



PHIẾU GIAO ĐỀ TÀI KHÓA LUẬN TỐT NGHIỆP

1. Họ và tên sinh viên/ nhóm sinh viên được giao đề tài (số trong nhóm: 2)

- (1) Đặng Ngọc Thành MSSV: 081649 Khóa: 2008-2011
(2) Quan Thị Trọng MSSV: 081684 khóa: 2008-2011

Chuyên ngành : Mạng máy tính Khoa : Khoa Học – Công Nghệ

2. Tên đề tài : Xây dựng hệ thống IDS – Snort trên hệ điều hành Linux

3. Các dữ liệu ban đầu:

Snort được xây dựng với mục đích phát hiện xâm nhập vào hệ thống. Snort có khả năng phát hiện một số lượng lớn các kiểu thăm dò, xâm nhập khác nhau như : buffer overflow, ICMP, virus,... Snort là phần mềm open source cung cấp cho nhà quản trị các thông tin cần thiết để xử lý các sự cố khi bị xâm nhập.

4. Các yêu cầu đặc biệt:

Hiểu được khái niệm, cách hoạt động của IDS – Snort. Cài đặt, cấu hình Snort trên hệ điều hành Ubuntu. Kiểm chứng kết quả đạt được sau khi cài đặt thành công.

5. Kết quả tối thiểu phải có:

Nắm rõ khái niệm về IDS và Snort

Cài đặt, cấu hình thành công Snort trên hệ điều hành Ubuntu

Ngày giao đề tài:/...../..... Ngày nộp báo cáo:/...../.....

Họ tên GV hướng dẫn: Nguyễn Ngọc Như Hằng Chữ ký:

Ngày ... tháng ... năm ...

TRÍCH YẾU

Được sự hướng dẫn nhiệt tình của cô Nguyễn Ngọc Như Hằng, nhóm chúng tôi gồm hai sinh viên là Quan Thi Trọng và Đặng Ngọc Thành đã tìm hiểu và nghiên cứu đồ án tốt nghiệp: “Xây dựng hệ thống IDS – Snort trên hệ điều hành Linux”. Chúng tôi đã thảo luận và tự phân công công việc cho nhau trong thời gian mười bốn tuần. Đồ án của chúng tôi tập trung nghiên cứu vào các nhiệm vụ chính sau:

- Cấu trúc hoạt động của một hệ thống phát hiện xâm nhập IDS. Phân loại, so sánh giữa các dạng (HIDS và NIDS), sau đó dựa vào các yêu cầu và mục đích triển khai để xác định hướng phát triển của hệ thống.
- Tìm hiểu về các thành phần và các chế độ hoạt động của một hệ thống IDS – Snort.
- Nguyên cứu về các option trong mỗi module của hệ thống Snort.
- Nguyên cứu về các option trong rules Snort.
- Tiến hành mô phỏng hệ thống multi sensor thực hiện việc giám sát cảnh báo bằng Snort và chức năng ngăn chặn của Snort_Inline (với sự hỗ trợ của phần mềm dynagen để thiết kế mirrorport trên switch ảo), tự thiết kế một số rule để kiểm tra việc thực hiện khả năng cảnh báo, ngăn chặn tấn công cũng như mức độ ổn định của hệ thống. Cài đặt chức năng hỗ trợ gửi mail bằng Swatch và kiểm tra việc gửi mail cảnh báo tới người quản trị.
- Thiết kế một đoạn mã script để tối ưu hóa việc hoạt động của hệ thống IPS

Tp.HCM, tháng 7 năm 2011

Nhóm sinh viên

QUÁ TRÌNH LÀM VIỆC

Nhóm chúng tôi đã tự phân chia nhiệm vụ cho nhau, có lịch làm việc cho từng giai đoạn:

Giai đoạn 1:

Tìm tài liệu về hệ thống Snort và cách cài đặt

Làm quen với Ubuntu Server.

Đọc dịch tài liệu.

Giai đoạn 2:

Nghiên cứu về các tiến trình xử lý hoạt động, các rule của hệ thống snort.

Xây dựng hệ thống Snort cơ bản.

Viết một số rules kiểm tra hoạt động của snort.

Nghiên cứu về multi sensor.

Triển khai xây dựng hệ thống cảnh báo qua mail.

Nghiên cứu một đoạn mã hỗ trợ cho hệ thống IPS.

Giai đoạn 3:

Thực hiện mô phỏng và quay video.

Hoàn tất báo cáo.

MỤC LỤC

TRÍCH YẾU	II
QUÁ TRÌNH LÀM VIỆC	III
MỤC LỤC.....	IV
NHẬP ĐỀ	IX
1. GIỚI THIỆU	1
1.1. Khái quát về tình hình Internet	1
1.2. Các kiểu tấn công	2
1.2.1. Kiểu tấn công thăm dò.....	2
1.2.2. Kiểu tấn công truy cập	2
1.2.3. Kiểu tấn công từ chối dịch vụ (DoS).....	2
1.2.4. Các mối đe doạ về bảo mật.....	3
2. TỔNG QUAN VỀ IDS.....	5
2.1. Khái niệm về hệ thống phát hiện xâm nhập	5
2.1.1. Phát hiện xâm nhập là gì?.....	5
2.2. Cấu trúc của hệ thống IDS	6
2.3. Phân loại IDS	7
2.3.1. Hệ thống phát hiện xâm nhập Host-Based (HIDS).....	7
2.3.2. Hệ thống phát hiện xâm nhập Network-Based (NIDS)	8
2.3.3. Những vị trí IDS nên được đặt trong Network Topology.....	9
2.4. Giới thiệu về hệ thống Snort.....	9
2.5. Các thành phần của Snort.....	10
2.6. Các chế độ hoạt động của Snort	13
2.6.1. Sniffer Mode	13
2.6.2. Packet Logger Mode	14
2.6.3. Network Intrusion Detection System Mode	15
2.6.4. Inline Mode	17
3. CẤU HÌNH SNORT	20
3.1. Các biến trong Snort.....	20
3.1.1. Biến IP và Lists IP	20
3.1.2. Biến port và List Port	21
3.2. Cấu hình tiền xử lý	22
3.2.1. Frag3	23

3.2.2. Stream5.....	24
3.2.3. sfPortscan	26
3.2.4. rpc_decode	29
3.2.5. Performance Monitor	30
3.2.6. HTTP Inspect	31
3.2.7. FTP/Telnet Preprocessor	33
3.2.8. SSH.....	34
3.2.9. DNS.....	34
3.2.10. ARP Spoof Preprocessor	34
3.3. Cấu hình luật giải mã và tiền xử lý	35
3.4. Cấu hình Module Output.....	36
3.4.1. alert_syslog	36
3.4.2. alert_fast.....	37
3.4.3. alert_full.....	37
3.4.4. alert_unixsock	38
3.4.5. log_tcpdump	38
3.4.6. Database	38
4. RULE SNORT.....	40
4.1. Giới thiệu	40
4.2. Rules Headers	40
4.2.1. Rule Actions	40
4.2.2. Protocol.....	42
4.2.3. IP address.....	42
4.2.4. Port number	43
4.2.5. The Direction Operator.....	43
4.3. Rule Options	44
4.4. General	44
4.4.1. msg	44
4.4.2. reference.....	44
4.4.3. sid	45
4.4.4. rev	45
4.4.5. classtype	45
4.4.6. priority	46
4.5. Payload Detection.....	46

4.5.1. content.....	46
4.5.2. nocase.....	47
4.5.3. rawbytes	47
4.5.4. depth	47
4.5.5. offset.....	48
4.5.6. distance.....	48
4.5.7. within.....	48
4.5.8. urilen.....	48
4.5.9. isdataat	48
4.5.10. pcre.....	48
4.5.11. byte_test.....	49
4.5.12. byte_jump.....	49
4.5.13. ftpbounce.....	49
4.6. Non-Payload Detection Rule Options.....	49
4.6.1. fragoffset	49
4.6.2. ttl	50
4.6.3. tos	50
4.6.4. id.....	50
4.6.5. ipopts	50
4.6.6. fragbits.....	50
4.6.7. dsizes	51
4.6.8. flags	51
4.6.9. flow.....	52
4.6.10. flowbits	52
4.6.11. seq	52
4.6.12. ack	52
4.6.13. window	53
4.6.14. itype	53
4.6.15. icode	53
4.6.16. icmp_id	53
4.6.17. icmp_seq.....	53
4.6.18. rpc.....	54
4.6.19. ip_proto	54
4.6.20. sameip	54

4.6.21. stream_size.....	54
4.7. Post-Detection Rule Options	55
4.7.1. logto.....	55
4.7.2. session	55
4.7.3. resp.....	55
4.7.4. react.....	55
4.7.5. tag.....	56
4.7.6. activates.....	56
4.7.7. activated_by.....	56
4.7.8. count.....	57
4.7.9. replace	57
4.7.10. detection_filter	57
5. MÔ PHỎNG	58
5.1. Cấu hình Snort IDS:	58
5.1.1. Mô hình:.....	58
5.1.2. Các bước cài đặt.....	59
5.2. Cấu hình Snort Inline IPS:	74
5.2.1. Mô hình.....	74
5.2.2. Các bước cài đặt.....	74
5.3. Cấu hình Mail Alert	77
5.3.1. Mô hình.....	77
5.3.2. Các bước cài đặt.....	77
5.4. NGĂN CHẶN CUỘC TẤN CÔNG DỰA TRÊN SỐ LƯỢNG GÓI TIN TỪ MỘT IP	81
6. ĐÁNH GIÁ KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN.....	83
7. KẾT LUẬN	84
8. PHỤ LỤC	85
9. TÀI LIỆU THAM KHẢO.....	89
NHẬN XÉT CỦA GIÁO VIÊN.....	90

LỜI CẢM ƠN

Để có được cuốn báo cáo này, chúng tôi xin gửi lời cảm ơn sâu sắc nhất tới cô Nguyễn Ngọc Như Hằng, người đã tận tình giúp đỡ, hướng dẫn chúng tôi trong suốt quá trình thực hiện đồ án tốt nghiệp.

Xin chân thành cảm ơn các thầy cô giáo trong khoa Khoa Học Công Nghệ trường Đại Học Hoa Sen đã tận tình giảng dạy, truyền đạt cho chúng tôi những kiến thức quý báu trong suốt thời gian qua.

Cuối cùng xin gửi lời cảm ơn tới gia đình, bạn bè và người thân – những người đã giúp đỡ động viên chúng tôi trong quá trình học tập để có thể đạt được kết quả này.

NHẬP ĐỀ

Trong bối cảnh tiến trình hội nhập, vấn đề an ninh mạng và bảo mật dữ liệu đang trở nên rất được quan tâm không chỉ ở Việt Nam mà trên toàn thế giới. Khi cơ sở hạ tầng và các công nghệ mạng đã đáp ứng tốt các yêu cầu về băng thông, chất lượng dịch vụ, đồng thời thực trạng tấn công trên mạng đang ngày một gia tăng thì vấn đề bảo mật càng được chú trọng hơn bao giờ hết. Không chỉ các nhà cung cấp dịch vụ Internet, các cơ quan chính phủ mà các doanh nghiệp, tổ chức cũng có ý thức hơn về an toàn thông tin.

Triển khai một hệ thống thông tin và xây dựng được cơ chế bảo vệ chặt chẽ, an toàn, là góp phần duy trì tính bền vững cho hệ thống thông tin của doanh nghiệp đó.

Chúng ta đều hiểu rằng an toàn thông tin là tài sản vô giá đối với một doanh nghiệp, không chỉ thuần tuý về mặt vật chất mà đây còn là giá trị về mặt chữ tín, nhất là đối với những tổ chức thường xuyên hoạt động trên mảng thương mại điện tử, dịch vụ, ngân hàng.... Chính vì vậy, tất cả những hệ thống này cần được trang bị những công cụ phòng chống đủ mạnh để có thể đối phó khi có những tấn công vào hệ thống mạng. Am hiểu các cách thức tấn công cũng như những phương pháp phòng vệ khi triển khai và xây dựng một hệ thống mạng là điều cần thiết đối với một người quản trị, và đó là lý do để chúng tôi thực hiện đề án “Xây dựng hệ thống IDS – Snort trên hệ điều hành Linux” này.

1. GIỚI THIỆU

1.1. Khái quát về tình hình Internet

Ngày nay, Internet phát triển rất mạnh mẽ và đóng một vai trò quan trọng trong đời sống con người. Mạng Internet mang lại rất nhiều tiện ích hữu dụng cho người sử dụng, phô thông như hệ thống thư điện tử, tán gẫu trực tuyến, công cụ tìm kiếm, các dịch vụ thương mại và các dịch vụ về y tế giáo dục như là chữa bệnh từ xa hoặc tổ chức các lớp học trực tuyến... Chúng cung cấp một khối lượng thông tin và dịch vụ khổng lồ trên Internet. Trong những năm gần đây, sự phát triển của điện toán đám mây, điện toán di động, mạng xã hội,... đã làm cho mạng Internet càng không thể thiếu trong đời sống con người.

Ngoài những lợi ích mà Internet mang lại cho con người thì hiểm họa từ Internet mang đến cũng không ít. Nhiều người đã dựa trên những lỗ hổng bảo mật của Internet để xâm nhập, chiếm dụng thông tin hoặc phá hoại các hệ thống máy tính khác. Những người như vậy thường được gọi với cái tên “hacker”.

Với định nghĩa trước đây, Hacker ám chỉ một người tài giỏi. Người này có khả năng chỉnh sửa phần mềm, phần cứng máy tính bao gồm lập trình, quản trị và bảo mật. Những người được mệnh danh là Hacker là người hiểu rõ hoạt động của hệ thống máy tính, mạng máy tính và dùng kiến thức bản thân để làm thay đổi, chỉnh sửa nó. Nhưng dần dần, khi mọi người nghe tới Hacker thì thường liên tưởng ngay tới một kẻ có mục đích phá hoại và tấn công các hệ thống mạng để ăn cắp thông tin.

Symantec nhận định: “Trước đây, những kẻ tấn công thường phải tự tạo dựng công cụ từ đầu. Quy trình phức tạp này khiến cho các cuộc tấn công chỉ bó hẹp trong phạm vi những kẻ tội phạm mạng có kỹ năng cao. Tuy nhiên, các công cụ tấn công ngày nay lại rất dễ sử dụng, và thậm chí chúng còn giúp những kẻ mới tập tành vào nghề cũng tự mình tấn công được mục tiêu. Do vậy, chúng tôi cho rằng sẽ có nhiều hoạt động tội phạm trong lĩnh vực này, và nhiều khả năng những người dùng trung bình cũng sẽ trở thành nạn nhân”^[1]. Theo thống kê: “Các doanh nghiệp Mỹ mỗi năm thiệt hại hàng tỷ đô-la vì tội phạm mạng.^[2]”, “bộ phận quản trị hệ thống của ngân hàng VietinBank cho biết mỗi ngày có 13.300 virus, gần 40 spyware/grayware và khoảng 67.000 thư rác được phát hiện trên toàn hệ thống nhà băng này.^[3]”, “Facebook và Twitter đồng loạt bị tấn công^[4] bằng DDoS”, “Hàng trăm nghìn trang web bị tấn công^[5]”...

1.2. Các kiểu tấn công

1.2.1. Kiểu tấn công thăm dò

Thăm dò là việc thu thập dữ liệu trái phép về tài nguyên, các lỗ hổng hoặc dịch vụ của hệ thống. Các cách tấn công truy cập hay DoS thường được tiến hành bởi kiểu tấn công thăm dò để tìm hiểu sơ lược về những thông tin bảo mật của một tổ chức, doanh nghiệp hay công ty nào đó. Kẻ tấn công sử dụng kỹ thuật này để khám phá hệ thống mục tiêu đang chạy trên hệ điều hành nào, các dịch vụ và các cổng nào đang được mở, địa chỉ IP, kiến trúc hệ thống mạng... nhằm đưa ra những hình thức thâm nhập hợp lý. Thăm dò và thu thập thông tin còn là cách duy nhất để biết được các kiểu kết nối, như Internet, Intranet, Wireless... và các cấu trúc hệ thống đang được mục tiêu sử dụng.

1.2.2. Kiểu tấn công truy cập

Tấn công truy cập là kiểu tấn công mà các hacker lợi dụng các lỗ hổng bảo mật và các lỗi cấu hình hệ thống để lấy quyền xâm nhập trái phép vào hệ thống và thay đổi cấu trúc thông tin của mục tiêu.

Kẻ tấn công thường tìm kiếm quyền truy cập bằng cách chạy một đoạn mã, các công cụ hack hay khai thác một số điểm yếu của ứng dụng hoặc một dịch vụ đang chạy trên máy chủ. Sau khi có quyền truy cập, kẻ tấn công sẽ tìm cách nâng cao đặc quyền của mình, cài đặt các phần mềm backdoor, trojan để chiếm quyền truy cập ở mức độ quản trị (superuser, admin, root). Khi đã nắm toàn quyền, kẻ tấn công có thể điều khiển hệ thống mạng mục tiêu để thực hiện các mục đích của mình, như một bước đệm để tấn công lên các hệ thống máy chủ mẹ, hay sử dụng hệ thống mục tiêu như một agent để tấn công DoS vào các hệ thống khác.

1.2.3. Kiểu tấn công từ chối dịch vụ (DoS)

Tấn công từ chối dịch vụ chỉ là tên gọi chung của cách tấn công làm một hệ thống bị quá tải không thể cung cấp dịch vụ, hoặc phải ngưng hoạt động.

Tấn công DoS nói chung không nguy hiểm như các kiểu tấn công khác ở chỗ nó không cho phép kẻ tấn công chiếm quyền truy cập hệ thống hay có quyền thay đổi hệ thống. Tuy nhiên, nếu một máy chủ tồn tại mà không thể cung cấp thông tin, dịch vụ cho người sử dụng, sự tồn tại là không có ý nghĩa nên thiệt hại do các cuộc tấn công DoS do máy chủ bị đình trệ hoạt động là vô cùng lớn, đặc biệt là các hệ thống phục vụ các giao dịch điện tử.

a) Mục đích của tấn công DoS

- Chiếm băng thông mạng và làm hệ thống mạng bị ngập (flood), khi đó hệ thống mạng sẽ không có khả năng đáp ứng những dịch vụ khác cho người dùng bình thường;
- Làm ngắt kết nối giữa hai máy, và ngăn chặn quá trình truy cập vào dịch vụ.

b) Mục tiêu của tấn công DoS

- Tạo ra sự khan hiếm, những giới hạn và không đổi mới tài nguyên
- Băng thông của hệ thống mạng (Network Bandwidth), bộ nhớ, ổ đĩa, và CPU Time hay cấu trúc dữ liệu đều là mục tiêu của tấn công DoS.
- Phá hoại hoặc thay đổi các thông tin cấu hình.

1.2.4. Các mối đe dọa về bảo mật

Chính vì một hệ thống thông tin luôn bị đe dọa tấn công bởi các hacker nên việc xây dựng một hệ thống bảo vệ xâm nhập là rất cần thiết đối với mỗi một tổ chức. Các hình thức tấn công của hacker ngày càng tinh vi, chau chuốt hơn, cũng như mức độ tấn công ngày càng khủng khiếp hơn, nên không một hệ thống nào có thể đảm bảo hoàn toàn không bị xâm nhập. Nếu các tổ chức antivirut đang cố gắng cập nhập, sửa đổi để cung cấp cho người dùng những phương pháp phòng chống hiệu quả thì bên cạnh đó, những kẻ tấn công cũng ngày đêm nghiên cứu tung ra các hình thức xâm nhập, phá hoại khác.

Để bảo vệ tốt được một hệ thống, đầu tiên bạn phải có cái nhìn tổng quát về các nguy cơ tấn công, nghĩa là đầu tiên bạn phải nhận định được bạn cần bảo vệ cái gì, và bảo vệ khỏi ai, cũng như phải hiểu các kiểu đe dọa đến sự bảo mật mạng của bạn. Thông thường sẽ có 4 mối đe dọa bảo mật sau:

- Mối đe dọa ở bên trong
- Mối đe dọa ở bên ngoài
- Mối đe dọa không có cấu trúc và có cấu trúc

a) Mối đe dọa ở bên trong

Mối đe dọa bên trong là kiểu tấn công được thực hiện từ một cá nhân hoặc một tổ chức được tin cậy trong mạng và có một vài quyền hạn để truy cập vào hệ thống. Hầu hết chúng ta chỉ quan tâm xây dựng một thông firewall và giám sát dữ liệu truy cập ở các đường biên mạng mà ít để ý đến các truy cập trong mạng nội bộ do sự tin tưởng vào các chính sách và ACL được người quản trị quy định trong hệ thống. Do sự bảo mật trong một mạng local thường rất lỏng lẻo nên đây là môi trường thường được các hacker sử dụng để tấn công hệ thống.

Mỗi đe doạ bên trong thường được thực hiện bởi các nhân viên do có bất đồng với công ty, các gián điệp kinh tế hay do một vào máy client đã bị hacker chiếm quyền truy cập. Mỗi đe doạ này thường ít được để ý và phòng chống vì các nhân viên có thể truy cập vào mạng và dữ liệu quan trọng của server.

b) Mối đe doạ ở bên ngoài

Mối đe doạ bên ngoài là việc các hacker cố gắng xâm nhập vào một hệ thống mạng nào đó bằng một vài kỹ thuật (thăm dò, truy cập...) hay việc phá hoại truy cập hệ thống (DoS, DDoS...). Xây dựng hệ thống firewall và cảnh báo để ngăn ngừa các mối đe doạ từ bên ngoài là việc mà các công ty và tổ chức thường phải bỏ nhiều thời gian và tiền bạc để đầu tư phát triển

c) Mối đe doạ không có cấu trúc và có cấu trúc

Mối đe doạ tấn công vào một hệ thống có thể đến từ rất nhiều loại. Phổ biến nhất là các hacker mới vào nghề, còn ít kiến thức và không có kinh nghiệm, thực hiện việc tấn công bằng cách sử dụng các công cụ hoặc thực hiện tấn công DoS (mối đe doạ không có cấu trúc). Hoặc việc tấn công được thực hiện bởi các hacker thực thụ hoặc cả một tổ chức (mối đe doạ có cấu trúc), họ là những người có kiến thức và kinh nghiệm cao, nắm rõ việc hoạt động của các hệ thống, giao thức mạng cũng như các phương pháp thường được sử dụng để ngăn chặn trong các firewall. Đây là mối đe doạ khó ngăn ngừa và phòng chống nhất đối với các hệ thống mạng.

2. TỔNG QUAN VỀ IDS

Hệ thống phát hiện xâm nhập – IDS (Intrusion Detection System) là một hệ thống có nhiệm vụ giám sát các luồng dữ liệu traffic đang lưu thông trên mạng, có khả năng phát hiện những hành động khả nghi, những xâm nhập trái phép cũng như khai thác bất hợp pháp nguồn tài nguyên của hệ thống mà từ đó có thể dẫn đến xâm hại tính toàn ổn định, toàn vẹn và sẵn sàng của hệ thống.

IDS có thể phân biệt được những cuộc tấn công xuất phát từ bên ngoài hay từ chính bên trong hệ thống bằng cách dựa vào một database dấu hiệu đặc biệt về những cuộc tấn công (smurf attack, buffer overflow, packet sniffers....). Khi một hệ thống IDS có khả năng ngăn chặn các cuộc tấn công thì nó được gọi là hệ thống ngăn chặn xâm nhập – IPS (Intrusion Prevention System).

Có rất nhiều công cụ IDS, trong đó Snort được sử dụng rất nhiều vì khả năng tương thích có thể hỗ trợ cài đặt trên cả hai môi trường Window và Linux. Khi Snort phát hiện những dấu hiệu của một cuộc tấn công, tùy thuộc vào cấu hình và những qui tắc do người quản trị qui định (Snort Rule) mà Snort có thể đưa ra những hành động khác nhau, như gửi cảnh báo đến người quản trị hay ghi log file, loại bỏ các gói tin xâm nhập hệ thống....

2.1. Khái niệm về hệ thống phát hiện xâm nhập

2.1.1. Phát hiện xâm nhập là gì?

Phát hiện xâm nhập là một tập hợp các kỹ thuật và phương pháp được sử dụng để phát hiện những hành vi đáng ngờ ở cấp độ mạng và máy chủ. Hệ thống phát hiện xâm nhập có hai loại cơ bản: phát hiện xâm nhập dựa trên dấu hiệu signature và phát hiện sự bất thường.

a) Phát hiện dựa trên dấu hiệu (signature)

Phương pháp này nhận dạng cuộc tấn công bằng cách so sánh dấu hiệu nhận được với một tập hợp các dấu hiệu đã biết trước được xác định là sự tấn công. Phương pháp này có hiệu quả với những dấu hiệu đã biết trước, như virus máy tính, có thể được phát hiện bằng cách sử dụng phần mềm để tìm các gói dữ liệu có liên quan đến sự xâm nhập trong các giao thức Internet. Dựa trên một tập hợp các dấu hiệu và các quy tắc, hệ thống phát hiện xâm nhập có thể tìm thấy và ghi log lại các hoạt động đáng ngờ và tạo ra các cảnh báo. Tuy nhiên phương pháp này hầu như không có tác

dụng với những cuộc tấn công mới, quy mô phức tạp, sử dụng các kỹ thuật lẩn tránh (evasion technique)... do chưa có được thông tin về cuộc tấn công.

b) Phát hiện sự bất thường

Phương pháp này thiết lập và ghi nhận trạng thái hoạt động ổn định của hệ thống, sau đó so sánh với trạng thái đang hoạt động hiện hành để kiểm tra sự chênh lệch. Khi nhận ra sự khác biệt lớn trong hệ thống thì có khả năng đã xảy ra một cuộc tấn công, Ví dụ như sự tăng đột biến các traffic truy cập vào một website.... Phát hiện xâm nhập dựa trên sự bất thường thường phụ thuộc vào các gói tin hiện diện trong phần tiêu đề giao thức. Trong một số trường hợp các phương pháp này cho kết quả tốt hơn so với IDS dựa trên signature. Thông thường một hệ thống phát hiện xâm nhập thu thập dữ liệu từ mạng và áp dụng luật của nó với dữ liệu để phát hiện bất thường trong đó. Snort là một IDS chủ yếu dựa trên các luật lệ, và những plug-in hiện nay để phát hiện bất thường trong tiêu đề giao thức.

Quá trình phát hiện có thể được mô tả bởi 3 yếu tố cơ bản nền tảng sau:

- Thu thập thông tin (information source): Kiểm tra các gói tin trên mạng.
- Sự phân tích (Analysis): Phân tích các gói tin đã thu thập để nhận biết hành động nào là tấn công.
- Cảnh báo (response): hành động cảnh báo cho sự tấn công được phân tích ở trên.

Snort sử dụng các quy tắc được lưu trữ trong các tập tin văn bản có thể sửa đổi. Nội quy được nhóm lại trong các chuyên mục và được lưu trữ trong các tập tin riêng biệt. Những tập tin này sau đó được tập hợp trong một tập tin cấu hình chính gọi là snort.conf. Snort đọc những quy định này trong thời gian khởi động và xây dựng cấu trúc dữ liệu nội bộ hoặc dây chuyền để áp dụng những quy tắc này capture dữ liệu. Tìm và xử lý dấu hiệu theo các luật là một việc khó khăn vì việc xử lý yêu cầu phải capture và phân tích dữ liệu trong một thời gian. Snort đi kèm với một tập hợp phong phú của các tiền quy tắc xác định để phát hiện hoạt động xâm nhập, bạn cũng có thể tự thêm hoặc loại bỏ các quy tắc tùy thuộc vào mục đích cảnh báo của hệ thống.

2.2. Cấu trúc của hệ thống IDS

Các thành phần cơ bản

- *Sensor/Agent*: Giám sát và phân tích các hoạt động. “Sensor” thường được dùng cho dạng Network-base IDS/IPS trong khi “Agent” thường được dùng cho dạng Host-base IDS/IPS. Sensor/Agent là các bộ cảm biến được đặt

trong hệ thống nhằm phát hiện những xâm nhập hoặc các dấu hiệu bất thường trên toàn mạng.

- *Management Server*: Là thiết bị trung tâm dùng thu nhận các thông tin từ Sensor/Agent và quản lý chúng, management server thường là các máy trạm trọng một hệ thống. Một số Management Server có thể thực hiện việc phân tích các thông tin sự và nhận dạng được các sự kiện này trong khi các Sensor/Agent đơn lẻ không thể nhận diện.
- *Database server*: Dùng lưu trữ các thông tin từ Sensor/Agent hay Management Server
- *Console*: Là chương trình cung cấp giao diện 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.

2.3. Phân loại IDS

Các hệ thống IDS được chia thành 2 loại sau:

- *Host-based IDS (HIDS)*: Sử dụng dữ liệu kiểm tra trên một hoặc vài máy 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 cá bộ cảm biến và một vài máy trạm để phát hiện xâm nhập.

2.3.1. Hệ thống phát hiện xâm nhập Host-Based (HIDS)

Host-base IDS (HIDS) kiểm tra sự xâm nhập bằng cách quan sát và phân tích các thông tin ở mức độ host hay hệ điều hành trên những giao diện của hệ thống, như những cuộc gọi (system call), bản ghi (audit log), hay những thông điệp lỗi (error message)... Một hệ thống phát hiện xâm nhập host-base có thể kiểm tra các file hệ thống và những file log ứng dụng để phát hiện dấu hiệu hoạt động của kẻ xâm nhập nhằm bảo vệ những tài nguyên đặc biệt của hệ thống bao gồm những tập tin mà chỉ có thể tồn tại trên mỗi host.

Nhiệm vụ của HIDS là đưa ra phản ứng, nghĩa là nó sẽ gửi các thông báo đến người quản trị khi phát hiện những sự kiện xảy ra trong thời gian thực. Khác với NIDS hoạt động cùng với các bộ cảm biến sensor có nhiệm vụ giám sát và ngăn chặn

các cuộc tấn công trên một network segment hay trên toàn hệ thống mạng, HIDS thường được cài đặt và giám sát các hoạt động trên mỗi máy tính độc lập nên nó có thể xác định xem một cuộc tấn công có thành công hay không dựa vào những ảnh hưởng trên hệ thống. HIDS thường được đặt trên các host xung yếu của hệ thống, và các server trong vùng DMZ – thường là mục tiêu bị các hacker tấn công đầu tiên. Nhiệm vụ chính của HIDS là giám sát các thay đổi trên hệ thống, bao gồm (không phải tất cả):

- Các tiến trình.
- Các entry của Registry.
- Mức độ sử dụng CPU.
- Kiểm tra tính toàn vẹn và truy cập trên hệ thống file.
- Một vài thông số khác. Các thông số này khi vượt qua một ngưỡng định trước hoặc những thay đổi khả nghi trên hệ thống file sẽ gây ra báo động.

HIDS có một vai trò quan trọng trong hệ thống bởi vì không phải tất cả các cuộc tấn công đều được thực hiện qua mạng. Ví dụ như bằng cách giành quyền truy cập ở mức vật lý (physical access) vào một hệ thống máy tính, kẻ xâm nhập có thể tấn công một hệ thống hay dữ liệu mà không cần phải tạo ra bất cứ lưu lượng mạng (network traffic) nào cả, do đó đối với các hệ thống sử dụng NIDS sẽ không thể phát hiện ra các tấn công này. Một ưu điểm khác của HIDS là nó có thể ngăn chặn các kiểu tấn công dùng sự phân mảnh hoặc TTL, vì một host phải nhận và tái hợp các phân mảnh khi xử lý lưu lượng nên IDS dựa trên host có thể giám sát chuyện này.

2.3.2. Hệ thống phát hiện xâm nhập Network-Based (NIDS)

Network-based IDS (NIDS) kiểm tra sự xâm nhập bằng cách sử dụng các bộ dò tìm và các bộ cảm biến (sensor) cài đặt trên toàn mạng để giám sát hoạt động của hệ thống. Những bộ cảm biến thu nhận và phân tích lưu lượng cũng như kiểm tra các header của tất cả các gói tin trong thời gian thực, sau đó so sánh các kết quả nhận được với một database các mô tả sơ lược được định nghĩa hay là những dấu hiệu để nhận định có xảy ra một cuộc tấn công hay không. Khi ghi nhận được một sự kiện bất thường, bộ cảm biến sẽ gửi tín hiệu cảnh báo đến trạm quản trị và thực hiện vài hành động dựa vào các rule đã được cấu hình trước.

NIDS thường được đặt ở những vị trí trọng yếu như những network interface kết nối hệ thống giữa mạng bên trong và mạng bên ngoài để giám sát toàn bộ lưu lượng vào ra, do đó nó có thể giám sát toàn bộ các traffic lưu thông trên mạng. Việc giám sát dựa vào các bộ cảm biến sensor và các trạm trên hệ thống không cần phải nạp

các phần mềm và quản lý trên mỗi máy trong mạng. NIDS phát hiện các tấn công ngay khi xảy ra, vì thế việc cảnh báo và đối phó có thể được thực hiện một cách nhanh chóng. Tuy nhiên NIDS sẽ gặp khó khăn trong việc xử lý toàn bộ các gói tin trên một mạng có mật độ traffic cao, dẫn đến việc bỏ sót một số gói tin có thể là nguyên nhân gây nên các lỗ hổng cho các cuộc tấn công vào hệ thống.

HIDS và NIDS có những ưu và khuyết điểm riêng trong việc giám sát và đưa ra các cảnh báo, tùy vào từng mô hình mà mỗi người quản trị sẽ lựa chọn cho mình những chính sách xây dựng các IDS phù hợp cho từng hệ thống mạng. Trong thực tế, NIDS thường được sử dụng tại biên mạng nhằm phát hiện các dấu hiệu tấn công và hạn chế các tấn công này ở mức network. Đối với những máy chủ hoặc máy client quan trọng, việc bổ sung HIDS cho các máy này là cần thiết để tăng cường khả năng bảo mật khi kết hợp với các hệ NIDS trong cùng hệ thống.

2.3.3. Những vị trí IDS nên được đặt trong Network Topology

Tùy thuộc vào cấu trúc liên kết mạng của bạn, bạn có thể muốn đặt hệ thống phát hiện xâm nhập tại một hoặc nhiều địa điểm. Nó cũng phụ thuộc vào loại hoạt động xâm nhập bạn muốn phát hiện: bên trong, bên ngoài hoặc cả hai. Ví dụ, nếu bạn chỉ muốn phát hiện hoạt động xâm nhập bên ngoài, và bạn chỉ có một router kết nối với Internet, nơi tốt nhất cho một hệ thống phát hiện xâm nhập có thể được là bên trong các bộ định tuyến hay tường lửa. Nếu bạn có nhiều đường vào Internet, bạn có thể muốn đặt IDS ở mỗi điểm ra vào. Tuy nhiên nếu bạn muốn phát hiện một cách tốt nhất các mối đe dọa trong một mạng nội bộ, bạn có thể muốn đặt một IDS trong mỗi phân đoạn mạng.

Trong nhiều trường hợp bạn không cần phải có các hoạt động phát hiện xâm nhập trong tất cả các phân đoạn mạng và bạn có thể muốn hạn chế nó chỉ đến các khu vực mạng nhạy cảm. Lưu ý rằng việc triển khai hệ thống phát hiện xâm nhập nhiều hơn có nghĩa là làm việc nhiều hơn và chi phí bảo trì hơn. Quyết định của bạn thực sự phụ thuộc vào chính sách bảo mật của bạn, trong đó xác định những gì bạn thực sự muốn bảo vệ từ tin tặc.

2.4. Giới thiệu về hệ thống Snort

Snort được phát triển năm 1998 bởi Sourcefire và CTO Martin Roesch, là 1 phần mềm miễn phí mã nguồn mở có khả năng phát hiện và phòng chống xâm nhập trái phép vào hệ thống mạng có khả năng phân tích thời gian thực lưu lượng mạng, và ghi log gói tin trên nền mạng IP. Ban đầu được gọi công nghệ phát hiện và phòng chống xâm nhập hạng nhẹ, Snort đã dần phát triển và trở thành tiêu chuẩn trong việc

phát hiện và phòng chống xâm nhập. Với hơn 3,7 triệu lượt tải về và hơn 250 ngàn người dùng đăng ký, Snort trở thành công nghệ phát hiện và phòng chống xâm nhập được sử dụng rộng rãi nhất hiện nay.

Snort thực hiện việc tìm kiếm và phân tích nội dung các giao thức của các traffic lưu thông trên mạng, từ đó có thể phát hiện ra các kiểu thăm dò và tấn công như buffer overflow, stealth ports scanning.... Các thông tin thu thập sẽ được ghi log lại và cảnh báo đến console của người quản trị trong thời gian thực.

Snort có thể thực hiện phân tích giao thức và tìm kiếm nội dung, từ đó có thể phát hiện rất nhiều kiểu thăm dò và tấn công như buffer-overflow, stealth ports scanning,..v.v. Để có thể làm được điều này, Snort dùng 1 loại ngôn ngữ mô tả các quy tắc giao thông mạng mà nó sẽ thu thập hoặc bỏ qua, cũng như sử dụng cơ chế phát hiện xâm nhập theo kiến trúc modular plug-ins. Nó cũng có khả năng cảnh báo tức thời, kết hợp với các cơ chế cảnh báo syslog, tập tin người dùng chỉ định, Unix socket hoặc Winpopup message.

Snort có thể sử dụng với 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.

2.5. Các thành phần của Snort

Snort bao gồm nhiều thành phần. Mỗi phần có một chức năng riêng biệt nhưng làm việc cùng nhau để góp phần đưa ra các nhận định giúp phát hiện các cuộc tấn công cụ thể và tạo ra output theo một định dạng cần thiết từ hệ thống. Một IDS dựa trên Snort bao gồm các thành phần chính sau đây:

- Module giải mã gói tin (Packet Decoder)

- Module tiền xử lý (Preprocessors)
- Module phát hiện (Detection Engine)
- Module log và cảnh báo (Logging and Alerting System)
- Module xuất thông tin (Output Module)

a) Module giải mã

Module giải mã lấy các gói dữ liệu từ các giao diện mạng khác nhau và chuẩn bị cho việc các gói tin sẽ được xử lý trước khi được gửi đến module phát hiện. Các giao diện có thể là Ethernet, SLIP, PPP....

b) Module tiền xử lý

Module tiền xử lý là những thành phần được sử dụng với Snort để sắp xếp hoặc sửa đổi các gói dữ liệu trước khi module phát hiện kiểm tra xem gói tin đã được xử dụng bởi kẻ xâm nhập hay không. Module tiền xử lý là phần rất quan trọng trong IDS để chuẩn bị các gói dữ liệu cho việc phân tích dựa vào các luật trong module phát hiện. Hacker sử dụng các kỹ thuật khác nhau để đánh lừa IDS bằng nhiều cách. Ví dụ, bạn có thể đã tạo ra một quy tắc để tìm một chuỗi ký tự "scripts/iisadmin" trong gói HTTP. Nếu bạn đang kết hợp chuỗi này chính xác, bạn có thể dễ dàng bị lừa bởi một hacker khi đã làm thay đổi chút ít các chuỗi này.

```
"scripts/.iisadmin"  
"scripts/examples/..iisadmin"  
"scripts\iisadmin"  
"scripts/.\\iisadmin"
```

Module tiền xử lý còn được dùng để chống phân mảnh gói tin. Khi một khối dữ liệu lớn được chuyển tới một host, gói tin thường bị phân mảnh. Ví dụ, chiều dài tối đa mặc định của bất kỳ gói dữ liệu trên một mạng Ethernet thường là 1500 byte. Giá trị này được qui định bởi giá trị MTU (Maximum Transfer Unit) cho mỗi giao diện mạng. Điều này có nghĩa rằng nếu bạn gửi dữ liệu hơn 1500 byte, nó sẽ được chia thành nhiều gói dữ liệu sao cho mỗi mảnh gói tin nhỏ hơn hoặc bằng 1500 byte. Các hệ thống tiếp nhận có khả năng ghép các đơn vị này để hình thành các gói dữ liệu gốc. Trên IDS, trước khi bạn có thể áp dụng bất kỳ quy tắc nào hoặc cố gắng tìm một chuỗi ký tự, bạn cần phải tập hợp lại các gói tin. Ví dụ, một nửa số ký tự có thể có mặt trong một gói và một nửa khác trong gói khác. Để phát hiện các ký tự đúng, bạn phải kết hợp tất cả các gói tin lại. Lợi dụng yếu tố phân mảnh của các gói tin, hacker có thể để đánh bại các hệ thống phát hiện xâm nhập.

Do đó module tiền xử lý được sử dụng để bảo vệ chống lại các cuộc tấn công. Module tiền xử lý trong Snort có thể chống phân mảnh, giải mã HTTP, URI.... Các chức năng này là một phần rất quan trọng của hệ thống phát hiện xâm nhập.

c) *Module phát hiện*

Đây là phần quan trọng nhất của Snort. Nó chịu trách nhiệm phát hiện các dấu hiệu xâm nhập vào hệ thống. 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. Nếu một gói tin phù hợp với bất kì luật nào, một hành động tương ứng sẽ được thực hiện như ghi log, tạo ra các cảnh báo, đưa ra các thông tin... hoặc gói tin sẽ bị hủy.

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 tồn 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

Khi xây dựng một hệ thống phát hiện xâm nhập, bạn phải lưu ý tất cả các yếu tố này để hệ thống có thể hoạt động hiệu quả. Lưu ý là module phát hiện có thể phân tích các gói tin và áp dụng các luật đối với các phần khác nhau của một gói tin. Đó là:

- IP header của gói tin
- Header của lớp vận chuyển, bao gồm TCP, UDP... Nó cũng có thể làm việc trên header ICMP
- Header của lớp ứng dụng như header của DNS, FTP, SNMP hay SMTP...
- Tải trọng của gói: Điều này có nghĩa bạn có thể tạo ra các luật sử dụng module phát hiện để tìm kiếm một chuỗi nằm trong dữ liệu của một gói tin.

d) *Module ghi và cảnh báo*

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 log file và cảnh báo.

e) *Module Output*

Module đầu ra có thể hoạt động theo nhiều cách phụ thuộc vào việc muốn lưu các kết quả được tạo ra bằng module ghi và cảnh báo như thế nào. Tùy theo việc cấu hình hệ thống mà nó có thể thực hiện các công việc như là:

- Ghi log file vào thư mục mặc định /var/log/snort/alerts hoặc một nơi khác.
- Gửi các cảnh báo SNMP.
- Gửi thông tin tới syslog.
- Đăng nhập vào một cơ sở dữ liệu như MySQL hoặc Oracle.
- Tạo ra các output XML.
- Cấu hình lại Router, firewall.
- Gửi các thông điệp Server Message Block (SMB) tới các máy tính Windows.

Module	Mô tả
Giải mã	Chuẩn bị các gói dữ liệu cho việc xử lý.
Tiền xử lý	Bình thường hóa header các giao thức, phát hiện dấu hiệu bất thường, chống phân mảnh và khôi phục dữ liệu gốc, sắp xếp lại các trường TCP.
Phát hiện	Áp dụng bộ luật cho các gói tin.
Log và cảnh báo	Tạo cảnh báo và các thông tin log.
Output	Xử lý cảnh báo, log và đưa ra kết quả output cuối cùng.

Bảng 1 Bảng tóm tắt các module của Snort

2.6. Các chế độ hoạt động của Snort

2.6.1. 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. 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 > ttooip.log
```

Ngoài ra, do một số thiết bị chuyển mạch có thể chia ra hay gộp chung các khóa trong một dòng lệnh, ta cũng có thể nhập ra như:

```
./snort -d -v -e
```

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 – Một vài khóa trong mode sniffer của Snort

2.6.2. 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
```

Tất nhiên, điều này giả định bạn có một thư mục với tên log trong thư mục hiện hành. Thư mục mặc định trong Snort là /var/log/snort. Khi Snort chạy trong chế độ này, nó thu thập tất cả các gói nó thấy và đặt nó vào một thư mục hệ thống phân cấp dựa trên địa chỉ IP của một trong những máy chủ trong các gói dữ liệu. Nói cách khác, một thư mục mới được tạo ra cho mỗi địa chỉ được capture và dữ liệu liên quan đến địa chỉ này được lưu trong thư mục đó. Snort lưu các gói tin thành các file ASCII, với tên file được tạo ra từ tên giao thức và số của cổng kết nối.

Nếu bạn chỉ định một chuyển đổi đồng bằng -l, bạn có thể nhận thấy rằng Snort đổi khi sử dụng địa chỉ của các máy tính từ xa như các thư mục trong đó nó đặt gói và đổi khi nó sử dụng địa chỉ host nội bộ. Để thiết lập Snort cho các mạng gia đình, bạn cần chỉ định cho Snort biết mạng cần đi đến là các mạng gia đình:

```
./snort -dev -l file./log -h 192.168.1.0/24
```

Quy luật này cho Snort biết bạn muốn in ra các liên kết dữ liệu và các TCP/IP header cũng như dữ liệu ứng dụng vào thư mục./log, và bạn muốn log các gói liên quan đến mạng lớp C 192.168.1.0. Tất cả các gói tin đến sẽ được ghi vào thư mục con của thư mục log, với các tên thư mục được dựa trên địa chỉ của các máy chủ từ xa.

2.6.3. Network Intrusion Detection System Mode

Kích hoạt hệ thống phát hiện xâm nhập mạng (NIDS) để bạn không phải ghi lại tất cả gói đơn lẻ được gửi xuống bằng lệnh:

```
./snort -d -h 192.168.1.0/24 -l file./log -c snort.conf
```

snort.conf là tên của tập tin nơi mà bạn qui định các luật của snort, tập tin này nằm trong /etc. 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 bạn 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 plain ASCII vào đĩa lưu trữ bằng cách sử dụng một cấu trúc thư mục phân cấp (giống như chế độ packet logger).

a) 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.

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 3 – Các tùy chọn của Snort ở chế độ NIDS

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**.

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. Ví dụ, sử dụng dòng lệnh sau đây để log các file theo chế độ mặc định (giải mã ASCII) và gửi thông báo đến syslog:

```
./snort -c snort.conf -l./log -h 192.168.1.0/24 -s
```

b) Understanding Standard Alert Output

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

[**] [116:56:1] (snort_decoder): T/TCP phát hiện [**]

Thông số đầu tiên (116) là Generator ID, 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). Đôi với một danh sách các SID tiền xử lý, xin vui lòng xem etc/gen-msg.map. 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 một T/TCP sự kiện.

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 quy tắc, nên tăng con số này để phân biệt các tùy chọn rev.

2.6.4. 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 quy 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 kỳ thông tin gì như hành động LOG.

a) Snort Inline Rule Application Order

Trong các phiên bản gốc, trình tự ưu tiên của các hành động trong Snort là:

activation → dynamic → drop → sdrop → reject → alert → pass → log

Trong inline-mode, trình tự ưu tiên này được thay đổi như sau:

activation → dynamic → pass → drop → sdrop → reject → alert → log

Điều này sẽ đảm bảo rằng một hành động drop sẽ được ưu tiên hơn một cảnh báo alert hoặc hành động log.

b) Cài đặt Snort Inline

Để cài đặt Snort Inline, sử dụng lệnh sau đây:

```
# ./configure --enable-inline
# make
# make install
```

c) *Chạy Snort Inline*

Trước tiên, bạn cần đảm bảo rằng các module ip queue đã được tải. Sau đó, bạn cần gửi lưu lượng truy cập để Snort Inline sử dụng các hàng đợi đối tượng. Ví dụ:

```
# iptables -A OUTPUT -p tcp --dport 80 -j QUEUE
```

Gửi tất cả lưu lượng TCP ra khỏi firewall đến cổng 80 với hàng đợi đối tượng. Đây là những gì sẽ gửi gói tin từ hạt nhân kernel tới không gian người dùng (Snort Inline). Cuối cùng, để bắt đầu Snort Inline:

```
# snort_inline -QDc./etc/drop.conf -l /var/log/snort
```

Bạn có thể sử dụng tùy chọn dòng lệnh sau đây:

- -Q lấy các gói tin từ iptables.
- -D Chạy Snort Inline ở chế độ nền. Các mã số tiến trình (process ID) được lưu ở /var/run/snort.pid
- -c Đọc các tập tin cấu hình.
- -l Ghi log vào thư mục.

Tốt nhất Snort Inline nên được chạy bằng cách sử dụng cấu hình drop.rules riêng của mình. Nếu bạn muốn sử dụng Snort để chỉ cảnh báo, một quá trình riêng biệt nên được chạy với bộ quy tắc riêng của mình.

d) *Pcaps*

Thay vì lắng nghe Snort trên giao diện, bạn có thể capture một gói tin để đọc. Snort sẽ đọc và phân tích các gói tin và điều này có thể hữu ích cho việc kiểm tra và phát hiện những lỗi trên hệ thống Snort.

e) *Đối số dòng lệnh*

Option	Description
-r <file>	Đọc file pcap
--pcap-single=<file>	Tương tự như -r nhưng cho thông tin đầy đủ hơn
--pcap-file=<file>	Tập tin có chứa một danh sách các pcaps để đọc. Có thể chỉ định đường dẫn đến pcap hoặc thư mục recurse để pcaps
--pcap-list=" <list>"</list>	Một không gian tách biệt chứa danh sách các pcaps để đọc.
--pcap-dir=<dir>	Một thư mục recurse để tìm pcaps. Sắp xếp theo thứ tự ASCII.
--pcap-filter=<filter>	Bộ lọc shell áp dụng khi nhận được pcaps từ tập tin hoặc thư mục. Điều này sẽ áp dụng cho bất kỳ - file-pcap hoặc đối số - pcap-dir đi theo. Sử dụng --pcap-no-filter để xóa các bộ lọc sau --pcap-file hoặc đối số --pcap-dir hay chỉ định --pcap-filter một lần nữa để xóa bộ lọc trước để

	thiết lập theo sau --pcap-file hay các đối số --pcap-dir.
--pcap-no-filter	Thiết lập lại và không sử dụng các bộ lọc khi nhận được pcaps từ tập tin hoặc thư mục.
--pcap-reset	Nếu phải đọc nhiều pcaps, Snort sẽ khởi động lại trạng thái post-configuration trước khi đọc tới pcap tiếp theo. Theo mặc định nếu không thiết lập tùy chọn này, hệ thống sẽ không khởi động lại trạng thái hoạt động.
--pcap-show	Hiển thị một dòng pcap để thông báo những gì đang được đọc.

Bảng 4 – Lựa chọn mô tả

3. CẤU HÌNH SNORT

3.1. Các biến trong Snort

a) *Includes*

Tùy khóa include cho phép định nghĩa một file tập hợp các luật được chỉ định trong Snort command line. Chức năng hoạt động của nó cũng gần giống như khóa #include trong ngôn ngữ lập trình C, như đọc các tên biến, tên tập tin và bổ sung nội dung tại một số vị trí được quy định trong tập tin. Tập tin include sẽ thay các giá trị biến bằng một biến số được xác định:

```
include <include file path/name>
```

b) *Variables*

Ba loại biến có thể được xác định trong Snort:

- Var
- portvar
- ipvar

Đây là các biến thay thế đơn giản được thiết lập với các biến ipvar, var, hoặc từ khóa portvar như sau:

```
var RULES_PATH rules/  
portvar MY_PORTS [22,80,1024:1050]  
ipvar MY_NET [192.168.1.0/24,10.1.1.0/24]  
alert tcp any any -> $MY_NET $MY_PORTS (flags:S; msg:"SYN packet";)  
include $RULE_PATH/example.rule
```

3.1.1. Biến IP và Lists IP

Địa chỉ IP có thể được quy định một cách riêng biệt, hoặc trong một danh sách như một block CIDR, hoặc tổng hợp của cả ba phần. Nếu hệ thống hỗ trợ Ipv6, các biến IP cần được sử dụng bằng biến “ipvar” thay vì “var” như thông thường.

IPs, danh sách IP, và block CIDR được phủ định bằng kí hiệu '!'.

Ví dụ sau đây phù hợp với IP 1.1.1.1 và IP từ 2.2.2.0 đến 2.2.2.255, ngoại trừ các IP 2.2.2.2 và 2.2.2.3:

```
[1.1.1.1,2.2.2.0/24,!{2.2.2.2,2.2.2.3}]
```

Thứ tự của các phần tử trong danh sách không quan trọng. Các yếu tố "any" có thể được sử dụng để liên kết các IP, mặc dù "!any" không được dùng.

```
ipvar EXAMPLE [1.1.1.1,2.2.2.0/24,!{2.2.2.2,2.2.2.3}]
```

```
alert tcp $EXAMPLE any -> any any (msg:"Example"; sid:1;)
```

```
alert tcp [1.0.0.0/8,!1.1.1.0/24] any -> any any (msg:"Example";sid:2;)
```

3.1.2. Biến port và List Port

Danh sách port hỗ trợ việc khai báo và quản lý phạm vi các port. Biến, phạm vi và danh sách port bị phủ nhận bằng kí hiệu “!”. “any” được sử dụng để chỉ tất cả các port, nhưng “!any” thì không được dùng. Phạm vi của port có giới hạn từ 0 đến 65535.

Danh sách các port phải được đặt trong dấu ngoặc vuông, và khi muốn định nghĩa phạm vi của các port nằm trong giới hạn nào, ta phải dùng dấu “:”, Ví dụ như:

```
[10:50,888:900]
```

Biến của port được quy định bằng cách sử dụng “portvar”. Đôi với tính tương thích ngược, một “var” vẫn có thể được sử dụng để khai báo một biến port, cung cấp tên biến, kết thúc với “_PORT” hoặc bắt đầu với “PORT_”.

Một số Ví dụ cụ thể về khai báo biến và danh sách port hợp lệ:

```
portvar EXAMPLE1 80
```

```
var EXAMPLE2_PORT [80:90]
```

```
var PORT_EXAMPLE2 [1]
```

```
portvar EXAMPLE3 any
```

```
portvar EXAMPLE4 [!70:90]
```

```
portvar EXAMPLE5 [80,91:95,100:200]
```

```
alert tcp any $EXAMPLE1 -> any $EXAMPLE2_PORT (msg:"Example"; sid:1;)
```

```
alert tcp any $PORT_EXAMPLE2 -> any any (msg:"Example"; sid:2;)
```

```
alert tcp any 90 -> any [100:1000,9999:20000] (msg:"Example"; sid:3;)
```

Khai báo biến: Tên biến có thể được sửa đổi theo một vài cách khác nhau bằng cách dùng các khóa “?” hay “-”... Bạn có thể định nghĩa meta-variables bằng cách sử dụng dấu \$ điều hành.

Variable Syntax	Description
var	Định nghĩa một meta-variable.
\$(var) or \$var	Thay thế nội dung của một biến var.
\$(var:-default)	Thay thế các nội dung của var biến là "mặc định" nếu var không được xác định.
\$(var:?message)	Thay thế với những nội dung của biến var hoặc in ra thông báo lỗi và thoát.

Bảng 5 – Thông tin một số biến variable

Ví dụ:

```
ipvar MY_NET 192.168.1.0/24
```

```
log tcp any any -> $(MY_NET:?MY_NET is undefined!) 23
```

Cách cấu hình và khai báo tùy chọn dòng lệnh của Snort được quy định trong file cấu hình:

```
config <directive> [:<value>]
```

3.2. Cấu hình tiền xử lý

Module tiền xử lý được xây dựng để thực hiện nhiều nhiệm vụ. Nó bình thường hoá traffic cho các services, để chắc chắn rằng dữ liệu trong các packet qua nó là tốt nhất để module phát hiện tiếp tục xử lý. Chức năng khác của module tiền xử lý là tự phòng thủ bằng cách dự đoán những dấu hiệu tấn công qua việc so sánh các packet. Module tiền xử lý được chạy trước khi module phát hiện được gọi, nhưng sau khi gói tin đã được giải mã. Gói tin có thể được sửa đổi hay phân tích trong các cơ chế hoạt động của module này.

Tiến trình tiền xử lý được nạp và cấu hình bằng cách sử dụng các khoá xử lý. Các định dạng của chỉ thị tiền xử lý trong tập tin quy tắc Snort là:

```
preprocessor <name>: <options>
```

```
preprocessor minfrag: 128
```

3.2.1. Frag3

Khi một packet đi qua các mạng khác nhau, nó thường cần phân mảnh thành các packet nhỏ hơn do MTU qui định trên mỗi mạng. Sau đó tất cả các sắp xếp lại khi đến nơi để khôi phục dữ liệu gốc. Một trong những phương pháp tấn công của attacker là việc can thiệp và sửa đổi những packet phân mảnh này để phục vụ cho mục đích của chúng. Frag3 trong module tiền xử lý có chức năng quản lý những gói tin phân mảnh này, bằng cách sắp xếp và so sánh các gói này với nhau, nó sẽ dễ dàng phát hiện ra những thay đổi đó. Frag3 có các options để chống lại các dạng tấn công này.

Tiền xử lý frag3 là một module IP dựa trên mục tiêu chống phân mảnh cho Snort. Frag3 sử dụng cấu trúc dữ liệu và danh sách liên kết sfxhash cho xử lý dữ liệu nội bộ, cho phép nó có nhiều hiệu suất dự báo và tính quyết định hơn trong việc quản lý bất kỳ môi trường làm việc nào tìm kiếm nhiều sự phân mảnh hệ thống.

a) Cấu hình Frag 3

Có ít nhất hai chỉ thị tiền xử lý cần thiết để kích hoạt frag3: Global Configuration và Engine Configuration.

Global Configuration

- Tên cơ chế tiền xử lý: frag3_global
- Các tùy chọn: Chú ý: tùy chọn cấu hình toàn cầu được quy định bằng dấu phẩy.
 - *max_frags <number>* – số lượng tối đa các phân mảnh đồng thời được theo dõi. Mặc định là 8192.
 - *memcap <bytes>* – Bộ nhớ tự quản. Mặc định là 4MB.
 - *Prealloc_frags <number>* – Thay thế chế độ quản lý bộ nhớ. Sử dụng các nút phân đoạn trước khi thay thế. (Nhanh hơn trong một số trường hợp).

Engine Configuration

- Tên cơ chế tiền xử lý: frag3_engine
- Các tùy chọn: Chú ý: những tùy chọn được cấu hình trong không gian riêng biệt.
 - *timeout <seconds>* – thời gian timeout của phân mảnh. Những phân mảnh trong hệ thống tồn tại trong hệ thống sau thời gian này sẽ bị hủy. Mặc định là 60 giây.

- *ttl_limit < hops >* – Giá trị TTL tối đa chấp nhận cho gói tin đầu tiên trong phân mảnh. Mặc định là 5.
- *min_ttl < value >* – Giá trị TTL tối thiểu chấp nhận cho một gói phân mảnh. Mặc định là 1.
- *detect_anomalies* – Phát hiện những đoạn phân mảnh bất thường
- *bind_to < ip_list >* – Danh sách IP liên kết với hệ thống. Hệ thống chỉ chạy những gói tin với địa chỉ đích nằm trong danh sách IP. Giá trị mặc định là tất cả.
- *policy < type >* – Lựa chọn chính sách dựa trên tiêu chí chống phân mảnh. Những chính sách có sẵn là first, last, bsd, bsd-right, linux. Mặc định là bsd.

b) *Cấu hình để nghị*

```
preprocessor frag3_global: prealloc_nodes 8192
```

```
preprocessor frag3_engine: policy linux, bind_to 192.168.1.0/24
```

```
preprocessor frag3_engine: policy first, bind_to [10.1.47.0/24,172.16.8.0/24]
```

```
preprocessor frag3_engine: policy last, detect_anomalies
```

3.2.2. Stream5

Tiền xử lý stream5 là một module của snort, có khả năng theo dõi các traffic TCP và UDP đang lưu thông trên mạng bằng cách sử dụng các khoá rule “flow” và “flowbits”.

Thông điệp ICMP được theo dõi cho mục đích kiểm tra các gói tin unreachable và các dịch vụ message không có sẵn, như việc chấm dứt một phiên TCP hoặc UDP.

Stream5 được giới thiệu để xử lý các dữ liệu chồng chéo, các nhãn thời gian TCP, dữ liệu về SYN, FIN, các reset sequence number... và các giao thức TCP dị thường.

a) *Stream API*

Stream5 hỗ trợ Stream API, là một chuẩn giao thức tiền xử lý tự động được cấu hình để sắp xếp các hành vi theo yêu cầu của giao thức lớp ứng dụng, xác định các phiên có thể được bỏ qua (do dữ liệu truyền lớn, vv) và cập nhập các thông tin nhận

dạng về phiên (giao thức ứng dụng, direction, vv) mà sau này có thể được sử dụng bởi các rule.

Stream5 có chức năng phát hiện các giao thức TCP dị thường, như dữ liệu về các gói SYN, dữ liệu nhận được nằm bên ngoài cửa sổ TCP...được cấu hình thông qua option detect_anomalies trong TCP.

b) Stream5 Global Configuration

```
preprocessor stream5_global: \
[track_tcp <yes|no>], [max_tcp <number>], \
[memcap <number bytes>], \
[track_udp <yes|no>], [max_udp <number>], \
[track_icmp <yes|no>], [max_icmp <number>], \
[flush_on_alert], [show_rebuilt_packets], \
[prune_log_max <bytes>], [disabled]
```

Ví dụ: Đây là cấu hình mặc định trong snort.conf

```
preprocessor stream5_global: \
max_tcp 8192, track_tcp yes, track_udp yes, track_icmp no
preprocessor stream5_tcp: \
policy first, use_static_footprint_sizes
preprocessor stream5_udp: \
ignore_any_rules
```

Cấu hình này map hai network segments trong hai OS policy khác nhau, một cho windows và một cho linux, với các traffic đi đến policy mặc định của Solaris.

```
preprocessor stream5_global: track_tcp yes
preprocessor stream5_tcp: bind_to 192.168.1.0/24, policy windows
preprocessor stream5_tcp: bind_to 10.1.1.0/24, policy linux
preprocessor stream5_tcp: policy solaris
```

3.2.3. sfPortscan

Các module sfPortscan được thiết kế để phát hiện các giai đoạn đầu tiên trong một cuộc tấn công mạng: do thám. Trong giai đoạn trinh sát, kẻ tấn công xác định những loại giao thức mạng, dịch vụ hỗ trợ máy chủ lưu trữ bằng cách scan port của hệ thống. Giả định kẻ tấn công chưa xác định được các loại giao thức cũng như các dịch vụ hỗ trợ của máy chủ nạn nhân.

Trong một mạng truyền thông ổn định, những phản ứng bất thường của hệ thống rất hiếm khi xảy ra, nhất là khi chúng lặp lại trong một số thời điểm nhất định. Mục tiêu của việc phát hiện scan port là để phát hiện những dấu hiệu bất thường của hệ thống trước các cuộc tấn công.

Một trong những công cụ phổ biến nhất của phương pháp scan port hiện nay là sử dụng Nmap, công cụ hỗ trợ rất nhiều các kỹ thuật scan port. sfPortscan được thiết kế để có thể phát hiện các loại hình quét Nmap có thể xảy ra.

sfPortscan sẽ cảnh báo khi phát hiện các hình thức scan Nmap sau:

- TCP Portscan
- UDP Portscan
- IP Portscan

Một trong những cảnh báo gặp là khi xuất hiện các kiểu scan one → one. Một host có thể scan nhiều port trên các host khác. Các truy vấn port thường không hợp lệ, vì hầu hết các máy có rất ít các dịch vụ sẵn sàng.

sfPortscan cũng cảnh báo cho các loại decoy scanport, distributed Portscan và portsweeps sau:

- | | |
|---|---|
| <ul style="list-style-type: none">• TCP Decoy Portscan• UDP Decoy Portscan• IP Decoy Portscan• TCP distributed Portscan• UDP distributed Portscan | <ul style="list-style-type: none">• IP distributed Portscan• TCP Portsweep• UDP Portsweep• IP Portsweep• ICMP Portsweep |
|---|---|

Decoy portscans cũng giống như những portscans Nmap mô tả ở trên, kẻ tấn công giả địa chỉ nguồn và thực hiện việc scanport đến máy nạn nhân. Chiến thuật này giúp che giấu danh tính thực sự của kẻ tấn công.

Distributed portscan là hình thức sử dụng nhiều máy để scan đến một port. Hình thức này xảy ra khi nhiều máy truy vấn đến các dịch vụ đang mở trên một máy

chủ. Điều này được sử dụng để tránh bị một IDS phát hiện, xáo trộn lệnh và kiểm soát máy chủ.

Những thông báo xuất hiện khi một → nhiều portsweeps. Một máy chủ quét một port trên máy nhiều. Điều này thường xảy ra khi kẻ tấn công mới bắt đầu khai thác về tìm kiếm một dịch vụ có thể tấn công.

sfPortscan chỉ tạo ra một cảnh báo cho mỗi cặp máy chủ trong cửa sổ thời gian (nhiều hơn vào cửa sổ bên dưới). Trên những cảnh báo TCP, sfPortscan cũng sẽ hiển thị tất cả các port mở đã được quét. Tuy nhiên khi quét TCP, sfPortscan sẽ chỉ theo dõi các port đã mở sau khi cảnh báo đã được kích hoạt. Việc các port được mở không phải là những thông báo cá nhân, tuy nhiên cần dựa vào các dấu hiệu cảnh báo gốc đã biết.

Cấu hình sfPortscan

Việc sử dụng tiền xử lý stream5 được yêu cầu cho sfPortscan để xác định những kết nối không được đảm bảo như ICMP hay UDP.

Các thông số bạn có thể sử dụng để cấu hình các module portscan là:

a) *proto <protocol>*

Các tùy chọn có sẵn:

- TCP
- UDP
- IGMP
- ip proto
- all

b) *scan type <scan type>*

Các tùy chọn có sẵn:

- portscan
- portsweep
- decoy portscan
- distributed portscan

c) *sense level <level>*

Có sẵn các tùy chọn:

- “low” – cảnh báo chỉ được tạo ra với các gói tin lỗi được gửi từ các host mục tiêu, cài đặt này sẽ không bao giờ gây ra một cảnh báo lọc vì thiếu các tín hiệu lỗi. Thiết lập này được dựa trên một cửa sổ thời gian tĩnh trong 60 giây, sau đó được thiết lập lại.

- “*medium*” – sẽ tạo ra cảnh báo lọc khi theo dõi các kết nối. Thiết lập này có thể sai trên các host hoạt động (NAT, proxy, DNS cache, vv), vì vậy user khi triển khai có thể điều chỉnh bỏ qua các chỉ thị này.
- “*high*” – liên tục cảnh báo theo dõi các host trên một mạng, sử dụng một cửa sổ thời gian để đánh giá số liệu thống kê portscan cho host đó. Thiết lập “*high*” sẽ quét chậm do phải giám sát liên tục, nhưng rất nhạy cảm với các host hoạt động. Điều này sẽ yêu cầu user điều chỉnh sfPortscan.

d) *watch ip <ip1/ip2/cidr[:[port/port2-port3]]>*

Xác định những địa chỉ IP, mạng lưới, và các port cụ thể trên các máy để theo dõi. Danh sách này là một danh sách bằng dấu phẩy của địa chỉ IP, địa chỉ IP bằng cách sử dụng ký hiệu CIDR. Tùy chọn, port được quy định sau các địa chỉ IP/CIDR sử dụng một không gian và có thể là một port hoặc một dải ký hiệu bằng dấu gạch ngang. Các địa chỉ IP hoặc mạng lưới không thuộc phạm vi này sẽ được bỏ qua nếu tùy chọn này được sử dụng.

e) *ignore scanners <ip1/ip2/cidr[:[port/port2-port3]]>*

Bỏ qua các nguồn quét cảnh báo.

f) *ignore scanned <ip1/ip2/cidr[:[port/port2-port3]]>*

Bỏ qua các điểm đến của quét cảnh báo.

g) *logfile <file>*

Tùy chọn này sẽ xuất port được scan đến các tập tin được chỉ định. Nếu tập tin không chứa một dấu gạch chéo đầu hàng, tập tin này sẽ được đặt trong thư mục cấu hình Snort.

h) *include midstream*

Tùy chọn này sẽ bao gồm các đặc điểm của Stream5. Điều này có thể dẫn đến các cảnh báo giả, đặc biệt là việc load những dữ liệu nặng có thể làm drop các gói, đó là lý do lựa chọn này được tắt theo mặc định.

i) *detect ack scans*

Tùy chọn này sẽ bao gồm các đặc điểm được tổng hợp lại giữa các dòng module cần thiết để phát hiện quét các gói ACK. Tuy nhiên, điều này có thể dẫn đến các cảnh báo giả, đặc biệt là việc load những dữ liệu nặng có thể làm drop các gói, đó là lý do lựa chọn này được tắt theo mặc định.

Ví dụ:

```
preprocessor flow: stats_interval 0 hash 2
preprocessor sfportscan:\n
proto { all } \
scan_type { all } \
sense_level { low }
```

3.2.4. rpc_decode

RPC's traffic có thể phân chia ra nhiều packet và mã hoá theo nhiều kiểu khác nhau. Rpc_decode preprocessor sẽ bình thường hóa các gói này vào một record duy nhất không bị phân mảnh (packet buffer). Nếu stream5 được kích hoạt, nó sẽ chỉ xử lý trên các traffic của máy client. Mặc định nó sẽ hoạt động trên port 111 và 32771.

Cấu trúc:

```
preprocessor rpc_decode: <ports> [ alert_fragments ] \
```

Có 4 option cho rpc_recode:

Option	Mô tả
alert_fragments	Báo động khi có fragmented RPC record
no_alert_multiple_requests	Không báo động khi nhiều hơn một RPC record được chứa trong một packet.
no_alert_large_fragments	Không báo động khi tổng dung lượng size của các fragment vượt quá giới hạn của một gói.
no_alert_incomplete	Không báo động khi dung lượng size của một fragment vượt quá giới hạn của một gói.

Cấu trúc:

```
preprocessor rpc_decode: \
<ports> [ alert_fragments ] \
[no_alert_multiple_requests] \
[no_alert_large_fragments] \
[no_alert_incomplete]
```

3.2.5. Performance Monitor

Yếu tố then chốt của module tiền xử lý là quản lý thời gian thực và hiệu suất xử lý tối đa. Bất cứ khi nào module tiền xử lý hoạt động, nó cũng kích hoạt một chế độ output, như in ra các số liệu thống kê lên màn hình console hoặc xuất ra một file dữ liệu thống kê với tên chỉ định. Theo mặc định, số liệu thống kê thời gian thực của Snort sẽ được xử lý.

Điều này bao gồm:

- Time Stamp
- Drop Rate
- Mbits/Sec
- Alerts/Sec
- K-Pkts/Sec
- Avg Bytes/Pkt
- Syns/Sec
- SynAcks/Sec
- New Sessions Cached/Sec
- Sessions Del fr Cache/Sec
- v.v...

Các tùy chọn sau đây có thể được sử dụng với Performance Monitor:

- *flow* – In ra các thông kê về tình trạng của các traffic và các protocol mà Snort là nhìn thấy. Tùy chọn này có thể xuất một lượng lớn output.
- *event* – Kích hoạt chức năng reporting và hiển thị trạng thái số lượng các signature match phù hợp.
- *max* – Thiết lập cho Snort hoạt động tối đa để tăng tốc độ và hiệu suất xử lý.
- *console* – In số liệu thống kê ra màn hình console.
- *file* – In số liệu thống kê trong một tập tin định dạng được phân định bằng dấu
- *pktcnt* – Điều chỉnh số lượng các gói dữ liệu xử lý trước khi kiểm tra thời gian sample. Mặc định là 10000.
- *time* – thời gian chờ (giây) các khoảng thời gian xử lý
- *accumulate or reset* – Xác định kiểu dữ liệu bị hủy được lưu giữ bởi hệ điều hành. Theo mặc định, tiến trình reset được sử dụng.
- *atexitonly* – thông kê dump cho toàn bộ hoạt động của Snort.
- *max_file_size* – Xác định kích thước tối đa của comma-delimited file.

Ví dụ:

```
preprocessor perfmonitor: \
    time 30 events flow file stats.profile max console ptcnt 10000
preprocessor perfmonitor: \
    time 300 file /var/tmp/snortstat ptcnt 10000
```

3.2.6. HTTP Inspect

HTTP Inspect là một bộ giải mã HTTP chung cho các ứng dụng, nó sẽ giải mã các buffer, tìm và bình thường hóa cái trường HTTP. HTTP Inspect hoạt động trên cả client request và server response.

Tính năng của HTTP Inspect rất phong phú. Người dùng có thể cấu hình các HTTP server cá nhân với nhiều tùy chọn trên bất kỳ các loại web server. Có 2 kiểu cấu hình http_inspect là global và server.

a) Global Configuration

Cấu trúc

```
preprocessor http_inspect: global \
    iis_unicode_map <map_filename> \
    codemap <integer> \
    [detect_anomalous_servers] \
    [proxy_alert]
```

Có 3 option dùng để config http traffic:

- *iis_unicode_map <map filename> [codemap <integer>]*
 - Đây là một bản đồ unicode toàn cầu chứa các tham số cấu hình cần thiết. iis_unicode_map chỉ đường dẫn tới map file unicode.map và giúp cho module tiền xử lý biết làm cách nào để map các kí tự unicode sang ASCII dạng text để match các signatures. Ta phải chỉ ra code để map, đối với US server, codemap thường là 1252.
- *detect_anomalous_servers*

- Theo dõi và cảnh báo nếu phát hiện những traffic HTTP trên các port không cho phép. Đểng bật tính năng này nếu bạn không có một máy chủ server cấu hình mặc định cho phép những user khác truy cập.
- *proxy_alert*
 - Nếu bạn sử dụng proxy server cho các user kết nối internet, bật chức năng này để giám sát và đưa ra cảnh báo khi một user ra ngoài mà không thông qua proxy.
- *compress_depth <integer>*
 - Tuỳ chọn này xác định số lượng tối đa các payload packet sẽ được giải nén. Giá trị mặc định là 1460 và nằm trong khoảng từ 1 đến 65535.
- *decompress_depth <integer>*
 - Xác định giới hạn tối đa dữ liệu có thể được giải nén từ một gói đã được nén. Giá trị này được đặt trong khoảng từ 1 đến 65535, mặc định là 2920.
- *max_gzip_mem*
 - Tuỳ chọn này sẽ xác định tối đa bộ nhớ mà tiền xử lý HTTP inspect sẽ sử dụng cho việc giải nén. Giá trị này được thiết lập từ 3276 byte đến 100MB. Giá trị mặc định là 838860.
- *disabled*
 - Tuỳ chọn này sẽ vô hiệu hoá tiền trình tiền xử lý, chỉ có tuỳ chọn “max_gzip_mem”, “compress_depth” và “decompress_depth” được sử dụng. Ví dụ:


```
preprocessor http_inspect: global iis_unicode_map unicode.map 1252
```

b) Server Configuration

Có 2 kiểu cấu hình chính là: “default” và “by IP address”:

- *default*: cấu hình mặc định cho tất cả các server khi bạn không có những cấu hình riêng, hầu hết các web server thường sử dụng cấu hình mặc định
 - Ví dụ:


```
preprocessor http_inspect_server: server default profile all ports { 80 }
```

- *IP address*: tương tự như “default” nhưng ở đây bạn cần xác định một IP address hay một dãy các IP cụ thể.
 - Ví dụ:

```
preprocessor http_inspect_server: server 10.1.1.1 profile all ports { 80 }
```

c) Server Configuration Options

Một số tùy chọn có thể được sử dụng:

- *profile <all/apache/iis >*
 - User có thể cấu hình HTTP Inspect bằng cách sử dụng các cấu hình HTTP server đã được xác định trước, cho phép user dễ dàng cấu hình các tiến trình tiền xử lý phù hợp cho mỗi loại server. Cấu hình profile có thể dùng apache, iis hoặc cả hai.
- *ports {<port> [<port><....>]}*
 - Khai báo các cổng dùng để mã hoá trên HTTP server. Tuy nhiên, các traffic HTTP được mã hoá sẽ không được giải mã với HTTP inspect. Để bỏ qua các traffic HTTP, hãy sử dụng tiền xử lý SSL
- *no_alerts*
 - Tuỳ chọn này sẽ tắt tất cả các cảnh báo được tạo bởi module tiền xử lý HTTP inspect. Điều này không ảnh hưởng tới các rule trong HTTP

Ví dụ:

```
preprocessor http_inspect_server: server 10.1.1.1 \
profile all \
ports { 80 3128 8080 } \
no_alerts
```

3.2.7. FTP/Telnet Preprocessor

FTP/Telnet là một cải tiến cho các bộ giải mã Telnet, cung cấp khả năng kiểm tra trạng thái stateful trên các luồng dữ liệu FTP và Telnet. Chức năng của FTP/Telnet Preprocessor là giải mã các luồng data, xác định các lệnh FTP, hồi đáp các tín hiệu, thoát khỏi Telnet và bình thường hóa các trường. FTP/Telnet hoạt động trên cả client

request và server response. FTP/Telnet có khả năng xử lý các tiến trình stateless, nghĩa là nó chỉ tìm kiếm thông tin trên cơ sở packet – by – packet.

Mặc định FTP/Telnet hoạt động trong chế độ kiểm tra của trạng thái stateful, có nghĩa nó sẽ tìm kiếm thông tin và xử lý tập hợp dữ liệu lại một cách chính xác.

3.2.8. SSH

Tiền xử lý SSH có khả năng phát hiện những khai thác sau: Gobbles, Challenge-Response Buffer Overflow, CRC 32, Secure CRT, và Protocol Mismatch exploit. Để phát hiện các dấu hiệu tấn công, các tiền xử lý SSH đếm số byte truyền đến server. Nếu số byte vượt quá giới hạn cho phép được xác định trước của các gói tin, cảnh báo được tạo ra. Theo mặc định, tất cả các cảnh báo đều được bật và tiến trình tiền xử lý sẽ kiểm tra các traffic trên port 22.

3.2.9. DNS

Tiền trình tiền xử lý giải mã các response DNS và có thể phát hiện các khai thác sau: DNS Client RDataOverflow, Obsolete Record Types, và Experimental Record Types. DNS tìm kiếm các response DNS trên các traffic UDP và TCP sau đó kích hoạt tiến trình tiền xử lý để giải mã TCP. Theo mặc định, tất cả cảnh báo sẽ bị tắt và tiến trình tiền xử lý sẽ kiểm tra trên port 53.

3.2.10. ARP Spoof Preprocessor

Arpspoof được thiết kế cho tiến trình tiền xử lý để giải mã các gói ARP và phát hiện các cuộc tấn công ARP, các request ARP và các yêu cầu mapping IP không đồng nhất.

Khi không có đối số được quy định trong arpspoof, tiền xử lý sẽ kiểm tra địa chỉ Ethernet và các địa chỉ trong gói ARP. Khi phát hiện mâu thuẫn, một cảnh báo với GID 112 và SID 2 hoặc 3 được tạo ra.

"unicast-" được quy định như là đối số của arpspoof, tiền xử lý sẽ kiểm tra các yêu cầu unicast ARP. Một cảnh báo với GID 112 và SID 1 sẽ được tạo ra nếu một yêu cầu ARP unicast được phát hiện.

Địa chỉ IP và địa chỉ phần cứng MAC là các đối số của arpspoof detect host. Snort sẽ lưu giữ các đối số này vào một danh sách và sử dụng danh sách này khi phát hiện các cuộc tấn công ARP cache overwrite.

Cấu trúc

preprocessor arpspoof[: -unicast]

preprocessor arpspoof_detect_host: ip mac

3.3. Cấu hình luật giải mã và tiền xử lý

Decoder và Preprocessor Rules dùng để tắt hay kích hoạt việc sử dụng các luật cho những tiền trình giải mã và tiền xử lý. Ví dụ nếu cấu hình disable_decode_alerts trong file Snort.conf, tiền trình giải mã sẽ bị disable cho dù xuất hiện những tình huống tương ứng với các luật.

Lưu ý rằng tùy chọn enable_decode_drops sẽ được ưu tiên hơn các tùy chọn khác khi hệ thống đang hoạt động ở chế độ Snort Inline

Cấu hình

Tùy chọn sau sẽ kích hoạt Decoder and Preprocessor Rules.

`./configure --enable-decoder-preprocessor-rules`

Decoder and preprocessor rules được lưu trong thư mục preproc rules/ directory với tên decoder.rules and preprocessor.rules respectively. File này sẽ được update khi có những sự kiện mới được thêm vào Snort.

Để kích hoạt các luật trong snort.conf, ta phải xác định đường dẫn đến nơi chứa các luật và bỏ các dòng include trong snort.conf để sử dụng các luật bao gồm các dòng trong snort.conf có tham khảo các tập tin quy tắc.

```
var PREPROC_RULE_PATH/path/to/preproc_rules
include $ PREPROC_RULE_PATH/preprocessor.rules
include $ PREPROC_RULE_PATH/decoder.rules
```

Để vô hiệu hóa các luật, cần thêm khóa # vào đầu các dòng luật hoặc xóa các luật này. Để thay đổi các luật hay các hành động, ta có thể sử dụng các tùy chọn:

- alert
- log
- pass
- drop
- sdrop
- reject

Ví dụ: Chuyển từ hành động cảnh báo sang hành động hủy:

```
alert ( msg: "DECODE_NOT_IPV4_DGRAM"; sid: 1; gid: 116; rev: 1; \
metadata: rule-type decode ; classtype:protocol-command-decode;)

to

drop ( msg: "DECODE_NOT_IPV4_DGRAM"; sid: 1; gid: 116; rev: 1; \
metadata: rule-type decode ; classtype:protocol-command-decode;)
```

3.4. Cấu hình Module Output

Snort có thể output vào các file log hoặc output ra console, nhiều administrator thích dùng các phần mềm của hãng thứ 3 (third party) để tăng thêm chức năng giám sát của Snort, các phần mềm data database đều có thể dùng được, lưu ý trước khi cài đặt Snort muốn dùng database nào thì cần chỉ rõ khi cài đặt ví dụ dùng MySQL, khi biên dịch source thêm vào –mysql.

Ví dụ: ./configure -- mysql

Module Output được kích hoạt sau các Module tiền xử lý và phát hiện để xuất kết quả ra các file log hoặc ra màn hình console. Những output plugin có thể được qui định bổ sung trong file cấu hình Snort, khi nhiều plugin cùng loại được quy định (log, cảnh báo), chúng được xếp chồng lên nhau và được gọi theo một thứ tự khi xảy ra một sự kiện nào đó. Đối với những logging và cảnh báo tiêu chuẩn, output plugin sẽ được gửi tới /var/log/snort theo mặc định hoặc tới một thư mục do user quy định (sử dụng khóa –l trong command line)

Module output được nạp bằng cách sử dụng các khóa trong tập tin rules

- output <name>: <options>
- output alert_syslog: log_auth log_alert

3.4.1. alert_syslog

Module Output gửi các cảnh báo tới syslog facility(giống như sử dụng khóa –s trong command line), cho phép user xác định logging facility và giá trị ưu tiên trong file luật Snort để xử lý linh hoạt hơn trong những logging cảnh báo.

a) Cấu trúc

alert_syslog: <facility> <priority> <options>

```
output alert_syslog: [host=<hostname[:<port>],] <facility> <priority>
<options>
```

```
output alert_syslog: 10.1.1.1:514, <facility> <priority> <options>
```

b) Các khóa trong syslog

Facilities

log_auth	log_local1	log_local5
log_authpriv	log_local2	log_local6
log_daemon	log_local3	log_local7
log_local0	log_local4	log_use

Priorities

log_emerg	log_err	log_info
log_alert	log_warning	log_debug
log_crit	log_notice	

Options

log_cons	log_perror
log_ndelay	log_pid

3.4.2. alert_fast

Các cảnh báo trong Snort sẽ được in ra một cách nhanh nhất đến một tập tin được xác định. Đây là một phương pháp cảnh báo nhanh hơn phương pháp cảnh báo đầy đủ vì nó không cần in ra tất cả các packet header trong file output

Cấu trúc

```
alert_fast: <output filename>
```

```
output alert_fast: alert.fast
```

3.4.3. alert_full

Cảnh báo sẽ được in ra với đầy đủ các packet header. Mặc định thông tin sẽ được lưu trong /var/log/snort hoặc một thư mục được xác định. Snort sẽ tạo ra các thư

mục con chứa thông tin cảnh báo ứng với mỗi một IP, điều này làm cho hoạt động của Snort bị chậm lại do đó nó không được khuyến khích sử dụng.

Cấu trúc

```
alert_full: <output filename>
```

```
output alert_full: alert.full
```

3.4.4. alert_unixsock

Thiết lập một UNIX domain socket và gửi cảnh báo với nó. Ngoài các chương trình/tiến trình có thể nghe ở trên socket ta còn nhận được những cảnh báo và gói dữ liệu trong thời gian thực. Đây là hiện một giao diện thử nghiệm.

Cấu trúc

```
alert_unixsock
```

```
output alert_unixsock
```

3.4.5. log_tcpdump

Chức năng này sẽ ghi các bản log ở định dạng tcmdump. Do có rất nhiều công cụ có thể đọc được định dạng này nên nó rất hữu ích cho việc tổng hợp và phân tích thông tin với số lượng lớn. Đối số của log_tcpdump là đầu ra tên của tập tin.

Cấu trúc

```
log_tcpdump: <output filename>
```

```
output log_tcpdump: snort.log
```

3.4.6. Database

Database là nơi chứa tất cả các thông tin lưu trữ của Snort như các cảnh báo, các host liên quan, các log packet gây ra các cảnh báo,...Snort sử dụng các đối số để quản lý database và giúp cho các admin có thể quản lý thông tin một cách có hiệu quả hơn.

Cấu trúc của câu lệnh output ra database

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

Các thông số có sẵn:

- *host* – Địa chỉ IP của database server
- *port* – Port đang kết nối tới server hoặc là tên socket đang kết nối tới UNIX domain
- *dbname* – Database name
- *user* – username được Snort xác định để log trong database.
- *password* – Password được user sử dụng để log vào database
- *sensor name* – Tên của sensor. Nếu bạn không cấu hình, tên sẽ tự động được tạo ra
- *encoding* – Vì thông tin và các option của dữ liệu là dạng nhị phân nên chúng cần được mã hóa dưới các định dạng khác để có thể lưu trữ trong database, như kiểu hex, base64, ascii...
- *detail* – xác định các mức để log vào database, gồm có các kiểu như full, fast. Mặc định là full.

4. RULE SNORT

4.1. Giới thiệu

Sử dụng Snort rule khá đơn giản và linh hoạt nhưng nó chưa đựng một sức mạnh không nhỏ. Một số nguyên tắc cần phải biết và nhớ để phát huy hết sức mạnh của rule trong Snort.

Các phiên bản trước đây, hầu hết các rule điều viết 1 dòng nhưng từ phiên bản 1.8 trở lên, một rule ta có thể viết nhiều dòng (bằng cách sử dụng “\” để xuống dòng).

Snort rules có 2 phần: rule header và rule options.

Rule header gồm có:

Actions	Protocol	IP address	Port	Hướng	IP address	Port
---------	----------	------------	------	-------	------------	------

Rule options: gồm có tên cảnh báo và thông tin chứa trong gói dữ liệu cần kiểm tra.

Ví dụ:

```
alert tcp any any -> 192.168.1.0/24 111 (content:"|00 01 86 a5|"; msg:"mountd access";)
```

rule header: alert tcp any any -> 192.168.1.0/24 111

rule options: (content:"|00 01 86 a5|"; msg:"mountd access";)

4.2. Rules Headers

4.2.1. Rule Actions

Khi hệ thống Snort so sánh những gói tin với những điều luật mà ta đã khai báo trước đó, nếu những gói tin thỏa những điều kiện mà điều luật đã đặt ra thì sẽ thực hiện những hành động tương ứng với điều luật đã thỏa. Ở các phiên bản 1.x, nếu Snort có nhiều rule phù hợp với gói tin bắt được thì chỉ có rule đầu tiên được áp dụng. Nhưng từ các phiên bản 2.x, tất cả các rule phù hợp với gói tin bắt được đều được áp dụng trước khi đưa ra cảnh báo. Mặc định Snort có 5 hành động: alert, log, pass, activate và dynamic. Nếu ta cấu hình trong inline mode thì có thêm 3 hành động: drop, reject và sdrop.

- *alert* – Dùng để cảnh báo và ghi nhận lại gói tin khi phù hợp với điều kiện của rule. Cảnh báo có thể được gửi nhiều cách (Cảnh báo có thể lưu trong file hoặc hiện trên màn hình).
- *log* – Dùng ghi nhận lại gói tin. Gói tin có thể được ghi nhận bằng cách khác nhau. Thông điệp có thể được trong file hay là database. Gói tin có thể được ghi nhận với các cấp độ chi tiết khác nhau dựa trên command line arguments và file cấu hình.
- *pass* – Bỏ qua gói tin.
- *activate* – Dùng để tạo cảnh báo và kích hoạt rule khác để kiểm tra các điều kiện khác. Activate được sử dụng khi chúng ta cần kiểm tra thêm một hoặc các gói tin nhận được. Dynamic rule sử dụng kèm với activate.
- *dynamic* – Chờ activate rule kích hoạt và hoạt động như log rule. Dynamic rule chỉ có thể kích hoạt bởi activate.

Ví dụ: activate/dynamic rule

```
activate tcp !$HOME_NET any -> $HOME_NET 143 (flags: PA; content: "|E8C0FFFFF/bin"; activates: 1; msg: "IMAP buffer overflow!");
```

```
dynamic tcp !$HOME_NET any -> $HOME_NET 143 (activated_by: 1; count: 50);
```

- *drop* – iptables hủy bỏ gói tin và ghi nhận gói tin.
- *reject* – iptables hủy bỏ gói tin và ghi nhận gói tin, và gửi thông báo từ chối đến máy nguồn.
- *sdrop* – iptables hủy bỏ gói tin và không ghi nhận gói tin.
- Luật do nhà quản trị thiết kế.

Chúng ta có thể tự thiết kế những hành động riêng biệt cho những mục đích khác nhau như:

- Gửi thông điệp tới syslog.
- Thực hiện nhiều hành động trên một gói tin.
- Thông điệp ghi nhận nằm ở một database. Snort có thể ghi thông điệp tới MySQL, POSgress SQL, Oracle và Microsoft SQL server.

Các qui định về các hành động của Snort được viết trong file snort.conf. Một hành động mới được định nghĩa như sau:

```
ruletype ten_hanh_dong
```

```
{      action definition }
```

Ví dụ:

<pre>ruletype smb_db_alert { type alert output alert_smb: workstation.list</pre>	<pre> output database: log, mysql, user=rr password=rr dbname=snort host=localhost }</pre>
---	---

4.2.2. Protocol

Protocol là phần thứ hai của rules Header. Phần protocol của Snort rule sẽ chỉ ra giao thức của gói tin đang chạy sẽ được áp dụng. Snort hỗ trợ các giao thức: TCP, UDP, ICMP và IP. Trong tương lai sẽ hỗ trợ thêm nhiều giao thức như: ARP, IGRP, GRE, OSPF, RIP, IPX,...

Nếu khai báo giao thức là IP thì Snort sẽ kiểm tra link layer header để xác định kiểu của gói tin. Nếu các giao thức khác được khai báo thì Snort sẽ kiểm tra IP header để xác định kiểu giao thức.

4.2.3. IP address

Snort không hỗ trợ cơ chế phân giải tên miền thành IP. Snort chỉ quản lý địa chỉ thông qua IP address và subnetmask được viết tắt lại theo kiểu Classless Inter-Domain Routing (CIDR).

Ví dụ: 192.168.10.1/24; 172.16.1.1/16, 192.168.1.3/32 (Chỉ ra một máy mang địa chỉ 192.168.1.3).

Muốn phủ định một địa chỉ IP ta thêm “!” trước IP đó.

Ví dụ: !192.168.10.1/24.

Muốn khai báo một nhóm source IP hoặc Destination IP ta viết như sau:

[192.168.10.10/24, 172.16.1.1/16]

Dùng “,” để phân biệt 2 nhóm địa chỉ

4.2.4. Port number

Port number là những con số đại diện cho những dịch vụ mà con người đã qui định.

Ví dụ: port 23: Telnet; port 53: DNS; port 80: http;....

Khai báo 1 dãy port number được viết như sau: port bắt đầu: port kết thúc

Ví dụ: 50:90 ;:2000(nhỏ hơn hoặc bằng port 2000).

Phủ định một port hoặc một dãy port được viết như sau: !port hoặc ! [port bắt đầu: port kết thúc].

Ví dụ: !23; !21; !1:200; !10000: (bằng hoặc lớn hơn port 10000).

4.2.5. The Direction Operator

IP address và port number nằm bên trái Direction Operator được gọi nguồn ; IP address và port number nằm phía bên phải Direction Operator được gọi là đích đến.

Direction Operator gồm có 2 loại:

- ->: Hoạt động 1 chiều, có cấu trúc như sau: một IP address (hoặc nhiều IP address) và một port (hoặc nhiều port) -> một IP address (hoặc nhiều IP address) và một port (hoặc nhiều port).

Ví dụ: log tcp 192.168.10.1/24 any -> any 80.

- < >: giống như “->” nhưng hoạt động ở hai chiều

Ví dụ: log tcp any any <> 192.168.10.1/24 23

Ghi chú: không có “<-”, vì khi chúng ta đọc luật, ta đọc từ trái qua phải.

4.3. Rule Options

Sức mạnh, sự linh hoạt của Snort đều nằm ở Rule options nên nó một vai trò vô cùng quan trọng đối với Snort. Mỗi rule options 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.

4.4. General

4.4.1. 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”;

Ví dụ: msg: “phat hien tan cong mail Server”

4.4.2. 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>;]

4.4.3. sid

Dùng để xác định điều luật cụ thể của Snort, được sử dụng chung với rev keyword

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

Cấu trúc: sid: <snort rules id>;

4.4.4. rev

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

Cấu trúc: rev: <revision integer>;

4.4.5. 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 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;)

4.4.6. priority

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

Cấu trúc: priority: <priority integer>;

Ví dụ:

```
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)
```

Cùng một rule nhưng khác nhau về độ ưu tiên.

4.5. Payload Detection

4.5.1. 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. Từ phiên bản 1.x thì không hỗ trợ giao thức tầng ứng dụng, tùy chọn content thì sử dụng chung với offset keyword để tìm thông tin trong phần application layer header.

content khá phức tạp, nội dung thông tin 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 “|”. Nhiều content rules có thể viết trong một rule

Ví dụ:

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

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

- Cùng một mục đích là tìm từ GET nhưng viết bằng 2 cách khác nhau:
- Rule thứ nhất viết dưới dạng ASCII.
- Rule thứ hai viết dưới dạng thập lục phân. Chuyển từ mã thập lục phân sang ASCII thì: số 47 là G, 45 là E, và 54 là T.

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.

Nếu như có “!” thì sự cảnh báo sẽ được kích hoạt nếu gói tin không chứa nội dung được khai báo. “!” rất hữu ích khi chúng ta muốn cảnh báo một gói tin không phù hợp với mô hình ta thiết kế.

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

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

4.5.2. 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;)

4.5.3. 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;)

4.5.4. depth

Xác định số byte đầu tiên của gói tin phải kiểm tra. depth keyword nằm phía sau tùy chọn content.

Cấu trúc: depth: <number>;

4.5.5. offset

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

Cấu trúc: offset: <number>;

4.5.6. distance

Tùy chọn này tương tự tùy chọn offset, điểm khác biệt 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>;

4.5.7. 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>;

4.5.8. 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

4.5.9. 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;)

4.5.10.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]";

4.5.11. byte_test

Kiểm tra một trường byte đổi 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];
```

4.5.12. byte_jump

Tùy chọn này thực hiện lấy một số byte, biến đổi chúng thành dạng số, thiết lập con trỏ trả 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];

```
Ví dụ: alert udp any any -> any 32770:34000 (content: "|00 01 86 B8|"; \
content: "|00 00 00 01|"; distance: 4; within: 4; byte_jump: 4,12,relative, align; \
byte_test: 4, >, 900, 20, relative; msg: "statd format string buffer overflow";)
```

4.5.13. ftpbounce

Dùng để phát hiện các tấn công FTP bounce.

Cấu trúc: ftpbounce;

4.6. Non-Payload Detection Rule Options

4.6.1. 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;)
```

4.6.2. ttl

Để kiểm tra thời gian sống của IP

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

Ví dụ: ttl: <3; ttl:3-7

4.6.3. tos

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

Cấu trúc: tos:[!]<number>;

4.6.4. 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)

4.6.5. ipopts

Kiểm tra IP cụ thể với các tùy chọn:

Các tùy chọn có thể được kiểm tra

rr – Record Route	esec – IP Extended Security
eol – End of list	lsrr – Loose Source Routing
nop – No Op	ssrr – Strict Source Routing
ts – Time Stamp	satid – Stream identifier
sec – IP Security	any – any IP options are set

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

Ví dụ: ipopts:lsrr;// Tùy chọn IP là Loose Source Routing

4.6.6. 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+;// bit phân mảnh và bit không phân mảnh được thiết lập.

4.6.7. 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;// kích thước nằm giữa 300 và 400 bytes

4.6.8. 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)	U – URG
S – SYN	1 – Reserved bit 1 (MSB trong TCP Flags)
R – RST	2 – Reserved bit 2
P – PSH	0 – No TCP Flags Set
A – ACK	

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;)// nếu SYN và FIN được thiết lập, ignoring reserved bit 1 and reserved bit 2.

4.6.9. 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)];
```

4.6.10. flowbits

Dùng để theo dõi các máy tính trong suốt quá trình thực hiện phiên giao thức trao đổi. Tùy chọn flowbits hữu ích 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, bắt chấp các tùy chọn phát hiện còn lại.

4.6.11. seq

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

Cấu trúc: seq:<number>;

Ví dụ: seq:0;// số TCP sequence là không.

4.6.12. ack

Dùng để kiểm tra số TCP acknowledge.

Cấu trúc: ack: <number>;

Ví dụ: ack:0;// số TCP acknowledge là 0.

4.6.13.window

Dùng để kiểm tra TCP windows size

Câu trúc: window:[!]<number>;

Ví dụ: window:55808;// TCP window size là 55808.

4.6.14.itype

Dùng để kiểm tra giá trị của trường type trong thông báo ICMP với một giá trị cụ thể.

Câu trúc: itype:[<|>]<number>[<><number>];

Ví dụ: itype:>30;// kiểu ICMP lớn hơn 30.

4.6.15.icode

Kiểm tra giá trị của trường mã trong thông báo ICMP với một giá trị cụ thể.

Câu trúc: icode: [<|>]<number>[<><number>];

Ví dụ: code:>30;// mã ICMP lớn hơn 30.

4.6.16.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.

Câu trúc: icmp_id:<number>;

Ví dụ: icmp_id:0;// ICMP ID of 0.

4.6.17.icmp_seq

Dùng để kiểm tra số thứ tự (sequence value) cụ thể của giao 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.

Câu trúc: icmp_seq:<number>;

Ví dụ: icmp_seq:0; //ICMP Sequence of 0.

4.6.18.rpc

Dùng để kiểm tra ứng dụng, số phiên bản, số thủ tục(procedure) của RPC trong SUNRPC CALL được yêu cầu. Wildcards thì phù hợp cho các phiên bản và số thủ tục (procedure) bằng cách sử dụng “*”.

Cấu trúc:

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

Ví dụ: alert tcp any any -> any 111 (rpc: 100000,*,3;)// GETPORT

4.6.19.ip_proto

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

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

Ví dụ: alert ip any any -> any any (ip_proto:igmp;)// dùng để nhìn luồng IGMP

4.6.20.sameip

Dùng để kiểm tra source IP và destination IP có giống nhau không.

Cấu trúc: sameip;

Ví dụ: alert ip any any -> any any (sampeip;)

4.6.21.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

4.7. Post-Detection Rule Options

4.7.1. logto

Ghi nhận tất cả các gói và xuất ra file.

Cấu trúc: logto:"filename";

4.7.2. session

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

Cấu trúc: session: [printable|all];

Ví dụ: log tcp any any <> any 23 (session:printable;)

4.7.3. 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>]];

Ví dụ: alert tcp any any -> any any (resp:rst_all;).

4.7.4. react

Mặc định thì Snort sẽ đóng các kết nối, khóa vài vị trí hoặc dịch vụ và gửi cảnh báo đến trình duyệt 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 <port nr> – Dùng port proxy 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;)

4.7.5. tag

Cho phép ghi lại các gói tin mà liên quan tới nguyên tắc đã được thực thi trước đó.

Cấu trúc: tag: <type>, <count>, <metric>, [direction];

- Các kiểu:
 - session – Ghi nhận các gói tin trong phiên khi luật đã được thiết lập.
 - host – Ghi nhận các gói tin từ host cụ thể.
- count là quy định như một con số của units. Units thì được quy định trong <metric>field.
- Metric:
 - packets – Tính bằng gói tin.
 - seconds – Tính bằng giây.
 - bytes – Tính bằng byte.
- direction – Chỉ liên quan nếu kiểu host được sử dụng.
 - src – Nhãn gói tin chứa địa chỉ IP nguồn có sự kiện phát sinh ban đầu.
 - dst – Nhãn gói tin chứa địa chỉ IP đích có sự kiện phát sinh ban đầu.

4.7.6. activates

Cho phép kích hoạt một rule khi có sự bất thường xảy ra trên mạng.

Cấu trúc: activates: <number>;

4.7.7. activated_by

Cho phép bật điều luật dynamically khi điều luật cụ thể activate được kích hoạt

Cấu trúc: activated_by: <number>;

4.7.8. count

Phải được sử dụng chung với activated_by.

Câu trúc: count: <number>;

4.7.9. replace

Được sử dụng khi hệ thống là IPS, dùng để thay thế nội dung gói tin bắt được, nhưng chiều dài nội dung gói tin thay thế phải bằng với chiều dài nội dung gói tin bắt được.

Câu trúc: replace:”string”;

4.7.10. detection_filter

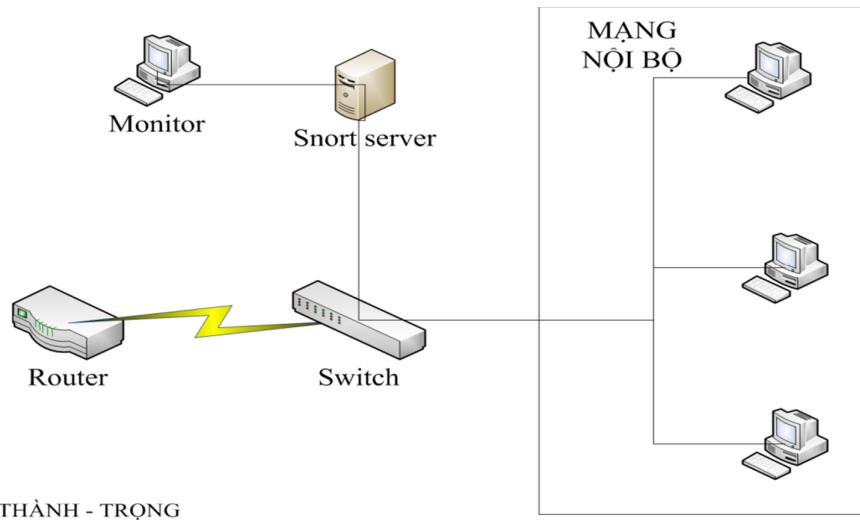
Dùng để kiểm đếm số lượng ip source hay ip destination(gói tin) nhất định trước khi rule được kích hoạt.

Câu trúc: detection_filter: track <by_src|by_dst>, count <c>, seconds <s>;

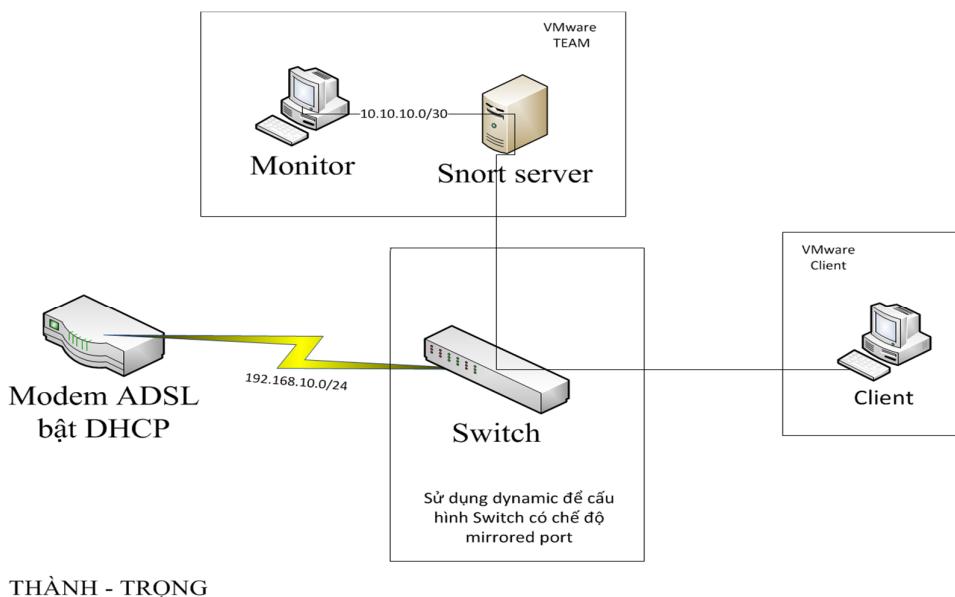
5. MÔ PHỎNG

5.1. Cấu hình Snort IDS:

5.1.1. Mô hình:



Vì hạn chế về mặt thiết bị nên chúng tôi đã xây dựng một mô hình bài lab như sau:



5.1.2. Các bước cài đặt

a) Bước 1:

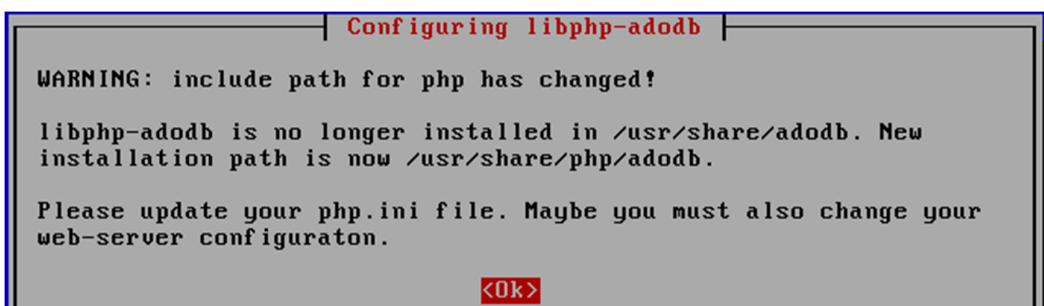
Trước khi cài snort trên Ubuntu server, ta phải cài một số phần mềm mà Ubuntu server cần cung cấp để hỗ trợ hoạt động của snort:

Cập nhập các cấu hình hệ thống

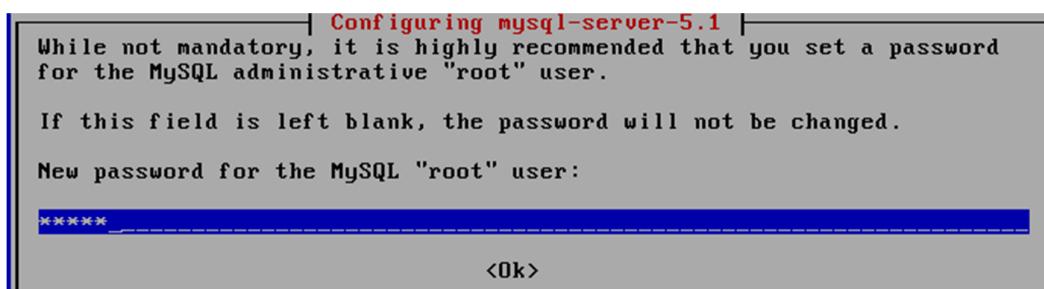
apt-get update apt-get upgrade

Cài đặt MySQL

apt-get install mysql-server



Nhấn ok



Ta nhập và xác nhận password MySQL của root.

Cài đặt các gói hỗ trợ trên mạng

apt-get install apache2

apt-get install php5

apt-get install php5-mysql

apt-get install php5-gd

apt-get install libpcap0.8-dev

apt-get install g++

apt-get install bison

apt-get install flex

apt-get install libmysqlclient16-dev

apt-get install libpcap-ruby

apt-get install php-pear

pear install --force Image_Color

pear install --force Image_Canvas

pear install --force Image_Graph

Sau khi cài đặt thành công các phần mềm trên , ta tải thêm các phần mềm khác từ các website khác :

barnyard2-1.9: <http://www.securixlive.com/download/barnyard2/barnyard2-1.9.tar.gz>

base-1.4.5: <http://nchc.dl.sourceforge.net/project/secureideas/BASE/base-1.4.5/base-1.4.5.tar.gz>

libpcap-1.0.0: <http://www.tcpdump.org/release/libpcap-1.0.0.tar.gz>

libdnet-1.11:<http://nchc.dl.sourceforge.net/project/libdnet/libdnet/libdnet-1.11/libdnet-1.11.tar.gz>

pcre-8.12: <http://nchc.dl.sourceforge.net/project/pcre/pcre/8.12/pcre-8.12.tar.gz>

snortreport-1.3.1: <http://www.symmetrixtech.com/ids/snortreport-1.3.1.tar.gz>

snort-2.9.0.5: <http://www.snort.org/dl/snort-current/snort-2.9.0.5.tar.gz>

daq-0.5 : <http://www.snort.org/dl/snort-current/daq-0.5.tar.gz>

snort-rules ta phải đăng ký một tài khoản của trang chủ snort :

<http://www.snort.org/snort-rules>

jpgraph-1.27.1: <http://hem.bredband.net/jpgraph/jpgraph-1.27.1.tar.gz>

b) Bước 2: Tiến hành cài đặt

Ta di chuyển đến vùng chứa source mà bước 1 đã tải về

- Cài đặt Libpcap

```
# tar zxvf libpcap-1.0.0.tar.gz
```

```
# cd libpcap-1.0.0
```

```
# ./configure && make && make install
```

- Cài đặt PCRE

```
# tar zxvf pcre-8.12.tar.gz
```

```
# cd pcre-8.12
```

```
# ./configure && make && make install
```

- Cài đặt Libdnet

```
# tar -zxvf libdnet-1.11.tar.gz
# cd libdnet-1.11
# ./configure && make && make install
# Ln -s /usr/local/lib/libdnet.1.0.1 /usr/lib/libdnet

• Cài đặt Daq
# tar -zxvf daq-0.5.tar.gz
# cd daq-0.5
# ./configure && make && make install
# ldconfig

• Cài đặt Snort
# tar -xvf snort-2.9.0.5.tar.gz
# cd snort-2.9.0.5
# ./configure --prefix=/etc/KLTN2011/snort/ --enable-ipv6 --enable-gre\
--enable-mpls --enable-targetbased --enable-decoder-preprocessor-rules\
--enable-ppm --enable-perfprofiling --enable-zlib --enable-active-response\
--enable-normalizer --enable-reload --enable-react --enable-flexresp3\
--with-mysql --enable-dynamicplugin --enable-build-dynamic-examples

# make && make install
# mkdir /var/log/snort
# useradd snort
# chown snort:snort /var/log/snort

• Cài đặt snort rules
# tar -zxvf snortrules-snapshot-2904.tar.gz -C /etc/KLTN2011/snort
# mkdir /etc/KLTN2011/snort/lib/snort_dynamicrules
```

```

# cp
/etc/KLTN2011/snort/so_rules/precompiled/Ubuntu-10-4/i386/2.9.0.4/*\
/etc/KLTN2011/snort/lib/snort_dynamicrules

• Cài đặt barnyard2

# tar -zxvf barnyard2-1.9.tar.gz

# cd barnyard2-1.9

# ./configure --with-mysql

# make && make install

# cp etc/barnyard2.conf /etc/KLTN2011/snort/etc

# mkdir /var/log/barnyard2

# chmod 666 /var/log/barnyard2

# touch /var/log/snort/barnyard2.waldo

# chown snort.snort /var/log/snort/barnyard2.waldo

```

c) *Bước 3:*

- Cấu hình MySQL

```
# mysql -u root -p
```

Nhập password mysql của root đã khai báo ở bước 1

Khai báo:

```
create database snort;
```

```
grant all privileges on snort.* to snort@localhost identified by "password";
```

```
flush privileges;
```

Theo dõi kết quả như hình:

```

mysql> create database snort;
Query OK, 1 row affected (0.01 sec)

mysql> grant all privileges on snort.* to snort@localhost identified by 'vt081a'
;
Query OK, 0 rows affected (0.04 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

```

```
# mysql -u root -p -D snort < /home/snort-2.9.0.5/schemas/create_mysql
```

Với /home/snort-2.9.0.5/ là nơi chứa source cài. Nhập password MySQL của root

- Cấu hình trong file snort .conf

```
# vim /etc/KLTN2011/snort/etc/snort.conf
```

Khai báo đường dẫn tới các file rule và file preprocessor như hình dưới

```
# Path to your rules files (this can be a relative path)
# Note for Windows users: You are advised to make this an absolute path,
# such as: c:\snort\rules
var RULE_PATH /etc/KLTN2011/snort/rules
var SO_RULE_PATH /etc/KLTN2011/snort/so_rules
var PREPROC_RULE_PATH /etc/KLTN2011/snort/preproc_rules

#####
# Step #2: Configure the decoder. For more information, see README.decoder
83,0-1      13%
```

```
# path to dynamic preprocessor libraries
dynamicpreprocessor directory /etc/KLTN2011/snort/lib/snort_dynamicpreprocessor/
# path to base preprocessor engine
dynamicengine /etc/KLTN2011/snort/lib/snort_dynamicengine/libsf_engine.so
# path to dynamic rules libraries
dynamicdetection directory /etc/KLTN2011/snort/lib/snort_dynamicrules
-- INSERT --          176,1      332
```

Mở khoá cho module preprocessor sfportscan hoạt động

```
# Portscan detection. For more information, see README.sfportscan
preprocessor sfportscan: proto { all } memcap { 10000000 } sense_level { high }
#
-- INSERT --          312,1      65%
```

Mở khoá cho module decoder và preprocessor hoạt động

```
# Step #8: Customize your preprocessor and decoder alerts
# For more information, see README.decoder_preproc_rules
#####

# decoder and preprocessor event rules
include $PREPROC_RULE_PATH/preprocessor.rules
include $PREPROC_RULE_PATH/decoder.rules
include $PREPROC_RULE_PATH/sensitive-data.rules

#####
# Step #9: Customize your Shared Object Snort Rules
```

Khai báo output cho database

```
#output alert_unified2: filename snort.alert, limit 128, nostamp
output log_unified2: filename snort.log,limit 128

# syslog
output alert_syslog: LOG_AUTH LOG_ALERT

# pcap
output log_tcpdump: tcpdump.log

# database
output database: alert, mysql, user=snort password=vt081a dbname=snort host=localhost
output database: log, mysql, user=snort password=vt081a dbname=snort host=localhost
-- INSERT --                                         357,12          76%
```

```
# set the appropriate paths to the file(s) your Snort process is using.
#
config reference_file:      /etc/KLTT2011/snort/etc/reference.config
config classification_file: /etc/KLTT2011/snort/etc/classification.config
config gen_file:             /etc/KLTT2011/snort/etc/gen-msg.map
config sid_file:             /etc/KLTT2011/snort/etc/sid-msg.map

# define dedicated references similar to that of snort.          11,1          3%
```

• Cấu hình barnyard2

```
# vim /etc/KLTN2011/snort/etc/barnyard2.conf
```

Khai báo như hình dưới:

```
#config hostname:      thor
#config interface:    eth0
config hostname:      localhost
config interface:    eth0
-- INSERT --                                         41,0-1          13%
```

```
output database: log, mysql, user=snort password=vt081a dbname=snort host=localhost
```

• Tạo một liên kết mềm từ /etc/KLTN2011/snort/bin/snort đến /bin/

```
# ln -s /etc/KLTN2011/snort/bin/snort /bin
```

• Cấu hình interface

```
# vim /etc/network/interface
```

Ta khai báo địa chỉ IP sao cho máy Monitor có thể ping thấy máy Snort

```
# /etc/init.d/networking restart
```

d) Bước 4: Cài đặt và cấu hình công cụ hỗ trợ xuất alert trên web

- Cấu hình snort report

```
# mkdir /var/www/jpgraph
# tar -zxvf /home/quanthitrong/jpgraph-1.27.1.tar.gz
# cp -r /home/quanthitrong/jpgraph-1.27.1/src /var/www/jpgraph/
# tar -zxvf /home/quanthitrong/snortreport-1.3.1.tar.gz -C /var/www/
# vim /var/www/snortreport-1.3.1/srconf.php
```

Khai báo như hình dưới:

```
// Put your snort database login credentials in this section
$server = "localhost";
$user = "snort";
$pass = "vt081a";
$dbname = "snort";
```

```
define("JPGRAPH_PATH", "../jpgraph/src/");
// Path to external utilities
// Enter the correct path (including the binary)
// them installed
// You can also include switches for each binary
define("NMAP_PATH", "/usr/bin/nmap -v");
define("NBTSCAN_PATH", "/usr/bin/nbtscan");
```

- Cấu hình base

```
# tar -zxvf base-1.4.5.tar.gz -C /var/www/
# chmod 757 /var/www/base-1.4.5/
```

Ta edit file “base_stat_ipaddr.php” để người quản trị dễ dàng biết được vị trí của IP đã bắt được:

```
' <A HREF="http://www.dshield.org/ipinfo.php?ip=' . $ip . '&Submit=Submit" target="_NEW">DShield.org IP Info</A> ! .
' <A HREF="http://www.trustedsource.org/query.php?q=' . $ip . '" target="_NEW">TrustedSource.org IP Info</A> ! .
' <A HREF="http://isc.sans.org/ipinfo.html?ip=' . $ip . '" target="_NEW">ISC Source/Subnet Report</A><BR> </FONT> .
' <A HREF="http://whatismyipaddress.com/ip/' . $ip . '#Geolocation-Map" target="_NEW">GOOGLE_IP_MAP</A><BR> </FONT> ';
```

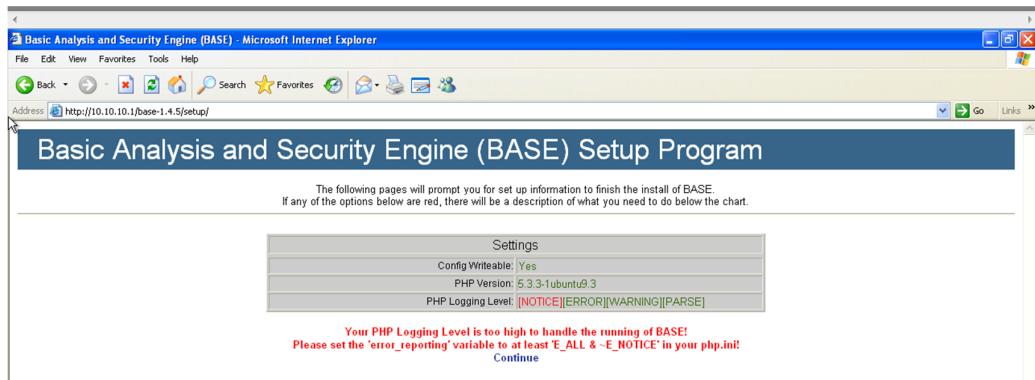
-- INSERT -- 362,7 76%

Trên máy Monitor:

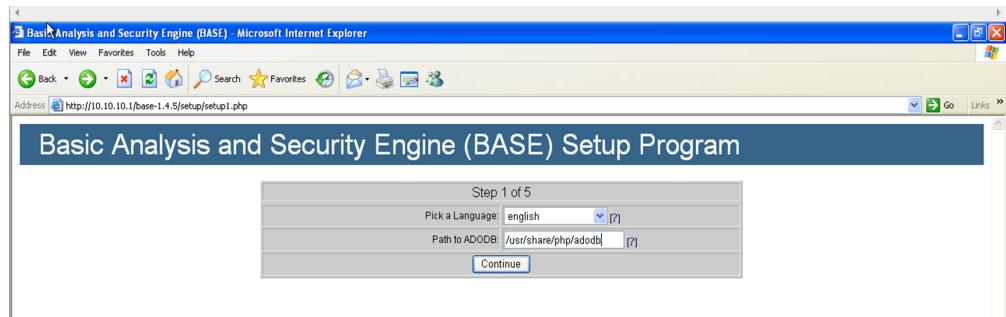
Trên trình duyệt web ta gõ như sau:

http://ip_snort/base-1.4.5/setup/

Với ip_snort là địa chỉ IP của máy Snort



Nhấn “Continue”



Khai báo “/usr/share/php/adodb” vào Path to ADODB

Nhấn “Continue”

Step 2 of 5

Pick a Database type: MySQL [?]

Database Name:	snort
Database Host:	localhost
Database Port:	Leave blank for default!
Database User Name:	snort
Database Password:	*****

Use Archive Database [?]

Archive Database Name:	snort
Archive Database Host:	localhost
Archive Database Port:	Leave blank for default!
Archive Database User Name:	snort
Archive Database Password:	*****

Continue

Khai báo thông tin

Step 3 of 5

Use Authentication System [?]

Admin User Name:	snort
Password:	*****
Full Name:	snort

Continue

Basic Analysis and Security Engine (BASE) - Microsoft Internet Explorer

Basic Analysis and Security Engine (BASE) Setup Program

Operation	Description	Status
BASE tables	Adds tables to extend the Snort DB to support the BASE functionality	Create BASE AG
• snort		

Nhấn “Create BASE AG”

The underlying Alert DB is configured for usage with BASE.

Additional DB permissions
In order to support Alert purging (the selective ability to permanently delete alerts from the database) and DNS/whois lookup caching, the DB user "snort" must have the DELETE and UPDATE privilege on the database "snort@localhost".

Now continue to [step 5...](#)

Ta thấy BASE đã được cấu hình thành công

Đăng nhập lại : http://ip_snort/base-1.4.5

Today's alerts: unique listing Source IP Destination IP

Last 24 Hours alerts: unique listing Source IP Destination IP

Last 72 Hours alerts: unique listing Source IP Destination IP

Most recent 15 Alerts: any protocol TCP UDP ICMP

Last Source Ports: any protocol TCP UDP

Last Destination Ports: any protocol TCP UDP

Most Frequent Source Ports: any protocol TCP UDP

Most Frequent Destination Ports: any protocol TCP UDP

Most frequent 15 Addresses: Source Destination

Most recent 15 Unique Alerts

Most frequent 5 Unique Alerts

Sensors/Total: 0 / 2

Unique Alerts: 0

Categories: 0

Total Number of Alerts: 0

Src IP addrs: 0

Dest IP addrs: 0

Unique IP links: 0

Source Ports: 0

Dest Ports: 0

o TCP (0) UDP (0)

o TCP (0) UDP (0)

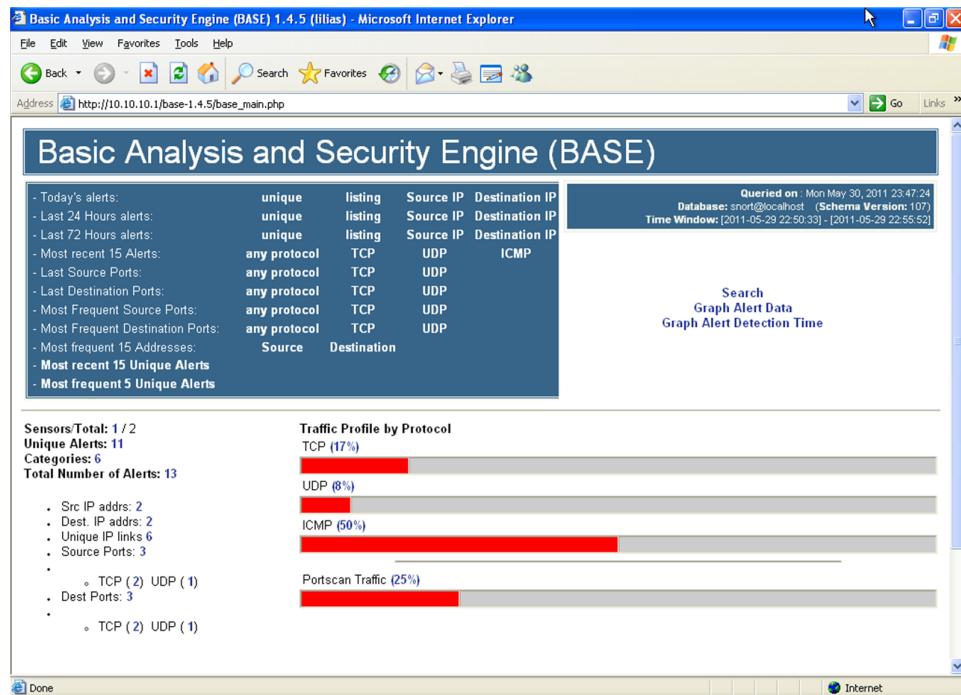
o TCP (0) UDP (0)

Search Graph Alert Data Graph Alert Detection Time

Alert Group Maintenance | Cache & Status | Administration

Đăng nhập thành công

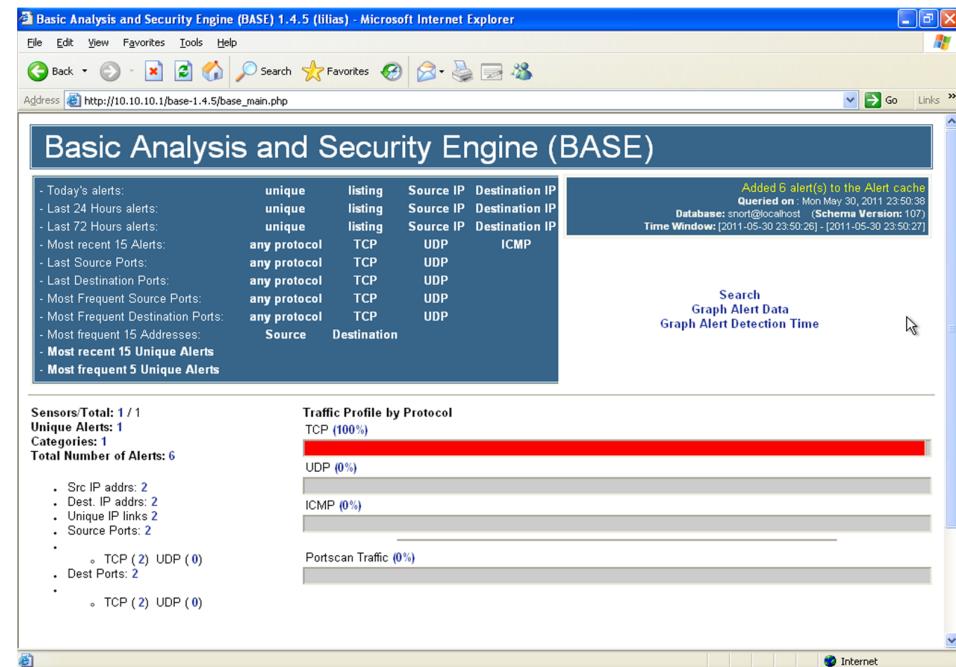
Từ máy client ta thử nmap tới máy snort



Viết một rule cảnh báo đơn giản

aler tcp any any <> any 23 (msg: “telnet is not allowed by snort”; sid: 1000511;)

telnet tới máy snort và xem kết quả



Basic Analysis and Security Engine (BASE) : Query Results - Microsoft Internet Explorer

Queried on : Mon May 30, 2011 23:51:28

Meta Criteria: any
IP Criteria: any
Layer 4 Criteria: none
Payload Criteria: any

Summary Statistics

- Sensors
- Unique Alerts
- (classifications)
- Unique addresses: Source | Destination
- Unique IP links
- Source Port: TCP | UDP
- Destination Port: TCP | UDP
- Time profile of alerts

Displaying alerts 1-6 of 6 total

ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(22-3)	[snort] telnet is not allowed by snort	2011-05-30 23:50:27	10.10.10.2.1070	10.10.10.1.23	TCP
#1-(22-4)	[snort] telnet is not allowed by snort	2011-05-30 23:50:27	10.10.10.1.23	10.10.10.2.1070	TCP
#2-(22-5)	[snort] telnet is not allowed by snort	2011-05-30 23:50:27	10.10.10.2.1070	10.10.10.1.23	TCP
#3-(22-6)	[snort] telnet is not allowed by snort	2011-05-30 23:50:27	10.10.10.1.23	10.10.10.2.1070	TCP
#4-(22-1)	[snort] telnet is not allowed by snort	2011-05-30 23:50:26	10.10.10.2.1070	10.10.10.1.23	TCP
#5-(22-2)	[snort] telnet is not allowed by snort	2011-05-30 23:50:26	10.10.10.1.23	10.10.10.2.1070	TCP

ACTION: { action } Selected ALL on Screen Entire Query

Alert Group Maintenance | Cache & Status | Administration

Xem kết quả trên snortreport

SNORT Report - Microsoft Internet Explorer

Timeframe: 2011-05-29 23:58:16 to 2011-05-30 23:58:16
Current Time: 2011-05-30 23:58:16
Unique Signatures: 4
Number of Alerts: 8

Earliest Alert: 2011-05-30 23:50:26
Latest Alert: 2011-05-30 23:58:16

Timeframe: Day: GO

Types of Traffic

Detail by Signatures

Num	Prio	Signature	# Alerts	# Sources	# Dest.	Detail
1		telnet is not allowed by snort [sid 1000511]	6	2	2	Summary
2	2	PSNG_TCP_PORTSWEET	1	1	1	Summary
3	3	(portscan) Open Port	1	1	1	Summary

Page begun: May 30, 2011, 23:58:16
Page finished: May 30, 2011, 23:58:16

Snort Report Version 1.3.1
Copyright 2000-2005, Symmetric Technologies, LLC.

all alerts with 222.255.236.114/32 as: [Source](#) | [Destination](#) | [Source/Destination](#)
 show: [Unique Alerts](#) | [Portscan Events](#)
 Registry lookup (whois) in: [ARIN](#) | [RIPE](#) | [APNIC](#) | [LACNIC](#)
 external: [DNS](#) | [whois](#) | [Extended whois](#) | [DShield.org IP Info](#) | [TrustedSource.org IP Info](#) | [ISC Source/Subnet Report](#)

[GOOGLE_IP_MAP](#) ←

222.255.236.114

FQDN: (no DNS resolution attempted) ([local whois](#))

Num of Sensors	Occurrences as Src.	Occurrences as Dest.	First Occurrence	Last Occurrence
1	3	3	2011-07-13 23:09:40	2011-07-13 23:09:52

Và sau đó:

Geolocation Information

Country: Vietnam 

State/Region: Ninh Bình

City: Nghiêm

Latitude: 9.8478

Longitude: 106.3456

Geolocation Map



e) *Bước 5: Tự động cập nhật Snort rule với Oinkmaster*

Oinkmaster là 1 Perl script giúp cập nhật và quản lý các rule của Snort, bao gồm các rule chính thức (official VRT rule), các rule được phát triển bởi cộng đồng sử dụng Snort (community rule), các rule được phát triển bởi nhà cung cấp thứ 3....

1. Cài đặt Oinkmaster

Download Oinkmaster tại : <http://oinkmaster.sourceforge.net/download.shtml>

Thực hiện các lệnh:

- tar zxvf oinkmaster-2.0.tar.gz
- cd oinkmaster-2.0
- cp oinkmaster.pl /usr/local/bin
- cp oinkmaster.conf /usr/local/etc
- cp oinkmaster.1 /usr/local/man/man1

2. Đăng ký account tại Snort.org sau đó vào Get Rules, chọn Get Oink Code
<https://www.snort.org/account/oinkcode>

Oinkcode

17bebd0259af139bc85d3361aa9ea3499aa6ba71

3. Sửa file Oinkmaster.conf

- *gedit oinkmaster.conf*
- tìm “url = ...” và sửa thành :

```

49 # Snort site in your registered user profile.
50
51 # Example for Snort 2.4
52 # url = http://www.snort.org/pub-bin/oinkmaster.cgi/<oinkcode>/snortrules-sn
53 url = http://www.snort.org/pub-bin/oinkmaster.cgi/17bebd0259af139bc85d3361aa
54 9ea3499aa6ba71/snortrules-snapshot-2905.tar.gz
55 # Example for Snort-current ("current" means cvs snapshots).
56 # url = http://www.snort.org/pub-bin/oinkmaster.cgi/<oinkcode>/snortrules-sn
      apshot-CURRENT.tar.gz

```

(với oinkcode là 17bebd0259af139bc85d3361aa9ea3499aa6ba71 đã lấy được ở trên , và Snort là ver. 2.9.0.5)

4. Khởi động Oinkmaster
oinkmaster.pl -o /usr/local/snort/rules

```
root@ubuntu:~# oinkmaster.pl -o /usr/local/snort/rules/
Loading /usr/local/etc/oinkmaster.conf
Downloading file from http://www.snort.org/pub-bin/oinkmaster.cgi/*oinkcode*/snortrules-snapshot-2905.tar.gz... done.
Archive successfully downloaded, unpacking... done.
Setting up rules structures...
WARNING: duplicate SID in your local rules, SID 3017 exists multiple times, you
may need to fix this manually!

```

Kiểm tra quá trình update rule sau khi hoàn tất

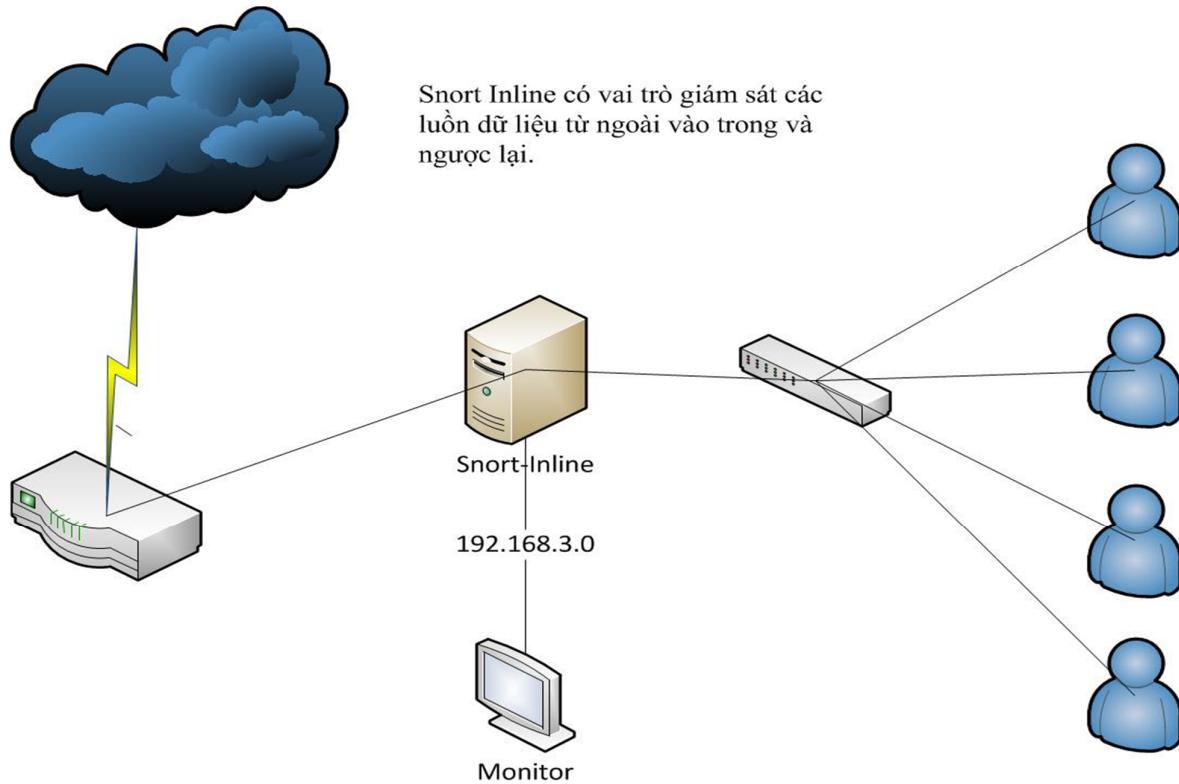
```
root@ubuntu:~# oinkmaster.pl -o /usr/local/snort/rules/
Loading /usr/local/etc/oinkmaster.conf
Downloading file from http://www.snort.org/pub-bin/oinkmaster.cgi/*oinkcode*/snortrules-snapshot-2905.tar.gz... done.
Archive successfully downloaded, unpacking... done.
Setting up rules structures...
WARNING: duplicate SID in your local rules, SID 3017 exists multiple times, you
may need to fix this manually!
[!] Non-rule line modifications: [*]
  None.

[!] Added files: [*]
  None.

root@ubuntu:~#
```

5.2. Cấu hình Snort Inline IPS:

5.2.1. Mô hình



5.2.2. Các bước cài đặt

Bước 1: Cài đặt

Ta cần cài thêm các gói như sau:

```
apt-get install iptables-dev
apt-get install libiptables-ipv4-ipqueue-perl
apt-get install bridge-utils
apt-get install libnetfilter-queue-dev
```

Ta tải gói iptables

<http://www.iptables.org/projects/iptables/files/iptables-1.4.10.tar.bz2>

Cài iptables

```
# tar -xvf iptables-1.4.10.tar.bz2
# cd iptables-1.4.10
```

```
# ./configure --enable-libipq && make && make install
```

Bước 2: Cấu hình

Sau khi cài xong iptables ta làm tiếp như sau:

Vào file /etc/rc.local tạo cổng br0 để bridge giữa hai cổng eth1 và eth2

```
ifconfig eth1 up
ifconfig eth2 up
ifconfig eth1 0.0.0.0 -arp promisc
ifconfig eth2 0.0.0.0 -arp promisc
brctl addbr br0
brctl addif br0 eth1
brctl addif br0 eth2
ifconfig br0 0.0.0.0 up -arp
modprobe ip_queue
iptables -A FORWARD -i br0 -p ALL -j QUEUE
exit 0
"/etc/rc.local" 24L, 558C
```

```
# vim /etc/network/interface
```

```
auto eth0
iface eth0 inet static
address 192.168.3.1
netmask 255.255.255.0
network 192.168.3.0
broadcast 192.168.3.255

auto eth1
iface eth1 inet manual

auto eth2
iface eth2 inet manual
```

```
# site specific rules
include $RULE_PATH/local.rules
include $RULE_PATH/drop.rules
include $RULE_PATH/attack-responses.rules
-- INSERT --
```

Ta tạo drop.rules trong /etc/KLTN2011/snort/rule/

```
drop TCP any any -> any 80 (content:"thanhnien";msg:"đang truy cập trang thanhnien";sid:20000000;)
```

Ta chạy lệnh snort -Q --daq ipq -c /etc/KLTN2011/snort/etc/snort.conf

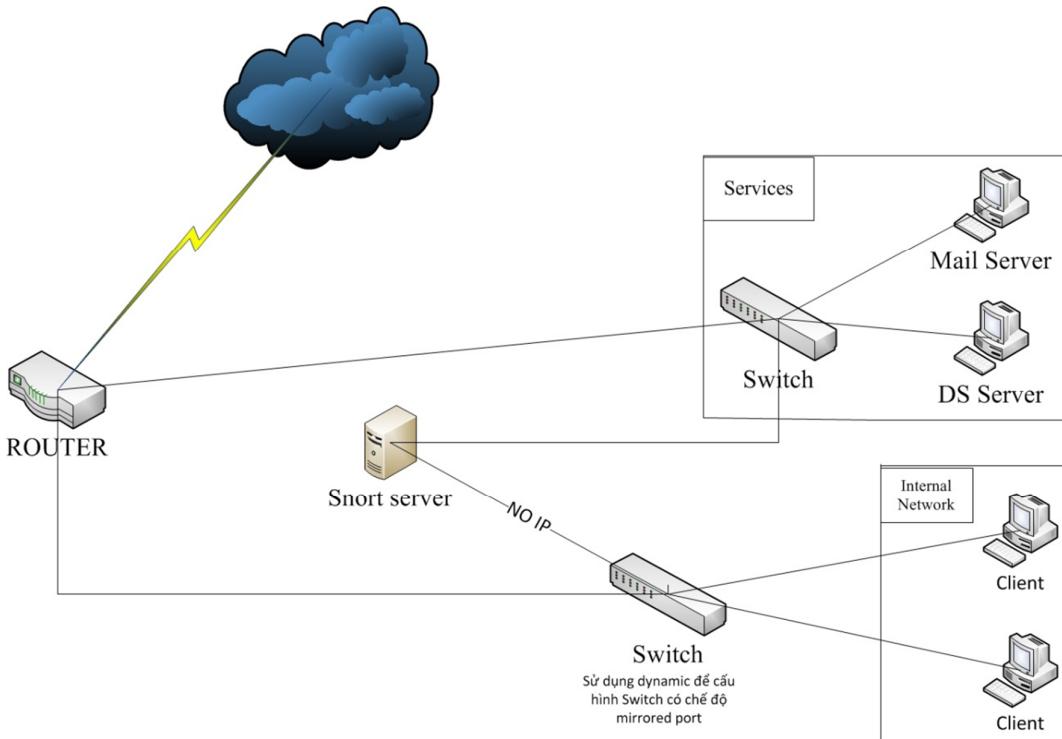
Khi client truy cập trang web “thanhnien” thì client sẽ không truy cập được và snort-inline log lại trong MySQL

<input type="checkbox"/>	ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
<input type="checkbox"/>	#0-(1- [snort] dang truy cap trang 51) thanhnien		2011-05-30 02:07:25	10.2.0.239:1062	222.255.236.114:80	TCP
<input type="checkbox"/>	#1-(1- [snort] dang truy cap trang 52) thanhnien		2011-05-30 02:07:25	10.2.0.239:1062	222.255.236.114:80	TCP
<input type="checkbox"/>	#2-(1- [snort] dang truy cap trang 49) thanhnien		2011-05-30 02:07:17	10.2.0.239:1061	222.255.236.114:80	TCP
<input type="checkbox"/>	#3-(1- [snort] dang truy cap trang 50) thanhnien		2011-05-30 02:07:17	10.2.0.239:1061	222.255.236.114:80	TCP

5.3. Cấu hình Mail Alert

Cấu hình để snort có thể gửi cảnh báo qua email

5.3.1. Mô hình



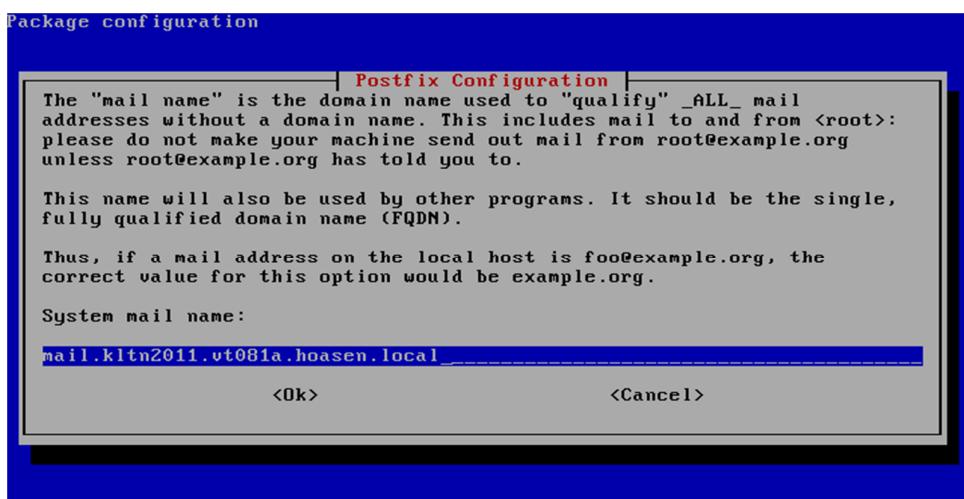
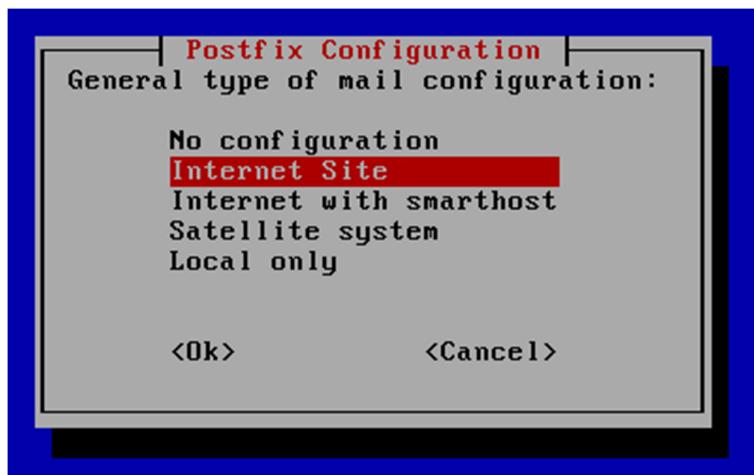
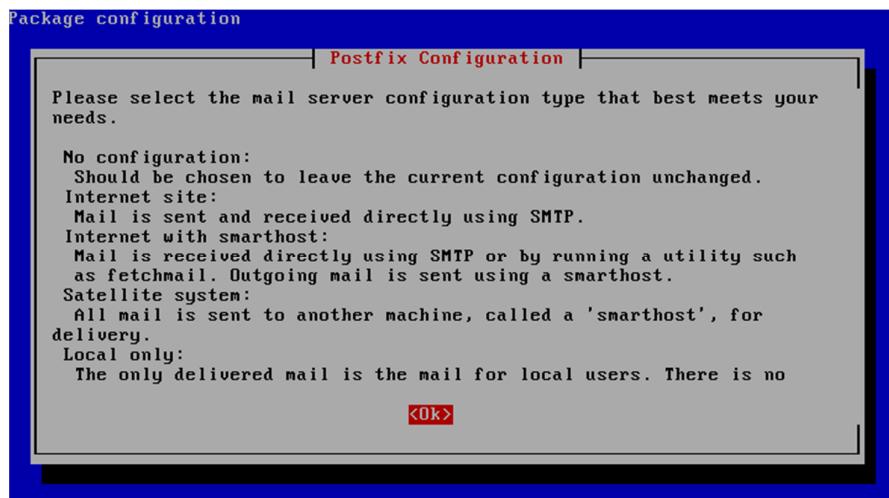
5.3.2. Các bước cài đặt

Các bước chuẩn bị:

- Một DNS server để có thể phân giải tên miền
- Một Mail server
- Snort Server có 2 interface:
 - eth0: dùng để thêm dõi các gói tin
 - eth1: dùng để gửi mail.

Bước 1: Các thêm phần mềm :

```
# apt-get install mailutils
```



Khai báo tên miền của DNS dùng để cấu hình mail

Bước 2: Cấu hình interface và file resolv.conf

```
auto eth0
iface eth0 inet manual

auto eth1
iface eth1 inet static
address 192.168.10.14
netmask 255.255.255.0
network 192.168.10.0
broadcast 192.168.10.255
```

```
nameserver 192.168.10.200
```

Bước 3: Thêm dòng một dòng output (gắn dòng output database) ở file snort.conf :

```
# output alert_full: /var/log/snort/alert
```

Bước 4: cấu hình mail postfix

```
# vim /etc/postfix/main.cf
```

```
myhostname = KLTN2011
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
mydestination = mail.kltn2011.vt081a.hoasen.local, KLTN2011, localhost.localdomain, localhost
relayhost =
mynetworks = 192.168.10.0/24, 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
```

Bước 5: khai báo file “.swatchrc” ở /root/

```
watchfor /thanhnien/
echo red
mail snort@kltn2011.vt081a.hoasen.local,subject=Dang Truy cap web
```

Kiểm tra sự hoạt động của mail

Ta tạo file test_simple_snort_rule.rules đơn giản ở thư mục chứa rule của snort (/etc/KLTN2011/snort/rule)

```
alert tcp any any -> any 80 (msg:"truy cap trang thanh nien";content:"thanhnien.com.vn"; sid: 30000000;)
```

Và file snort.conf ta thêm vào như sau:

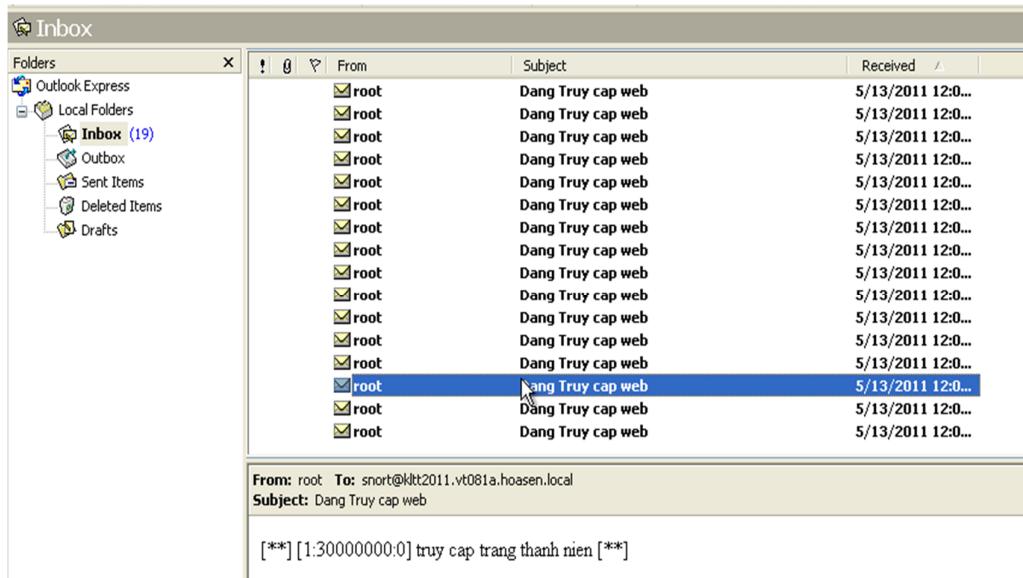
```
include $RULE_PATH/test_simple_snort_rule.rules
```

Và chạy câu lệnh swatch -t /var/log/snort/alert

Khi user trong nội bộ truy cập trang web “thanhnien.com.vn”:

```
root@KLTT2011:~# swatch -t /var/log/snort/alert
*** swatch version 3.2.3 (pid:3327) started at Fri May 13 12:03:51 ICT 2011
[**] [1:30000000:0] truy cap trang thanh nien [**]
```

Check mail trên outlook



5.4. NGĂN CHẶN CUỘC TẤN CÔNG DỰA TRÊN SỐ LƯỢNG GÓI TIN TỪ MỘT IP

Khi các attacker không tìm được lỗ hổng bảo mật của server thì các attacker sẽ đi đến bước cuối là phá huỷ server bằng các tool DoS, nên chúng tôi đã viết đoạn mã để ngăn chặn cuộc tấn công này.

Trong file snort.conf ta cấu hình như sau :

```
ruletype auto
{
  type alert
  output alert_fast: /var/log/snort/fast
}

include $RULE_PATH/auto.rules
include $RULE_PATH/autodrop.rules
```

Trong file auto.rules ta khai báo như sau:

```
auto TCP !$DMZ any -> any any (msg:"log";sid:200000001;)
auto UDP !$DMZ any -> any any (msg:"log";sid:200000002;)
auto IP !$DMZ any -> any any (msg:"log";sid:200000003;)
auto ICMP !$DMZ any -> any any (msg:"log";sid:200000004;)
```

Trong file autorun ta khai báo một đoạn mã script

Cấu hình crontab như sau:

```
* * * * * sh /KLTN2011/snort/etc/autorun
"/tmp/crontab.bFc3up/crontab" 23L, 929C
```

Ta khởi động 2 dịch vụ :

- /etc/init.d/cron restart
- snort -Q -daq nfq -c /KLTN2011/snort/etc/snort.conf &

Trên máy attacker có địa chỉ như sau:

```
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:
  Connection-specific DNS Suffix . :
  IP Address . . . . . : 192.168.10.6
  Subnet Mask . . . . . : 255.255.255.0
  Default Gateway . . . . . : 192.168.10.100

C:\Documents and Settings\Administrator>
```

Máy attacker truy cập được web và bắt đầu tấn công Web server bằng tool Donut HTTP Flooder; attacker tấn công thành công, tuy nhiên sau thời gian 1 đến 2 phút thì cuộc tấn công thất bại vì ip của attacker đã đưa vào file black_list:

```
# 192.168.10.6
#
#
#
"~/KLTN2011/snort/etc/black_list" 2L, 150 1,1      All
```

File autodrop.rules :

```
# drop TCP 192.168.10.6 any -> any any (msg:"Auto drop IP (TCP)";sid:190000001;)
drop IP 192.168.10.6 any -> any any (msg:"Auto drop IP (IP)";sid:190000002;)
drop ICMP 192.168.10.6 any -> any any (msg:"Auto drop IP (ICMP)";sid:190000003;)
drop UDP 192.168.10.6 any -> any any (msg:"Auto drop IP (UDP)";sid:190000004;)
#
#
"~/KLTN2011/snort/rules/autodrop.rules" 5L, 318C 1,1      All
```

6. ĐÁNH GIÁ KẾT QUẢ VÀ HƯỚNG PHÁT TRIỀN

Trong quá trình làm việc chúng tôi có một số thuận lợi như : chúng tôi đã vận dụng các kiến thức đã học, có sự hỗ trợ của các anh trong phòng kĩ thuật, tài liệu về hệ thống Snort-IDS rất phong phú trên internet,... Và đạt được một số kết quả như: xây dựng thành công hệ thống giám sát snort; hỗ trợ khả năng ngăn chặn của snort_inline thực hiện được các nhiệm vụ cơ bản của một hệ thống cảnh báo và ngăn chặn xâm nhập; hỗ trợ chức năng giám sát và thông kê cảnh báo qua web; gửi cảnh báo qua mail tới người quản trị; xây dựng một script hoạt động với chức năng tối ưu hoá khả năng làm việc của hệ thống Snort;... Bên cạnh đó chúng tôi đã gặp một số khó khăn như: còn hạn chế về khối lượng kiến thức và thời gian để phát triển chuyên sâu về hệ thống; thiếu các thiết bị cần thiết khi xây dựng hệ thống trên mô hình máy thật (Switch có chức năng mirror port); hệ thống cảnh báo qua mail hoạt động còn chưa hoàn chỉnh; đoạn mã script còn nhiều thiếu sót, làm cho hệ thống hoạt động không như ý muốn.

Chúng tôi sẽ tiếp tục nghiên cứu trong việc sử dụng các module xử lý và phát triển các rule để tăng cường khả năng ngăn chặn các cuộc tấn công, phát triển các hệ thống gửi cảnh báo tới người quản trị như SNMP, mail, sms..., phát triển snort để có thể hỗ trợ sử dụng các firewall phần cứng trong việc ngăn chặn xâm nhập như hệ thống snortsam, hoàn thiện và bổ sung script để hệ thống hoạt động hiệu quả hơn.

7. KẾT LUẬN

Ngày nay, sự phát triển mạnh mẽ của công nghệ thông tin cùng với mức độ gia tăng không ngừng của các ứng dụng, việc đảm bảo cho hệ thống mạng thông tin của doanh nghiệp vận hành một cách ổn định, hiệu quả và an toàn là một việc làm tương đối khó khăn và cần có đầu tư về vật chất, con người và thời gian một cách hợp lý. Khi mà việc xuất hiện ngày càng nhiều các cuộc tấn công lên các hệ thống mạng không chỉ với mục đích phá hoại mà còn tiềm ẩn nhiều nguy cơ chính trị, kinh tế thì vấn đề bảo mật trong hệ thống mạng lại càng được quan tâm chú ý. Các giải pháp kết nối WAN như leasedline không có sự mềm dẻo linh hoạt về mặt kết nối, mở rộng mạng cũng như an toàn thông tin, các giải pháp bảo mật firewall cũng chỉ đảm bảo chống tấn công từ phía bên ngoài vào trong mạng tại điểm cửa ngõ mạng, trong khi thông tin trên đường truyền thì hoàn toàn ở dạng plaintext, do đó cần có thêm các cơ chế giám sát hợp lý để tăng cường độ bảo mật cho hệ thống.

Với khả năng phân tích mạnh mẽ các luồng traffic, hỗ trợ cơ chế cảnh báo ngăn chặn cũng như sự linh hoạt trong việc cấu hình và các xây dựng các rules trên nền mã nguồn mở, Snort đã trở thành một công cụ không thể thiếu đối với người quản trị mạng trong việc bảo vệ các hệ thống thông tin quan trọng hiện chống lại sự tấn công ồ ạt bởi những kẻ phá hoại đang ngày càng gia tăng về số lượng, loại hình, kỹ thuật như hiện nay.

Trong suốt quá trình làm việc, chúng tôi đã đạt một số thành quả như: hiểu rõ những điều cơ bản về hệ thống IDS-IPS, có cái nhìn cơ bản về mô hình thật tế, và bổ sung kiến thức cho chúng tôi về bảo mật.

Tuy mức độ nghiên cứu của đồ án còn hạn chế cũng như những kiến thức còn chưa đầy đủ về hệ thống mạng máy tính song chúng tôi hy vọng đồ án này sẽ mang lại cho các bạn một cái nhìn khái quát hơn về việc xây dựng các hệ thống cảnh báo xâm nhập, các phương pháp phòng thủ hiệu quả cho một hệ thống thông tin.

8. PHỤ LỤC

Đoạn mã để tăng thêm sự linh hoạt cho máy Snort-inline

```
#!/bin/sh

rm -rf /var/log/temp/now
mkdir /var/log/temp/now
dir=/var/log/temp/now
temp=$dir/temp
temp1=$dir/temp1
drop=$dir/drop
drop1=$dir/drop1
ip=$dir/ip
kl=/KLTN2011/snort
local_ip=$kl/etc/local_ip
black_list=$kl/etc/black_list
id=$kl/etc/sid
c=0
m=0
n=0
touch $temp
cp /var/log/snort/fast $temp
sed 's/ /:/g' $temp >$temp1
cut -d } -f2 $temp1> $temp
cut -d : -f2 $temp> $temp1
awk '$1==$(cat temp1) {count[$1]++}END{for(j in count) print j, count[j]}' $temp1 >
$temp
sed 's/ /:/g' $temp > $ip
for a in $(cat $dir/ip)
do
    echo $a > $temp
    cut -d : -f2 $temp > $temp1
```

```

for x in $(cat $dir/temp1)
do
    if test $x -gt 15000
    then
        cut -d : -f1 $temp >> $drop
        m=1
    fi
done
done
#####
#So sach Ip voi IP vung DMZ
if test $m -eq 1
then
    cat /KLTN2011/snort/etc/local_ip >> $drop
    cat /KLTN2011/snort/etc/local_ip >> $drop
    awk '$2=$(cat drop) {count[$2]++}END{for(j in count) print j, count[j]}' $drop >
$drop1
    cp $drop1 $drop
    sed 's/ /:/g' $drop >$drop1
    rm -rf $drop
    rm -rf $temp
    rm -rf $temp1

for y in $(cat $dir/drop1)
do
    echo $y > $temp
    cut -d : -f2 $temp > $temp1
    for z in $(cat $dir/temp1)
    do
        if test $z -lt 2
        then

```

```
        cut -d : -f1 $temp >> $drop
        n=1
    fi
done
done
cat $dir/drop >> /KLTN2011/snort/etc/bl_day
fi
#####
##Black list
if test $n -eq 1
then
cat $kl/etc/black_list >> $drop
cat $kl/etc/black_list >> $drop
awk '$3=$(cat drop) {count[$3]++}END{for(j in count) print j, count[j]}' $drop >
$drop1
cp $drop1 $drop
sed 's/ /:/g' $drop > $drop1
rm -rf $drop
touch $drop
for y in $(cat $dir/drop1)
do
    echo $y > $temp
    cut -d : -f2 $temp > $temp1
    for z in $(cat $dir/temp1)
    do
        if test $z -lt 2
        then
            c=1
            cut -d : -f1 $temp >> $drop
        fi
    done
```

```

done
fi
drop_rules=/KLTN2011/snort/rules/autodrop.rules
if test $c -eq 1
then
sid=$( cat $kl/etc/sid)
echo "1" > /KLTN2011/snort/etc/signal
cat $dir/drop >> $kl/etc/black_list
for k in $(cat $dir/drop)
do
sid=`expr $sid + 1`
echo 'drop TCP '$k' any -> any any (msg:"Auto drop (TCP)";threshold:
type limit, track by_src, count 1, seconds 60; sid:'$sid';)' >> $drop_rules
sid=`expr $sid + 1`
echo 'drop IP '$k' any -> any any (msg:"Auto drop (IP)";threshold: type
limit,track by_src, count 1, seconds 60; sid:'$sid';)' >> $drop_rules
sid=`expr $sid + 1`
echo 'drop ICMP '$k' any -> any any (msg:"Auto drop (ICMP)";threshold:
type limit, track by_src, count 1,seconds 60; sid:'$sid';)' >> $drop_rules
sid=`expr $sid + 1`
echo 'drop UDP '$k' any -> any any (msg:"Auto drop
(UDP)";threshold:type limit, track by_src, count 1, seconds 60; sid:'$sid';)'
>> $drop_rules
done
mail -s "Auto Drop IP" snort@kltn2011.vt081a.hoasen.local< $dir/drop
pkill snort
rm -rf /var/log/snort/fast
snort -Q --daq nfq -c /KLTN2011/snort/etc/snort.conf &
echo $sid > $kl/etc/sid
fi

```

9. TÀI LIỆU THAM KHẢO

- [1] David Gullett, “Snort 2.9.0.5 and Snort Report 1.3.1 on Ubuntu 10.04 LTS Installation Guide”, Symmetrix Technologies, April 9 2011
- [2] Rafeeq Rehman, “Intrusion Detection with Snort”, Prentice Hall, 2003
- [3] The Snort Team, “SNORT Users Manual 2.9.0”, Sourcefire, Inc, March 25, 2011
- [4] Jay Beale, James C. Foster, Jeffrey Posluns, Brian Caswell, “Snort 2.0 Intrusion Detection”, Syngress Publishing, Inc, 800 Hingham Street, Rockland, MA 02370, 2003
- [5] A. Baker, J. Beale, B. Caswell, M. Poor, “Snort 2.1 Intrusion Detection Second Edition”, Syngress Publishing, Inc, 800 Hingham Street Rockland, MA 02370, May 2004
- [6] <http://vtc.vn/congnghe/560-277747/thu-thuat-meo-vat/soi-dong-the-gioi-ngam-cong-nghe.htm>
- [7] <http://www.go.vn/news/689-9208/bao-mat-amp-antivirus/doanh-nghiep-my-mat-hang-ty-do-la-va-toi-pham-mang.htm>
- [8] <http://www.tinmoi.vn/Ngan-hang-chong-do-hon-100000-cuoc-tan-cong-moi-ngay-0440332.html>
- [9] <http://vnexpress.net/gl/vi-tinh/hacker-virus/2009/08/3ba12157/>
- [10] <http://nhipsongso.tuoitre.vn/nhip-song-so/431740/Hang-tram-nghin-trang-web-bi-tan-cong.html>

NHẬN XÉT CỦA GIÁO VIÊN