Thiết kế và triển khai hệ thống phát hiện xâm nhập dựa trên kĩ thuật máy học

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Nguyễn Đức Thông
Phạm Ngọc Hiếu Minh
Trần Thanh Tín

Ngày 16 tháng 5 năm 2018

Mục lục

Danh sách hình vẽ

Chương 1

Giới thiệu

1.1 Đề tài

1.2 Đặt vấn đề

1.3 Mục tiêu

Chương 2

Lí thuyết

2.1 An toàn thông tin

2.2 Hệ thống phát hiện xâm nhập

2.2.1 Giới thiệu

IDS (Intrusion Detection System – hệ thống phát hiện xâm nhập) là một hệ thống giám sát lưu lượng mạng, có khả năng phát hiện các hoạt động khả nghi và cảnh báo cho hệ thống hoặc người quản trị mạng. IDS phát hiện dựa trên các dấu hiệu đặc biệt về các nguy cơ đã biết hoặc dựa trên việc so sánh lưu lượng mạng hiện tại với baseline. Trong đó baseline cho những hành vi bình thường được thiết lập bởi hồ sơ người dùng, máy chủ, hoặc hoạt động mạng trong quá trình học. Sau khi quá trình học kết thúc, những phát hiện bất thường được tìm ra sau quá trình thăm dò của IDS bị chệch đi so với baseline này. Phát hiện xâm nhập là một công việc khó khăn do sự phát triển nhanh chóng các của mạng lưới mạng, quá nhiều môi trường máy tính dẫn đến tính bất đồng bộ, nhiều giao thức mạng và sự phân loại đáng kể của các ứng dụng thông dụng và độc quyền. Hầu hết IDS sử dụng các dấu hiệu đặc biệt về nguy cơ xâm nhập đã biết(tương tự đối với cách mà các chương trình diệt virus hiện nay sử dụng để phát hiện và diệt virus), và sự khác biệt ứng xử của dấu hiệu xâm nhập so với những dấu hiệu đến từ người dùng thông thường.

IDS thường được đặt ở nhiều vị trí trong hệ thống mạng, trong đó, vị trí phía sau Firewall được nhiều nhà quản trị tin dùng nhất. ở vị trí này, IDS có nhiệm vụ phân tích các gói tin đã được Firewall thông qua, xác định các dấu hiệu đã được định nghĩa mà Firewall không thể kiểm tra hoặc ngăn chặn. Từ các dấu hiệu này, IDS cung cấp thông tin và đưa ra các cảnh báo cho người quản trị viên.

Với việc bảo vệ an toàn thông tin mạng ở một mức độ cao. Nhiều chuyên gia cho rằng IDS có giá trị giống như Firewall và VPN là ngăn ngừa các cuộc tấn công mà trong đó IDS cung cấp sự an toàn bằng cách trang bị cho bạn thông tin về các cuộc tấn công. Chính vì điều này, IDS có thể đáp ứng nhu cầu về an toàn hệ thống bằng cách cảnh báo về khả năng có thể xảy ra của các cuộc tấn công và đôi khi bên cạnh các cảnh báo đúng thì chúng cũng mắc phải một số nhầm lẫn dẫn đến các cảnh báo chưa chính xác.

2.2.2 Chức năng

Nhìn chung, bản thân IDS không có khả năng tự động ngăn chặn các cuộc tấn công, tuy nhiên, các phiên bản hiện đại của IDS như IPS (sẽ được nhắc đến ở những chương sau) đã có thể thực hiện nhiều vai trò hơn và có thể ngăn chặn các cuộc tấn công khi nó xảy ra. Thực tế, IDS dường như chỉ thông báo cho chúng ta biết rằng mạng đang có dấu hiệu bị tấn công và đang ở trong giai đoạn nguy hiểm.

Hệ thống phát hiện xâm nhập cho phép các tổ chức bảo vệ hệ thống mạng khỏi những đe dọa với việc gia tăng kết nối mạng và sự tin cậy của hệ thống thông tin, bổ sung cho những điểm yếu của hệ thống khác... Sau đây là một vài lý do mà một hệ thống nên sử dụng IDS:

- Bảo vệ tính toàn vẹn dữ liệu, đảm bảo sự nhất quán của dữ liệu trong hệ thống. Các biện pháp đưa ra có khả năng ngăn chặn được sử thay đổi bất hợp pháp hoặc phá hoại dữ liệu.
- Bảo vệ tính riêng tư, nghĩa là đảm bảo cho người sử dụng khai thác tài nguyên của hệ thống theo đúng chức năng nhiệm vụ đã được phân quyền, ngăn chặn được sự truy cập thong tin bất hợp pháp.
- Bảo vệ tính bí mật, giữ cho thông tin không bị để lộ ra ngoài phạm vi cho phép.
- Bảo vệ tính khả dụng, nghĩa là hệ thống luôn sẵn sàng thực hiện yêu cầu truy cập thông tin của người dùng hợp pháp.
- Cung cấp thông tin về sự truy cập, đưa ra các chính sách đối phó, khôi phục, sửa chữa...

Về cơ bản, hệ thống IDS có thể giúp chúng ta ngăn ngừa các sự kiện tấn công trước khi nó xảy ra, cung cấp một số các giải pháp cho mạng và máy chủ, thậm chí cũng có thể hoạt động như một chuông báo động. Tuy nhiên chức năng chính của nó là thông báo cho người quản trị biết về các sự kiện có liên quan đến an ninh hệ thống đang sắp sửa xảy ra trong mạng và hệ thống mà người quản trị hiện đang kiểm soát.

Thông thường, để phân loại IDS (IPS), người ta thường dựa vào đặc điểm của nguồn dữ liệu thu thập được. Trong trường hợp này, các hệ thống IDS được chia thành hai loại phổ biến như sau:

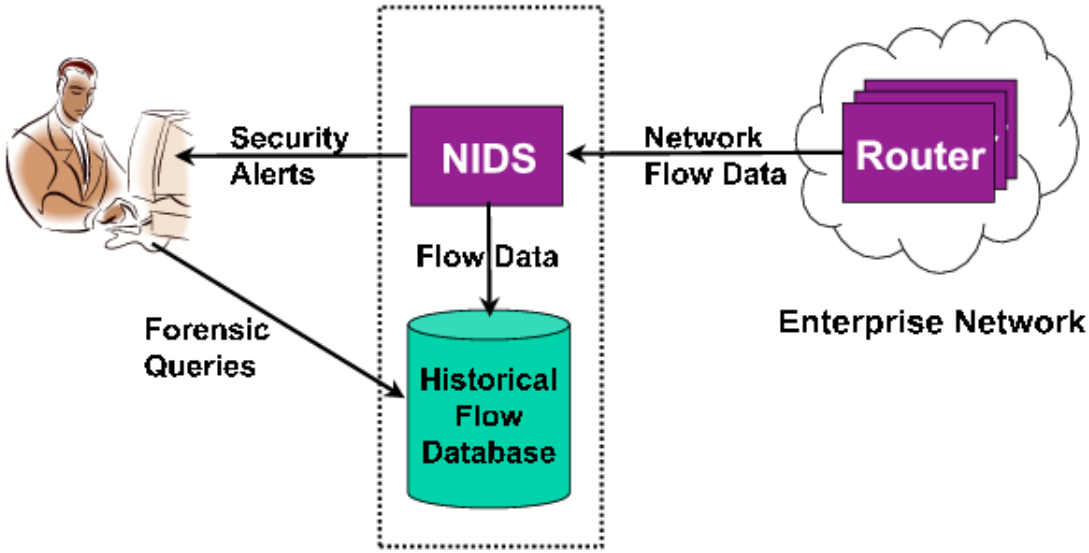
- Host-Based IDS (HIDS): Sử dụng dữ liệu kiểm tra từ một máy trạm đơn để phát hiện xâm nhập.
- Network-Based IDS (NIDS): Sử dụng dữ liệu trên toàn bộ lưu thông mạng, cùng với dữ liệu kiểm tra từ một hoặc một vài máy trạm để phát hiện xâm nhập.

2.2.3 NIDS

Hệ thống IDS dựa trên mạng sử dụng bộ dò và bộ cảm biến được cài đặt trên toàn mạng. Những bộ dò này theo dõi trên mạng với mục đích tìm kiếm những lưu lượng mạng khớp với những dấu hiệu được mô tả, định nghĩa từ trước. Những bộ cảm biến thu nhận và phân tích lưu lượng trong hệ thống thời gian thực. Khi nhận được

mẫu lưu lượng hay dấu hiệu, bộ cảm biến gửi cảnh báo đến trạm quản trị và có thể được cấu hình nhằm tìm ra biện pháp ngăn chặn những xâm nhập xa hơn. NIDS là tập hợp nhiều cảm biến được cài đặt ở trên toàn mạng nhằm theo dõi những gói tin trong mạng, so sánh với mạng được định nghĩa để phát hiện đó là tấn công hay không.

NIDS được đặt giữa hệ thống mạng bên trong và hệ thống mạng bên ngoài để giám sát toàn bộ lưu lượng vào ra. Có thể là một thiết bị phần cứng riêng biệt được thiết lập sẵn hay phần mềm cài đặt trên máy tinh, chủ yếu dùng để đo lưu lượng mạng được sử dụng.



Hình 2.1: Luồng hoạt động của NIDS

NIDS giám sát toàn bộ mạng con của nó bằng cách lắng nghe tất cả các luồng dữ liệu trên mạng con đó. (Nó thay đổi chế độ hoạt động của card mạng NIC vào trong chế độ promisuous). Bình thường, một NIC hoạt động ở chế độ nonpromisuous nghĩa là nó chỉ nhận các gói tin mà có địa chỉ MAC khớp với địa chỉ của nó, các gói tin khác sẽ không nhận, hoặc không xử lý và bị loại bỏ. Để giám sát tất cả các đường truyền trong mạng con, NIDS phải chấp nhận tất cả các gói tin và chuyển chúng tới ngăn xếp để xử lý. Do vậy nó sẽ phải cài đặt chế độ hoạt động cho card mạng là promisuous.

Ưu điểm

- Chi phí triển khai thấp
- 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 hầu như không bỏ sót các dấu hiệu xuất phát từ đây.
- Khó xóa bỏ dấu vết (evidence)
- Phát hiện và đối phó kịp thời
- Có tính độc lập cao

Nhược điểm

- Bị hạn chế với Switch
- Hạn chế về hiệu năng
- Tăng băng thông mạng
- Một hệ thống NIDS thường gặp khó khăn trong việc xử lý các cuộc tấn công trong một phiên được mã hóa
- Một hệ thống NIDS cũng gặp khó khăn khi phát hiện các cuộc tấn công mạng từ các gói tin phân mảnh

2.2.4 HIDS

Host-based IDS tìm kiếm dấu hiệu của xâm nhập trong một máy chủ cục bộ; thường sử dụng các cơ chế kiểm tra và phân tích các thông tin được lưu lại. Chủ yếu tìm kiếm các hoạt động bất thường như đăng nhập, truy cập tập tin không được cấp phép, 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 ở quyền bậc cao chỉ có thể thành công thông qua lệnh su-select user, như vậy những hành vi đăng nhập liên tục vào tài khoản root có thể được coi là một cuộc tấn công.

Ưu điểm

- Xác định được kết quả của cuộc tấn công
- Khả năng giám sát các hoạt động cụ thể của hệ thống
- Phát hiện các hoạt động xâm nhập mà NIDS không phát hiện được: chẳng hạn kẻ tấn công sử dụng bàn phím xâm nhập vào một máy chủ sẽ không bị NIDS phát hiện.
- Không yêu cầu thêm phần cứng

Nhược điểm

- Khó quản trị
- Nguồn thông tin phân tích không an toàn
- Hệ thống host-based khá đắt
- Chiếm tài nguyên hệ thống

2.3 Khai phá dữ liệu

2.3.1 Khái niệm

Là quá trình tính toán để tìm ra các mẫu trong các bộ dữ liệu lớn liên quan đến các phương pháp tại giao điểm của máy học, thống kê và các hệ thống cơ sở dữ liệu. Đây là một lĩnh vực liên ngành của khoa học máy tính. Mục tiêu tổng thể của quá trình khai thác dữ liệu là trích xuất thông tin từ một bộ dữ liệu và chuyển nó thành một cấu trúc dễ hiểu để sử dụng tiếp. Ngoài bước phân tích thô, nó còn liên quan tới cơ sở dữ liệu và các khía cạnh quản lý dữ liệu, xử lý dữ liệu trước, suy xét mô hình và suy luận thống kê, các thước đo thú vị, các cân nhắc phức tạp, xuất kết quả về các cấu trúc được phát hiện, hiện hình hóa và cập nhật trực tuyến.

2.3.2 Các bước trong quá trình khai phá

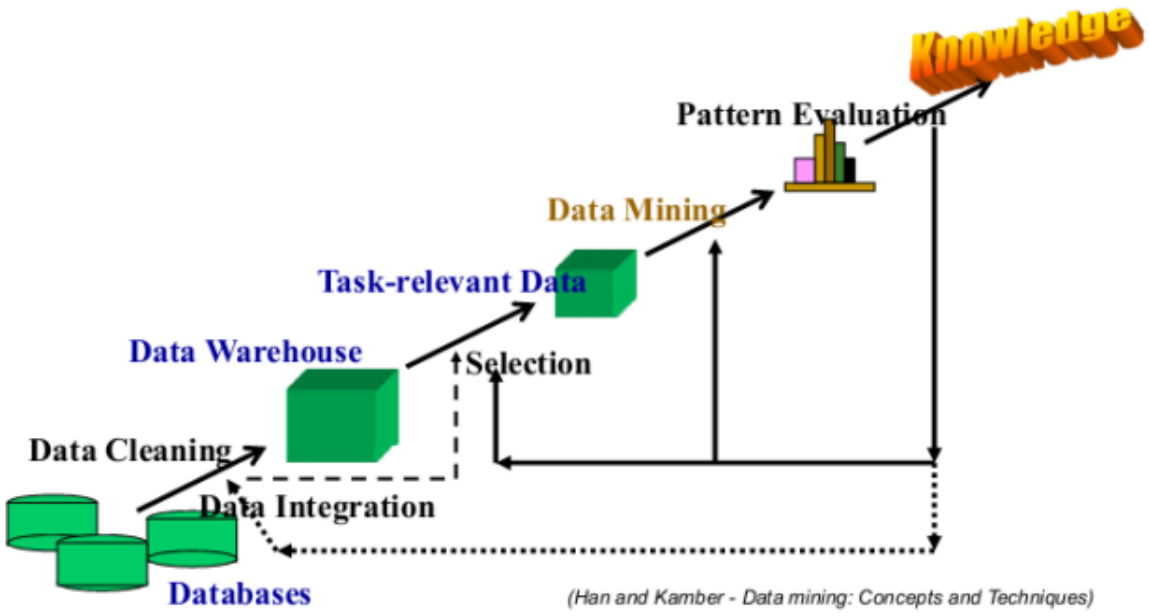
Quá trình được thực hiện qua 9 bước:

- Tìm hiểu lĩnh vực của bài toán (ứng dụng): Các mục đích của bài toán, các tri thức cụ thể của lĩnh vực.
- Tạo nên (thu thập) một tập dữ liệu phù hợp.
- Làm sạch và tiền xử lý dữ liệu.
- Giảm kích thức của dữ liệu, chuyển đổi dữ liệu: Xác định thuộc tính quan trọng, giảm số chiều (số thuộc tính), biểu diễn bất biến.
- Lựa chọn chức năng khai phá dữ liệu: Phân loại, gom cụm, dự báo, sinh ra các luật kết hợp.
- Lựa chọn/ Phát triển (các) giải thuật khai phá dữ liệu phù hợp. Tiến hành khai phá dữ liệu.
- Đánh giá mẫu thu được và biểu diễn tri thức: Hiển thị hóa, chuyển đổi, bỏ đi các mẫu dư thừa,...
- Sử dụng tri thức được khai phá.

Quá trình khám phá tri thức theo cách nhìn của giới nghiên cứu về các hệ thống dữ liệu và kho dữ liệu về quá trình khám phá tri thức

- Chuẩn bị dữ liệu (data preparation), bao gồm các quá trình làm sạch dữ liệu (data cleaning), tích hợp dữ liệu (data integration), chọn dữ liệu (data selection), biến đổi dữ liệu (data transformation).
- Khai thác dữ liệu (data mining): xác định nhiệm vụ khai thác dữ liệu và lựa chọn kỹ thuật khai thác dữ liệu. Kết quả cho ta một nguồn tri thức thô.
- Đánh giá (evaluation): dựa trên một số tiêu chí tiến hành kiểm tra và lọc nguồn tri thức thu được.
- Triển khai (deployment).

- Quá trình khai thác tri thức không chỉ là một quá trình tuần tự từ bước đầu tiên đến bước cuối cùng mà là một quá trình lặp và có quay trở lại các bước đã qua.



Hình 2.2: Quá trình khai phá tri thức

2.3.3 Ứng dụng của khai phá dữ liệu

- Kinh tế - ứng dụng trong kinh doanh, tài chính, tiếp thị bán hàng, bảo hiểm, thương mại, ngân hàng, ... Đưa ra các bản báo cáo giàu thông tin; phân tích rủi ro trước khi đưa ra các chiến lược kinh doanh, sản xuất; phân loại khách hàng từ đó phân định thị trường, thị phần; ...
- Khoa học: Thiên văn học – dự đoán đường đi các thiên thể, hành tinh, ...; Công nghệ sinh học – tìm ra các gen mới, cây con giống mới, ...; ...
- Web: các công cụ tìm kiếm.

2.4 Máy học

Chương 3

Công nghệ tiếp cận

3.1 Snort 3

3.1.1 Giới thiệu

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.

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.

3.1.2 Snort 3.0

Snort 3.0 là một phiên bản cập nhật của Hệ thống Ngăn chặn xâm nhập Snort (IPS) có tính năng thiết kế mới được cung cấp từ Snort 2.X với thông lượng lớn hơn, phát hiện tốt hơn, khả năng mở rộng và khả năng sử dụng tốt hơn. Một số tính năng chính của Snort 3.0 là:

- Hỗ trợ nhiều luồng xử lý gói
- Sử dụng cấu hình được chia sẻ và bảng thuộc tính
- Dịch vụ tự động phát hiện cho cấu hình không cần cổng
- Thiết kế mô-đun
- Nền tảng plugin với hơn 200 plugin
- Cấu hình bộ nhớ mở rộng hơn

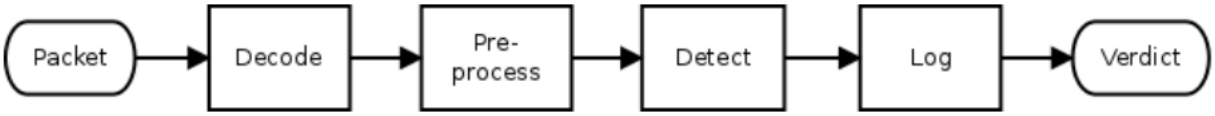
- Cấu hình LuaJIT, logger, và các tùy chọn quy tắc
- Hỗ trợ Hyperscan
- Xử lý TCP được ghi đè
- Trình phân tích cú pháp và cú pháp quy tắc mới
- Các quy tắc dịch vụ như http cảnh báo
- Quy tắc Bộ đệm "dính"
- Quy tắc SO tốt hơn
- Kiểm soát HTTP mới
- Hiệu suất giám sát mới
- Hồ sơ thời gian và không gian mới
- Độ trễ giám sát và thực thi mới
- Piglets để tạo điều kiện thử nghiệm thành phần
- Kiểm tra các sự kiện
- Automake và Cmake
- Tự động tạo tài liệu tham khảo

Các tính năng đang được phát triển

- Sử dụng mạng lưới network đã chia sẻ
- Hỗ trợ phần cứng giảm tải để tăng tốc mẫu nhanh
- Cung cấp hỗ trợ cho DPDK và ODP
- Hỗ trợ pipelining xử lý gói tin
- Hỗ trợ chế độ proxy
- Hỗ trợ đa tennant
- Gia tăng tải lại
- Tuần tự hóa hiệu suất của dữ liệu và sự kiện mới
- Xử lý quy tắc nâng cao
- Hỗ trợ Windows
- Phát hiện bất thường
- và hơn thế nữa!

3.1.3 Kiến trúc

Snort 2.x



Hình 3.1: Kiến trúc Snort 2.x

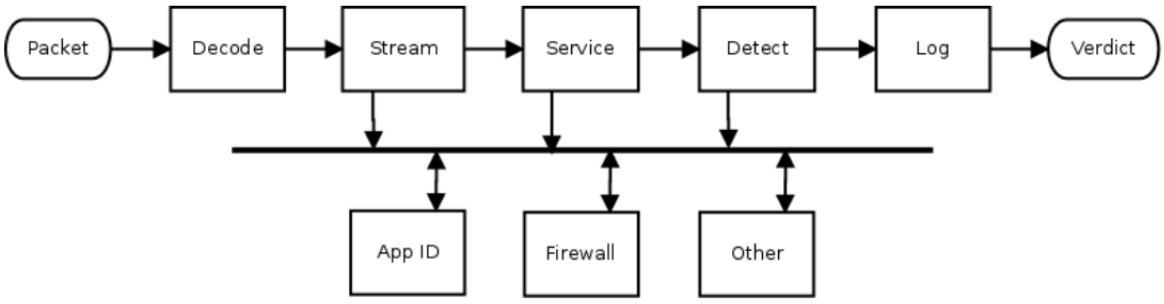
Bước tiền xử lý trong Snort 2 có khả năng cấu hình cao. Các bộ tiền xử lý tùy ý có thể được nạp tự động khi khởi động, được cấu hình trong snort.conf, và sau đó được thực thi khi chạy. Về cơ bản, các bộ tiền xử lý được đưa vào danh sách được lặp lại cho mỗi gói. Các phiên bản gần đây đã tinh chỉnh một số danh sách xử lý, nhưng kiến trúc cơ bản giống nhau đã cho phép Snort 2 phát triển từ một trình thám thính, không có tiền xử lý,cho đến một IPS chính thức, với rất nhiều tiền xử lý.

Trong khi phương pháp "list of plugins"("danh sách plugin") này có tính linh hoạt đáng kể, nó cản trở sự phát triển trong tương lai khi luồng dữ liệu từ một bộ tiền xử lý tới bộ xử lý tiếp theo phụ thuộc vào điều kiện lưu lượng dữ liệu, tình huống chung với các tính năng nâng cao như nhận dạng ứng dụng. Trong trường hợp này, một bộ tiền xử lý như HTTP có thể trích xuất và chuẩn hóa dữ liệu mà cuối cùng không được sử dụng, hoặc appID có thể liên tục kiểm tra dữ liệu không có sẵn.

Callbacks giúp thoát ra khỏi bộ tiền xử lý đang quá tải. Đây là nơi mà một bộ tiền xử lý cung cấp một bộ xử lý khác có chức năng gọi khi có sẵn một số dữ liệu nhất định. Snort đã bắt đầu thực hiện phương pháp này để chuyển một số dữ liệu tiền xử lý HTTP và SIP tới appID. Tuy nhiên, nó vẫn là một tính năng ngoại vi và vẫn yêu cầu dữ liệu mà không được xử lý

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.

Snort 3.x



Hình 3.2: Kiến trúc Snort 3.x

Một trong những mục tiêu của Snort 3 là cung cấp một khuôn khổ linh hoạt hơn để xử lý gói tin bằng cách thực hiện một phương pháp hướng sự kiện. Khác là để

sản xuất dữ liệu chỉ khi cần thiết để giảm thiểu chuẩn hóa đắt tiền. Tuy nhiên, xử lý gói cơ bản cung cấp chức năng rất giống nhau.

Các bước xử lý cơ bản Snort 3 tương tự như Snort 2 như đã thấy trong sơ đồ. Bước tiền xử lý sử dụng các loại kiểm duyệt cụ thể thay vì một danh sách tổng quát, nhưng thủ tục cơ bản bao gồm giải mã gói không trạng thái, khôi phục luồng TCP và phân tích dịch vụ cụ thể trong cả hai trường hợp. (Snort 3 cung cấp các móc cho các kiểm duyệt tùy ý, nhưng chúng không tập trung vào xử lý luồng cơ bản và không được hiển thị.)

Tuy nhiên, Snort 3 cũng cung cấp một cơ chế linh hoạt hơn các hàm gọi lại. Bằng cách sử dụng các sự kiện kiểm tra, một kiểm duyệt viên có thể cung cấp dữ liệu mà các kiểm duyệt viên khác có thể xử lý. Điều này được gọi là mẫu quan sát hoặc mẫu đăng ký xuất bản.

Lưu ý rằng dữ liệu không thực sự được xuất bản. Thay vào đó, quyền truy cập vào dữ liệu được xuất bản và điều đó có nghĩa là người đăng ký có thể truy cập (các) phiên bản thô hoặc chuẩn hóa nếu cần. Việc chuẩn hóa chỉ được thực hiện trên lần truy cập đầu tiên và các truy cập tiếp theo nhận được dữ liệu đã chuẩn hóa trước đó. Điều này dẫn đến việc xử lý kịp thời (JIT).

3.1.4 Cấu hình

Cấu hình Snort hiệu quả được thực hiện thông qua môi trường, dòng lệnh, tệp cấu hình Lua và một bộ quy tắc.

Lưu ý rằng khả năng tương thích ngược với Snort 2 đã được hy sinh để có được chức năng mới và được cải thiện. Trong khi Snort 3 sử dụng một số cơ sở mã Snort 2, rất nhiều đã thay đổi. Cấu hình của Snort 3 được thực hiện bằng Lua, vì vậy ký tự cũ của bạn sẽ không hoạt động như cũ. Quy tắc vẫn dựa trên văn bản nhưng với các chỉnh sửa cú pháp, vì vậy các quy tắc 2.X của bạn phải được sửa. Tuy nhiên, snort2lua sẽ giúp bạn chuyển đổi conf và quy tắc của bạn sang định dạng mới.

Môi trường

LUA_PATH phải được đặt dựa trên cài đặt của bạn:

```
LUA_PATH=$install_prefix/include/snort/lua/\?.lua\;\;
```

SNORT_LUA_PATH phải được thiết lập để tải các tệp cấu hình phụ nếu bạn sử dụng snort.lua mặc định. Ví dụ:

```
export SNORT\_LUA\_PATH=$install\_prefix/etc/snort
```

Dòng lệnh

Một dòng lệnh đơn giản có thể trông giống như sau:

```
snort -c snort.lua -R cool.rules -r some.pcap -A cmg
```

Để hiểu điều đó, bạn có thể bắt đầu bằng cách chạy snort mà không có đối số bằng cách chạy snort –help. Trợ giúp cho tất cả các tùy chọn cấu hình và quy tắc có sẵn thông qua một dòng lệnh phù hợp. Trong trường hợp này:

-c snort.lua là tệp cấu hình chính. Đây là một kịch bản lệnh Lua được thực hiện khi được nạp.

-R cool.rules chứa một số quy tắc phát hiện. Bạn có thể viết của riêng bạn hoặc có được chúng từ Talos (nguyên tắc 3.0 nguyên bản chưa có sẵn từ Talos, do

đó bạn phải chuyển đổi chúng với snort2lua). Bạn cũng có thể đặt các quy tắc của mình trực tiếp trong tệp cấu hình của mình.

-r some.pcap yêu cầu Snort đọc lưu lượng mạng từ tệp chụp gói tin đã cho. Thay vào đó, bạn có thể sử dụng -i eth0 để đọc từ giao diện trực tiếp. Có nhiều tùy chọn khác có sẵn quá tùy thuộc vào DAQ bạn sử dụng.

-A cmg cho biết các sự kiện xâm nhập đầu ra ở định dạng "cmg", có các chi tiết tiêu đề cơ bản theo sau bởi tải trọng trong hex và văn bản.

Tập tin cấu hình

File cấu hình cho phép bạn hoàn toàn kiểm soát cách Snort xử lý các gói tin. Bắt đầu với snort.lua mặc định được bao gồm trong bản phân phối vì có chứa một số thành phần chính. Lưu ý rằng hầu hết các cấu hình trông giống như:

```
stream = { }
```

Điều này có nghĩa là cho phép mô-đun luồng sử dụng các giá trị mặc định nội bộ. Để xem những thứ đó là gì, bạn có thể chạy:

```
$ snort --help-config stream
```

Snort được tổ chức thành một bộ sưu tập các mô-đun dựng sẵn và plugin. Nếu một module có các tham số, nó được cấu hình bởi một bảng Lua có cùng tên. Ví dụ, chúng ta có thể thấy những gì module đang hoạt động cung cấp với lệnh này:

```
$ snort --help-module active
```

Quy tắc

Quy tắc xác định những gì Snort đang tìm kiếm. Chúng có thể được đặt trực tiếp trong tệp cấu hình Lua của bạn với mô đun ips, trên dòng lệnh với –lua hoặc trong các tệp bên ngoài. Nói chung, bạn sẽ có nhiều quy tắc thu được từ nhiều nguồn khác nhau như Talos và tải các tập tin bên ngoài là cách để đi vì vậy chúng tôi sẽ tóm tắt ở đây. Thêm vào cấu hình Lua của bạn:

```
ips = { include = 'rules.txt' }
```

để tải tệp quy tắc bên ngoài có tên rules.txt. Bạn chỉ có thể chỉ định một tệp theo cách này nhưng quy tắc tệp có thể bao gồm các tệp quy tắc khác với câu lệnh include. Ngoài ra, bạn có thể tải các quy tắc như:

```
$ sort -c snort.lua –R rules.txt
```

3.1.5 Mở rộng tính năng

Plugin

Các plugin có một API được liên kết được xác định cho từng loại, tất cả đều có chung một tiêu đề, được gọi là BaseApi. Một thư viện động làm cho các plugin của nó có sẵn bằng cách xuất khẩu biểu tượng đó là snort_plugins, một mảng null kết thúc của các con trỏ BaseApi.

BaseApi bao gồm loại, tên, phiên bản API, phiên bản plugin và các con trỏ hàm để xây dựng và hủy một Mô-đun. API cụ thể thêm nhiều dữ liệu và chức năng khác cho vai trò đã cho của chúng.

Mô đun

Nếu chúng ta định nghĩa một Inspector, gadget nó có thể được cấu hình trong snort.lua như sau:

```
gadget =
{
    brain = true,
    claw = 3
}
```

Khi bảng tiện ích được xử lý, Snort sẽ tìm kiếm một mô-đun được gọi là tiện ích. Nếu Module đó có một API liên quan, nó sẽ được sử dụng để cấu hình một thể hiện mới của plugin. Trong trường hợp này, một GadgetModule sẽ được khởi tạo, não và móng sẽ được thiết lập, và cá thể Module sẽ được truyền cho hàm tạo GadgetInspector.

Module có ba phương thức ảo chính:

- **begin()** - được gọi khi Snort bắt đầu xử lý bảng Lua được liên kết. Đây là một nơi để phân bổ bất kỳ dữ liệu cần thiết và thiết lập mặc định.
- **set()** - được gọi để đặt từng thông số sau khi xác thực.
- **end()** - được gọi khi Snort kết thúc xử lý bảng Lua được liên kết. Đây là nơi kiểm tra tính toàn vẹn bổ sung của các tham số liên quan nên được thực hiện.

Mô-đun được cấu hình sẽ chuyển đến trình tạo plugin để lấy dữ liệu cấu hình từ Mô-đun. Đối với các cấu hình không quan trọng, mô hình làm việc là Mô-đun đưa một con trỏ tới dữ liệu đã được cấu hình cho cá thể của plugin có quyền sở hữu.

Lưu ý rằng có tối đa một phiên bản của một Mô-đun cụ thể, ngay cả khi nhiều phiên bản plugin được tạo sử dụng Mô-đun đó. (Nhiều trường hợp yêu cầu cấu hình ràng buộc Snort.)

Inspector

Có một số loại kiểm duyệt, xác định kiểm duyệt nào được thực hiện khi:

- IT_BINDER - xác định kiểm duyệt nào áp dụng cho các luồng đã cho
- IT_WIZARD - xác định kiểm duyệt dịch vụ nào sẽ sử dụng nếu không ràng buộc rõ rang
- IT_PACKET - được sử dụng để xử lý tất cả các gói trước khi xử lý phiên và dịch vụ (ví dụ: chuẩn hóa)
- IT_NETWORK - xử lý các gói dịch vụ w / o (ví dụ: arp_spoof, back_orifice)
- IT_STREAM - để theo dõi lưu lượng, chống phân mảnh ip và khôi phục tcp
- IT_SERVICE - cho http, ftp, telnet, v.v.
- IT_PROBE - xử lý tất cả các gói sau tất cả các gói ở trên (ví dụ: perf_monitor, port_scan)

Codec

Snort Codec giải mã các gói dữ liệu thô. Các Codec này bây giờ hoàn toàn có thể cắm được; hầu như mọi Snort Codec đều có thể được xây dựng động và được thay thế bằng Codec tùy biến, thay thế. Bản chất có thể cắm được cũng giúp việc xây dựng các Codec mới cho các giao thức dễ dàng hơn mà không cần phải chạm vào mã Snort.

Bước đầu tiên trong việc tạo một Codec là định nghĩa lớp và giao thức của nó. Mỗi Codec phải kế thừa từ lớp Snort Codec được định nghĩa trong "framework / codec.h".

IPS action

Các plugin thực thi một hành động dựng sẵn trong API được sử dụng để xác định phán quyết. (Ngược lại, hành động được tạo sẵn không có sự liên kết với chức năng plugin.)

Piglet Test Harness

Để hỗ trợ phát triển plugin, chế độ thử nghiệm được gọi là chế độ "piglet"được cung cấp. Với chế độ Piglet, bạn có thể gọi các phương thức riêng cho một plugin cụ thể. Các xét nghiệm heo con được xác định là kịch bản Lua. Mỗi kịch bản thử nghiệm heo con xác định một thử nghiệm cho một plugin cụ thể.

3.2 NSL-KDD

3.2.1 Giới thiệu

3.2.2 Đặc tính

3.3 Thuật toán KMeans

3.4 Scikit

3.4.1 Giới thiệu

Scikit-learn (viết tắt là **sklearn**) là một thư viện mã nguồn mở trong ngành machine learning, rất mạnh mẽ và thông dụng với cộng đồng Python, được thiết kế trên nền NumPy và SciPy. Scikit-learn chứa hầu hết các thuật toán machine learning hiện đại nhất, đi kèm với comprehensive documentations. Điểm mạnh của thư viện này là nó được sử dụng phổ biến trong academia và industry, do đó nó luôn được updated và có một very active user community.

Chương 4

Thử nghiệm và kết quả

4.1 Thiết lập dữ liệu và môi trường thử nghiệm

4.2 Kết quả và nhận xét

Chương 5

Kết luận

5.1 Nhận định

5.2 Hướng phát triển

Tài liệu tham khảo