make && make install (kiểm tra câu hình và biên dịch libnet, dấu && có ý nghĩa nếu câu lệnh trước nó thành công thì mới thực hiện câu lệnh đứng sau)

+ biên dịch gói libdnet
cd /tmp (di chuyển vào thư mục tmp)
tar libdnet-1.12.tgz
cd libdnet-1.12 (di chuyển vào thư mục libdnet-1.12 vừa giải nén)
./configure && make && make install (kiểm tra câu hình và biên dịch libdnet, dấu && có ý nghĩa nếu câu lệnh trước nó thành công thì mới thực hiện câu lệnh đứng sau)

+ biên dịch gói daq
cd /tmp (di chuyển đến thư mục tmp)
tar zxvf daq-0.3.tar.gz (giải nén daq)
cd daq-0.3 (di chuyển đến thư mục daq-0.3 vừa giải nén được)
./configure && make && make install (kiểm tra câu hình và biên dịch daq, dấu && có ý nghĩa nếu câu lệnh trước nó thành công thì mới thực hiện câu lệnh đứng sau)
+ biên dịch pcre
cd /tmp
tar xvzf pcre-7.9.tar.gz
cd pcre-7.9
./configure && make && make install (kiểm tra câu hình và biên dịch pcre, dấu && có ý nghĩa nếu câu lệnh trước nó thành công thì mới thực hiện câu lệnh đứng sau)
+ biên dịch snort_inline
cd /tmp

tar -xvf snort_inline-2.4.5a.tar.gz

cd snort_inline

./configure --with-mysql && make && make install (kiểm tra câu hình và biên dịch snort, dấu && có ý nghĩa nếu câu lệnh trước nó thành công thì mới thực hiện câu lệnh đứng sau)

+ tạo password cho tài khoản root trong mysql

mysqladmin -u root password new_root_password

+ tạo database

mysql -u root -p

>create database snort;

+ cấp toàn quyền cho tài khoản snort trong cơ sở dữ liệu snort

grant all on snort.* to snortuser@localhost identified by 'snortpassword';

+ cấu hình các cảnh báo sẽ được xuất vào cơ sở dữ liệu mysql

#nano /etc/snort_inline/snort_inline.conf

chỉnh lại dòng

output database: log, mysql, user=snortuser password=snortpassword dbname=snort host=localhost

+sau khi cấu hình song snort_inline vậy là quá trình cài đặt snort_inline đã song. giờ muốn hiển thị và quản lý các cảnh bão một cách dễ dàng ta cài đặt thêm base và adodb.

+ cài đặt base

tar -xvzf base-1.4.5.tar.gz

mv /tmp/base-1.4.5 /var/www/html/base

+ cài đặt adodb

#tar -xvzf adodb490.tgz

#mv /tmp/adodb490 /var/www/html/adodb

+ cấu hình base

#mv /var/www/html/base/base_conf.php.dist /var/www/html/base/base_conf.php

cấu hình các biến như sau

$DBlib_path="./adodb";

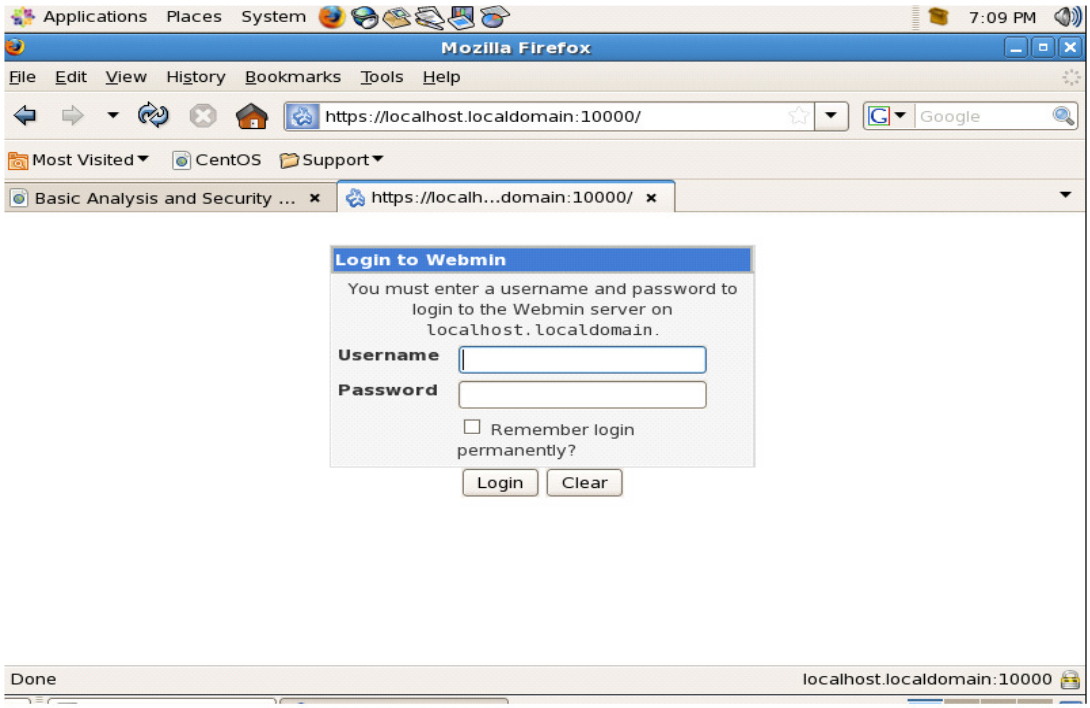
$DBtype="mysql";

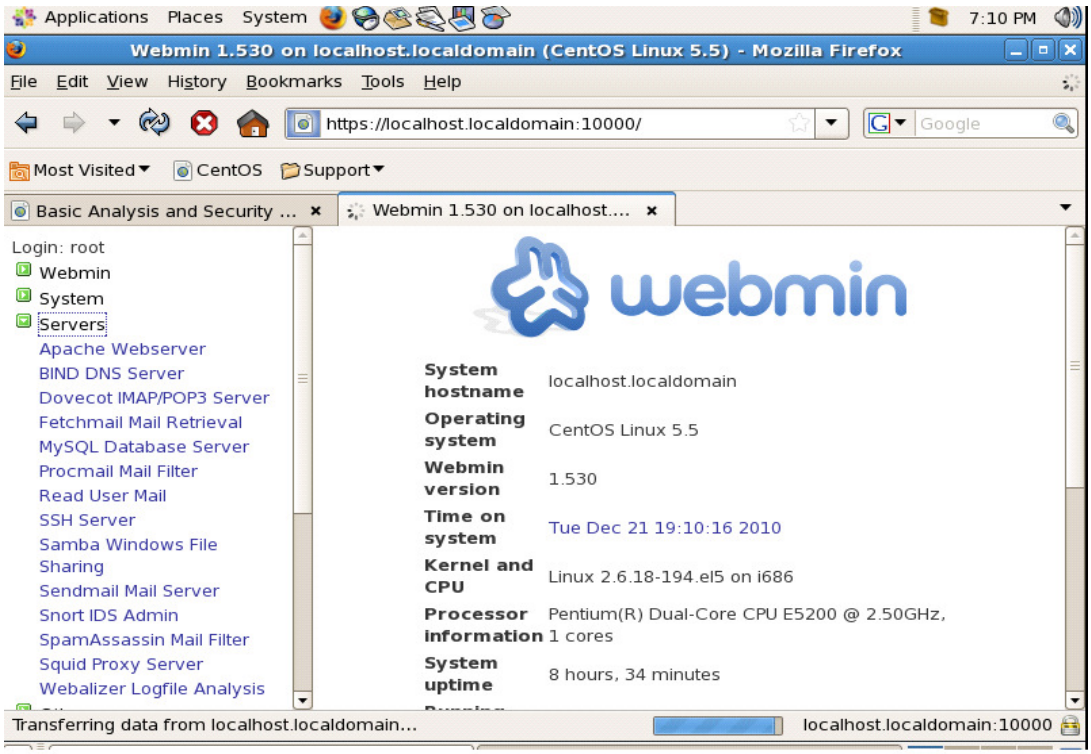
$alert_dbname = snort;

$alert_host = localhost;

$alert_port = "";
$alert_user = snortuser;
$alert_password = snortpassword;
$archive_dbname = snort;
$archive_host = localhost;
$archive_port = "";
$archive_user = snortuser;
$archive_password = snortpassword;

+ bây giờ cài đặt thêm cá gói sau để hiển thị ảnh trên base
#pear install --force Image_Color
#pear install --force Image_Canvas
#pear install --force Image_Graph
+ cài đặt thêm webmin để dễ dàng quản lý
# yum install webmin
sau khi cài đặt song webmin ta khởi động các dịch vụ
services httpd start
services mysql start
https://localhost.localdomain:10000





Tất cả các thông tin cấu hình của Snort được lưu trong file *snort.conf*. File snort.conf bao gồm 4 phần :

- Định nghĩa các biến xác định cấu hình mạng.
- Cấu hình môđun tiền xử lý.
- Cấu hình môđun kết xuất thông tin.
- Cấu hình bộ luật sử dụng.

Sau đây là nội dung cụ thể và ý nghĩa của các thông tin trong *snort.conf*.

3.1 Định nghĩa các biến

Snort cho phép định nghĩa các biến xác định các thông số mạng theo định dạng :

var : <name> <value>

Các biến này sẽ được sử dụng trong toàn bộ file cấu hình từ đó về sau. Ví dụ, nếu định nghĩa : "**var : MY_NET 192.168.1.0/24**" thì trong toàn bộ file config hay các file luật ký hiệu MY_NET sẽ được thay thế bằng giá trị 192.168.1.0/24.

3.2 Cấu hình môđun tiền xử lý

Các thông tin cấu hình cho môđun tiền xử lý được định nghĩa như sau :

preprocessor <name>:<options>

Quy định về name và options tùy thuộc vào từng plugin của môđun tiền xử lý. Ví dụ : cấu hình của plugin Portscan detection do Patrick Mullen viết như sau :

preprocessor portscan 192.168.0.1/24 5 7 /var/log/portscan.log

trong đó :

192.168.0.1/24 là mạng được theo dõi nguy cơ quét cổng.

5 là số lượng cổng truy cập đồng thời trong quá trình quét.

7 là thời gian theo dõi đê xác định nguy cơ quét cổng.

/var/log/portscan.log là file ghi lại log của quá trinh phát hiện.

3.3Cấu hình môđun kết xuất thông tin

Cấu hình cho môđun kết xuất thông tin cũng được định nghĩa tương tự cấu hình cho môđun tiền xử lý.

output <name>:<options>

Ví dụ, cấu hình cho Snort kết xuất thông tin cảnh báo ra syslog của một máy trong mạng như sau :

output alert_syslog: host=192.168.0.1:123, LOG_AUTH LOG_ALERT

Trong đó, host là ip và cổng syslog của máy được ghi, LOG_AUTH và LOG_ALERT là các loại log được ghi lại.

Để Snort kết xuất thông tin ra cơ sở dữ liệu, cấu hình như sau :

database: <log | alert>, <database type>, <parameter list>

Trong đó :

- log | alert : chỉ ra ghi lại thông tin gì? Log hay alert ?
- database type : Loại cơ sở dữ liệu. Snort hỗ trợ mysql, postgre sql và ms sql server.
- Parameter list : danh sách tham số phục vụ cho việc kết nối với cơ sở dữ liệu. Cụ thể tùy thuộc vào từng loại cơ sở dữ liệu cụ thể. Ví dụ, parameter list của mysql là như sau : dbname=snort user=snort host=localhost password=xyz.
-

3.4 Cấu hình bộ luật

Phần này chỉ ra các file luật được dùng. Cú pháp như sau :

include RULE_PATH/RULE_FILE

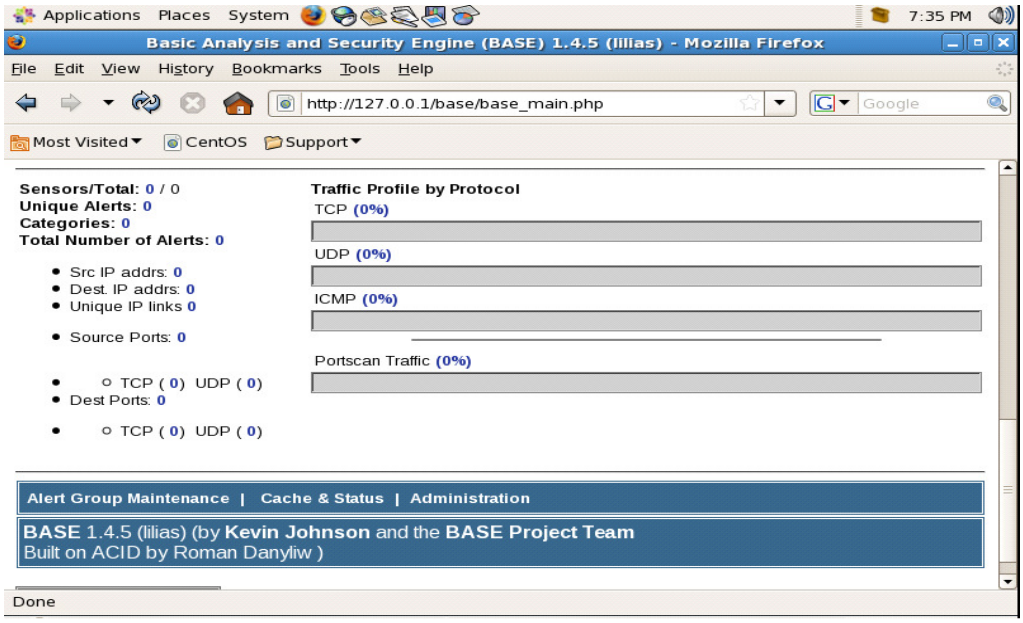
Ví dụ : yêu cầu Snort sử dụng luật phát hiện ddos bằng dòng lệnh sau :

include $RULE_PATH/ddos.rules

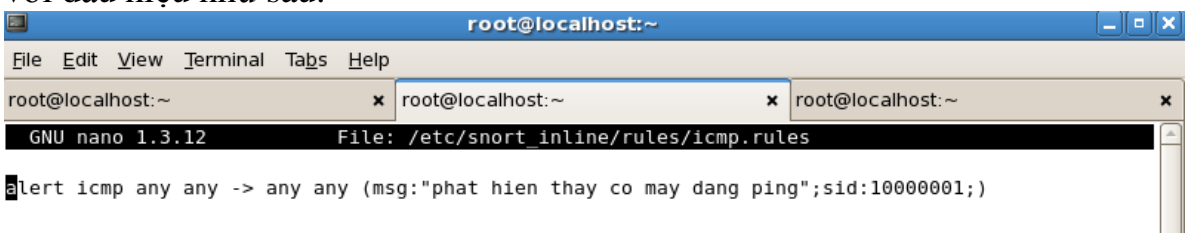
Trong đó, $RULE_PATH là biến chỉ đến thư mục chứa các file luật đã được định nghĩa trong phần định nghĩa các biến còn ddos.rules là file luật.

3.5 THỬ NGHIỆM KHẢ NĂNG PHẢN ỨNG CỦA SNORT IDS/IPS

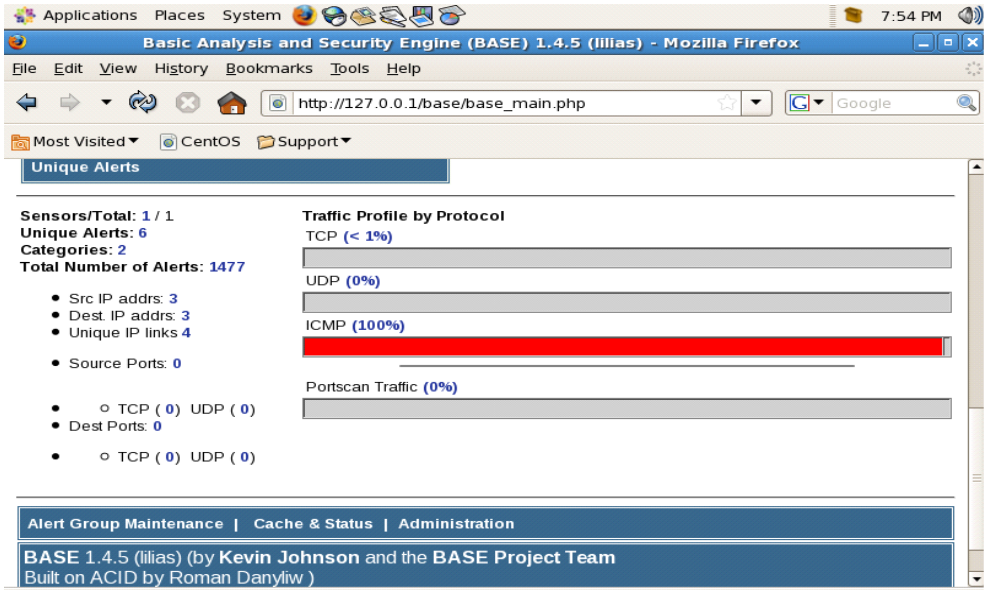
truy cập vào base http://127.0.0.1/base

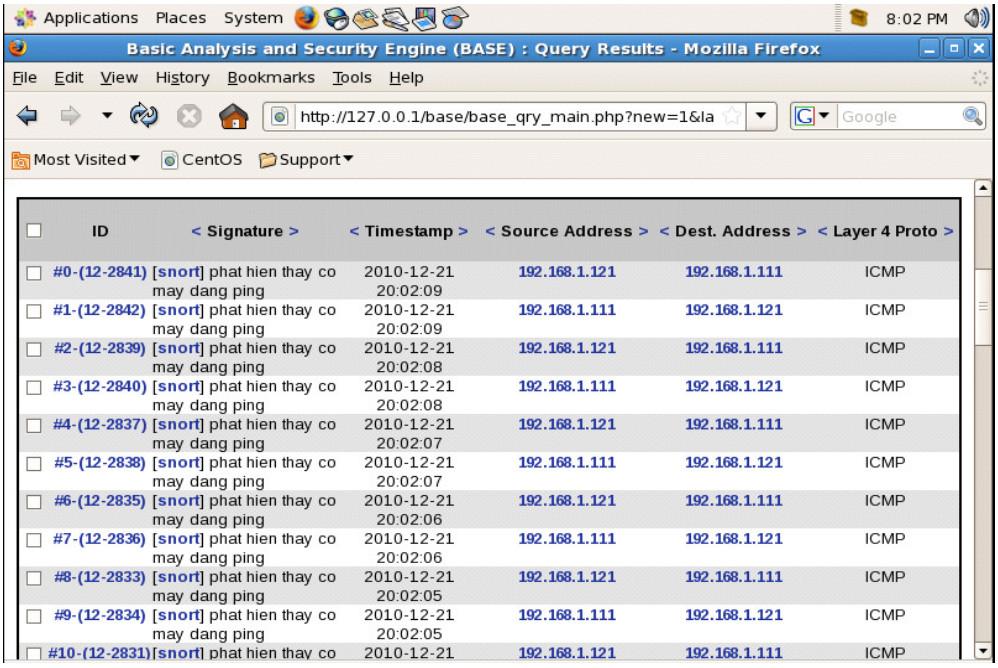


lúc này chưa có cảnh báo nào vì ta chưa khởi chạy snort. giả sử ta tạo một rules với dấu hiệu như sau:

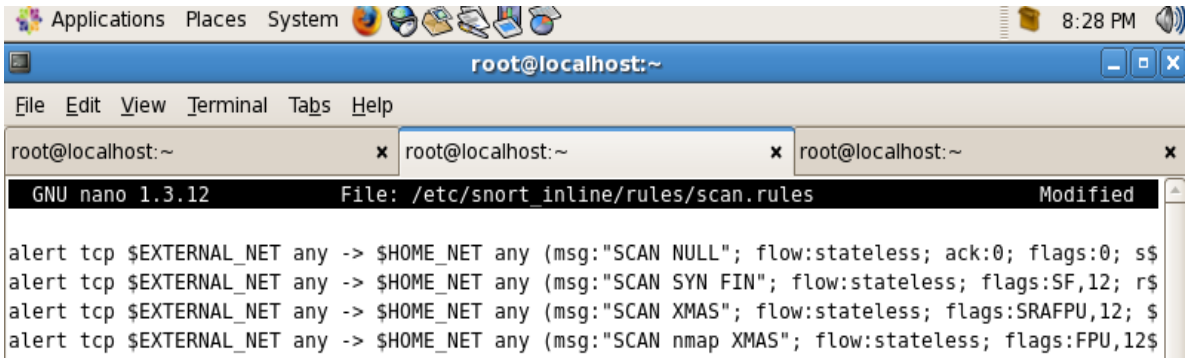


sau đó include nó vào file /etc/snort_inline/snort_inline.conf
và khởi chạy snort: # snort_inline -c /etc/snort_inline/snort_inline.conf -Q
rồi từ một máy khác ping đến với địa chỉ của máy ping là 192.168.1.121
và địa chỉ của máy IDS là 192.168.1.111 ta có kết quả sau.



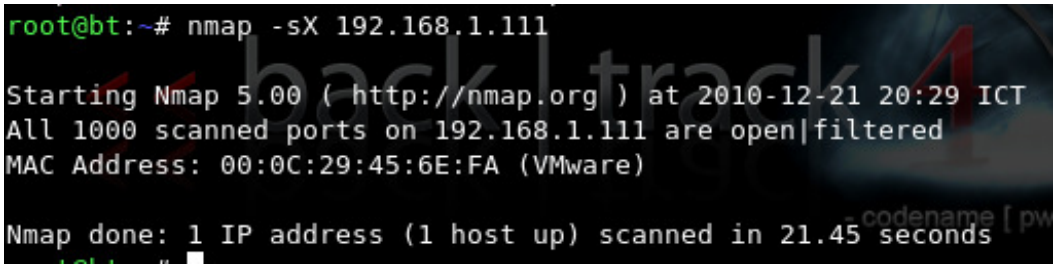


như vậy snort IDS đã hoạt động tốt, ta thử rules sau cho trường hợp phát hiện nmap scan cổng.

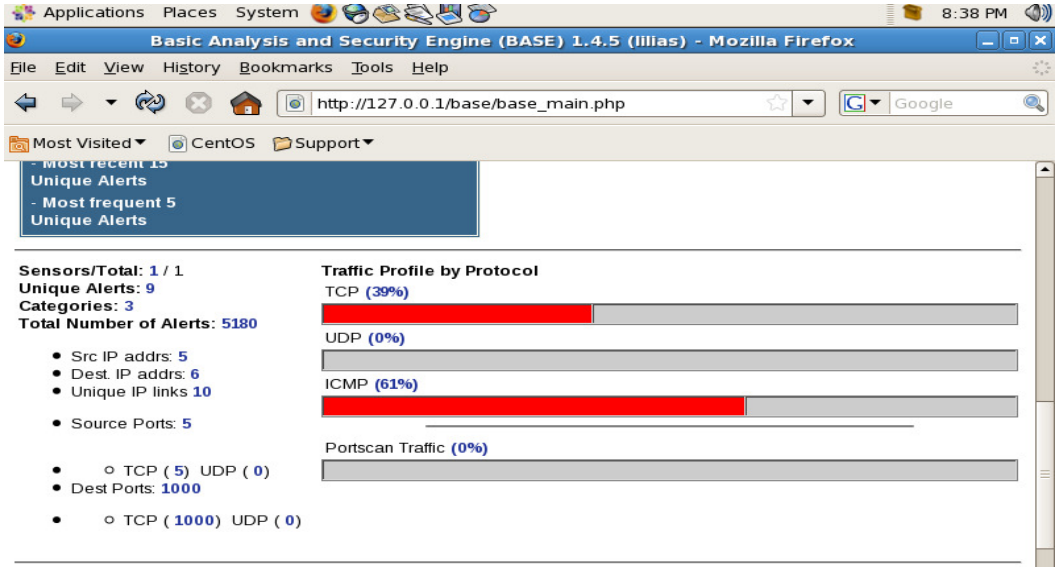


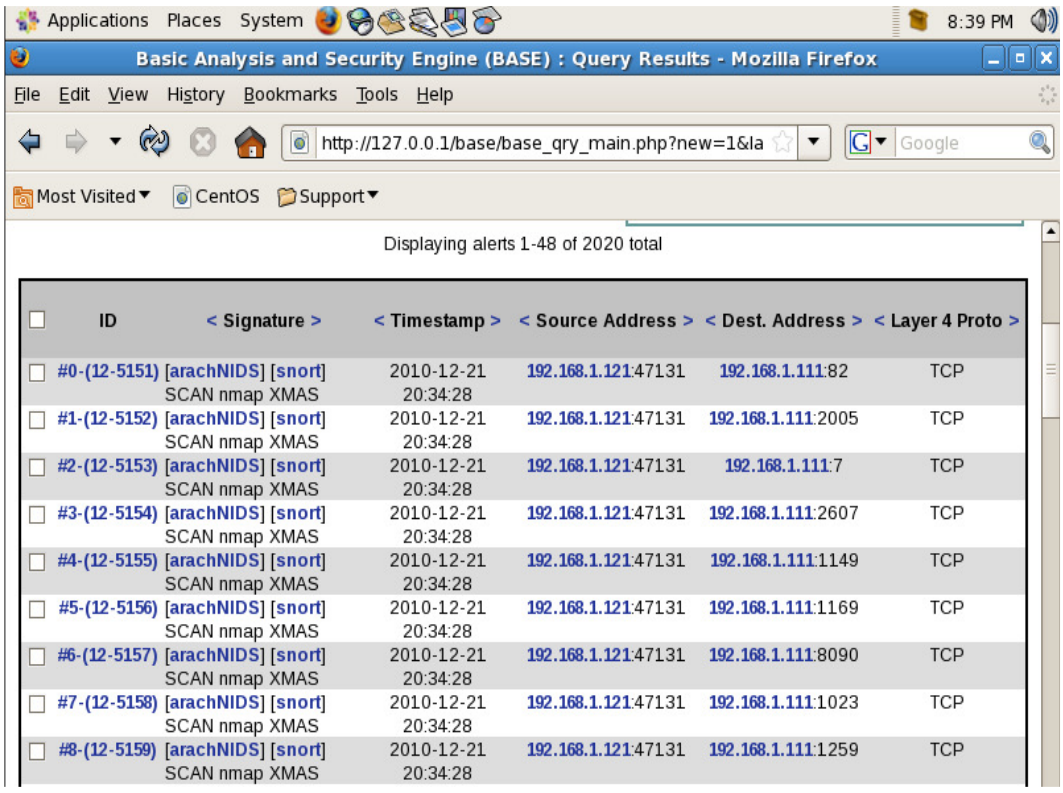
sau đó include scan.rules vào file /etc/snort_inline/snort_inline.conf

khởi động lại snort_inline. từ máy tấn công bật nmap và scan cổng ta nhận được kết quả. như vậy snort đã thể hiên là một IPS



vào mày snort và xem kết quả.





TÀI LIỆU THAM KHẢO

Tài liệu tiếng Việt :

[1]	Mạng máy tính và các hệ thống mở Tác giả : GSTS Nguyễn Thúc Hải NXB Giáo dục – 1999
[2]	Lập trình LINUX –tập 1 Tác giả : Nguyễn Phương Lan, Hoàng Đức Hải NXB Giáo Dục – 2001

Tài liệu tiếng Anh :

[3]	Intrusion Detection with Snort Tác giả : Rafeeq Rehman NXB Prentice Hall – 2003
[4]	Snort User Manual Tác giả : Martin Roesch, Chris Green The Snort Project – 2003
[5]	Snort 2.1 Intrusion Detection

Websites :

[6]	http://www.snort.org http://netfilter.org
[7]	http://snortinline.sourceforge.net http://hoclinux.net