HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÁO CÁO BÀI TẬP LỚN

Đề tài: Tìm hiểu Zeek Scripting để phát hiện sự kiện bất thường và độc hại

Giảng viên: Ninh Thị Thu Trang

Nhóm lớp: 01

Nhóm BTL: 06

Thành viên: Hoàng Trung Kiên-B20DCAT098

Ninh Chí Hướng-B20DCAT094

Nguyễn Mạnh Hưng-B20DCAT090

Nguyễn Văn Khang-B20DCAT102

Hà Nội – 2024

Mục lục

Lời mở đầu

Ngày nay, bạn cần kiểm soát hoàn toàn các sự cố mạng tiềm ẩn, đặc biệt là khi nói đến an ninh. Ngoài ra, có một cái nhìn tổng quan toàn cầu về chúng: nguyên nhân, tác động đến các nhiệm vụ hàng ngày và các giải pháp có thể được áp dụng. Thời gian đang chạy, buộc kết nối phải đáng tin cậy và cung cấp bảo vệ chống lại nhiều mối đe dọa.

Mạng bảo mật và quản lý mạng đang phát triển một cách nhanh chóng nhờ các công cụ giúp mọi thứ trở nên dễ dàng và thiết thực hơn nhiều. Đã qua rồi cái thời mà nhiều giải pháp, rất đắt và khó sử dụng, không đưa ra câu trả lời mong muốn. Các cuộc tấn công mạng đang ngày càng ít đình chiến hơn và các mạng phải có một lá chắn bảo vệ thực sự.

Zeek được trình bày như một công cụ để hỗ trợ quản lý ứng phó sự cố an ninh. Nó hoạt động bằng cách bổ sung dựa trên chữ ký các công cụ để tìm và theo dõi các sự kiện mạng phức tạp. Nó được đặc trưng bằng cách cung cấp phản hồi nhanh, ngoài việc sử dụng nhiều luồng và giao thức. Nó không chỉ giúp xác định các sự kiện bảo mật, mà còn nhằm mục đích tạo điều kiện khắc phục sự cố.

I. Tìm hiểu về phát hiện xâm nhập dựa trên bất thường.

a, Phát hiện xâm nhập dựa trên bất thường là gì?

Phát hiện xâm nhập dựa trên bất thường là một phương pháp phát hiện các hoạt động độc hại trong mạng hoặc hệ thống bằng cách phân tích các hành vi và tìm kiếm những điểm bất thường so với mô hình hành vi bình thường đã được thiết lập.

Phương pháp này dựa vào quan sát sự cố mạng và nhận biết lưu lượng bất thường thông qua các chẩn đoán và thống kê và có khả năng nhận ra các mẫu tấn công khác biệt với hành vi mạng thông thường.

b, Ưu, nhược điểm của phát hiện xâm nhập dựa trên bất thường.

- Ưu điểm:
    - Có tiềm năng phát hiện các loại xâm nhập mới mà không yêu cầu biết trước thông tin về chúng.
- Nhược điểm:
    - Tỷ lệ cảnh báo sai tương đối cao so với phương pháp dựa trên chữ ký.
    - Tiêu tốn nhiều tài nguyên hệ thống cho việc xây dựng hồ sơ đối tượng và phân tích hành vi hiện tại.

c, Các phương pháp xử lý, phân tích dữ liệu và mô hình hoá trong phát hiện xâm nhập dựa trên bất thường.

- Thống kê (statistics).
- Học máy (machine learning): HMM, máy trạng thái (state-based).
- Khai phá dữ liệu (data mining).
- Mạng nơ ron (neural networks).

II. Zeek là gì?

1. Tổng quan về zeek.

Zeek (trước đây được gọi là Bro) là một công cụ phân tích lưu lượng mạng mã nguồn mở thụ được phát triển bởi Lawrence Berkeley Labs. Nhiều nhà khai thác sử dụng Zeek như một trình giám sát an ninh mạng (NSM) để hỗ trợ điều tra hoạt động đáng ngờ hoặc độc hại. Zeek cũng hỗ trợ một loạt các nhiệm vụ phân tích lưu lượng ngoài miền bảo mật, bao gồm đo lường hiệu suất và khắc phục sự cố.

Lợi ích đầu tiên mà người dùng mới nhận được từ Zeek là bộ nhật ký mở rộng mô tả hoạt động mạng. Những nhật ký này không chỉ bao gồm bản ghi toàn diện về mọi kết nối nhìn thấy trên đường dây mà còn bao gồm cả bản ghi của lớp ứng dụng. Chúng bao gồm tất cả các phiên HTTP có URI được yêu cầu, tiêu đề chính, loại MIME và phản hồi của máy chủ; Yêu cầu DNS kèm theo câu trả lời; Chứng chỉ SSL; nội dung chính của phiên SMTP; và nhiều hơn nữa. Theo mặc định, Zeek ghi tất cả thông tin này vào các tệp nhật ký JSON hoặc được phân tách bằng tab có cấu trúc tốt, phù hợp để xử lý hậu kỳ bằng phần mềm bên ngoài. Người dùng cũng có thể chọn để cơ sở dữ liệu bên ngoài hoặc các sản phẩm SIEM sử dụng, lưu trữ, xử lý và trình bày dữ liệu để truy vấn.

Ngoài nhật ký, Zeek còn có chức năng tích hợp sẵn cho một loạt nhiệm vụ phân tích và phát hiện, bao gồm trích xuất tệp từ phiên HTTP, phát hiện phần mềm độc hại bằng cách can thiệp vào cơ quan đăng ký bên ngoài, báo cáo các phiên bản phần mềm dễ bị tấn công nhìn thấy trên mạng, xác định trang web phổ biến. các ứng dụng, phát hiện hành vi cưỡng bức SSH, xác thực chuỗi chứng chỉ SSL, v.v.

Ngoài việc cung cấp chức năng mạnh mẽ như vậy ngay lập tức, Zeek còn là một nền tảng có thể mở rộng và tùy chỉnh hoàn toàn để phân tích lưu lượng truy cập. Zeek cung cấp cho người dùng một ngôn ngữ kịch bản hoàn chỉnh Turing dành riêng cho miền để thể hiện

các tác vụ phân tích tùy ý. Hãy coi ngôn ngữ Zeek như một "Python dành riêng cho miền" (hoặc Perl): giống như Python, hệ thống này đi kèm với một bộ chức năng dựng sẵn ("thư viện tiêu chuẩn"), tuy nhiên người dùng cũng có thể đưa Zeek vào sử dụng theo những cách mới lạ bằng cách viết mã tùy chỉnh. Thật vậy, tất cả các phân tích mặc định của Zeek, bao gồm cả việc ghi nhật ký, đều được thực hiện thông qua các tập lệnh; không có phân tích cụ thể nào được mã hóa cứng vào cốt lõi của hệ thống.

Zeek chạy trên phần cứng thông dụng và do đó cung cấp giải pháp thay thế chi phí thấp cho các giải pháp độc quyền đắt tiền. Theo nhiều cách, Zeek vượt xa khả năng của các công cụ giám sát mạng khác, vốn thường bị giới hạn ở một nhóm nhỏ các tác vụ phân tích được mã hóa cứng. Zeek không phải là hệ thống phát hiện xâm nhập dựa trên chữ ký (IDS) cổ điển; mặc dù nó cũng hỗ trợ chức năng tiêu chuẩn như vậy nhưng ngôn ngữ kịch bản của Zeek tạo điều kiện cho nhiều cách tiếp cận rất khác nhau để tìm ra hoạt động độc hại. Chúng bao gồm phát hiện lạm dụng ngữ nghĩa, phát hiện bất thường và phân tích hành vi.

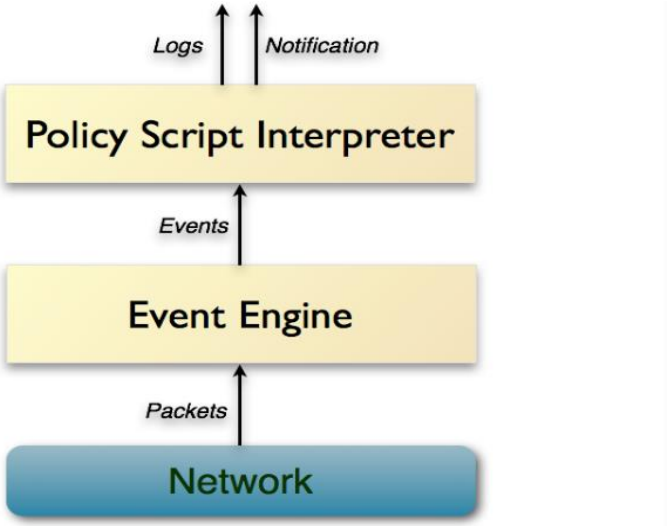
Rất nhiều trang web triển khai Zeek để bảo vệ cơ sở hạ tầng của họ, bao gồm nhiều trường đại học, phòng thí nghiệm nghiên cứu, trung tâm siêu máy tính, cộng đồng khoa học mở, các tập đoàn lớn và các cơ quan chính phủ. Zeek đặc biệt nhắm mục tiêu giám sát mạng tốc độ cao, khối lượng lớn và ngày càng nhiều trang web hiện đang sử dụng hệ thống để giám sát mạng 10GE của họ, trong đó một số đã chuyển sang liên kết 100GE.

Zeek cung cấp các cài đặt hiệu suất cao bằng cách hỗ trợ cân bằng tải có thể mở rộng. Các trang web lớn thường chạy "Zeek Clusters" trong đó bộ cân bằng tải giao diện người dùng tốc độ cao phân phối lưu lượng trên một số lượng PC phụ trợ thích hợp, tất cả đều chạy các phiên bản Zeek chuyên dụng trên các lát lưu lượng riêng lẻ của chúng. Hệ thống quản lý trung tâm điều phối quy trình, đồng bộ hóa trạng thái trên các mặt sau và cung cấp cho người vận hành giao diện quản lý trung tâm để định cấu hình và truy cập vào nhật ký tổng hợp. Khung quản lý tích hợp của Zeek, ZeekControl, hỗ trợ các thiết lập cụm như vậy ngay lập tức.

Các tính năng cụm của Zeek hỗ trợ thiết lập một hệ thống và đa hệ thống. Đó là một phần lợi thế về khả năng mở rộng của Zeek. Tóm lại, Zeek được tối ưu hóa để diễn giải lưu lượng mạng và tạo nhật ký dựa trên lưu lượng đó. Nó không được tối ưu hóa để khớp byte và người dùng đang tìm kiếm các phương pháp phát hiện chữ ký sẽ được phục vụ tốt hơn bằng cách thử các hệ thống phát hiện xâm nhập như Suricata. Zeek cũng không phải là một công cụ phân tích giao thức theo nghĩa của Wireshark, đang tìm cách mô tả mọi thành phần của lưu lượng mạng ở cấp khung hoặc một hệ thống lưu trữ lưu lượng ở dạng chụp gói (PCAP). Đúng hơn, Zeek ngồi ở "phương tiện vui vẻ" đại diện cho nhật ký mạng nhỏ gọn nhưng có độ trung thực cao, giúp hiểu rõ hơn về lưu lượng và cách sử dụng mạng.

Tóm lại, Zeek được tối ưu hóa để diễn giải lưu lượng mạng và tạo nhật ký dựa trên lưu lượng đó. Nó không được tối ưu hóa để đối sánh byte và người dùng tìm kiếm các phương pháp phát hiện chữ ký sẽ được phục vụ tốt hơn bằng cách thử các hệ thống phát hiện xâm nhập như Suricata. Zeek cũng không phải là một công cụ phân tích giao thức theo nghĩa của Wireshark, tìm cách mô tả mọi yếu tố của lưu lượng mạng ở cấp khung hoặc một hệ thống lưu trữ lưu lượng ở dạng bắt gói (PCAP). Thay vào đó, Zeek nằm ở "phương tiện hài lòng" thể hiện nhật ký mạng nhỏ gọn nhưng có độ trung thực cao, giúp hiểu rõ hơn về lưu lượng mạng và việc sử dụng.

2. Kiến trúc của Zeek.



Hình ảnh 1. Kiến trúc của Zeek

- Zeek được chia thành hai thành phần chính về mặt kiến trúc. “Event Engine” và “Policy Script Interpreter. Lớp Event Engine là nơi các gói được xử lý; nó được gọi là lõi sự kiện và chịu trách nhiệm mô tả sự kiện mà không tập trung vào chi tiết sự kiện. Đó là nơi các gói được chia thành các phần như địa chỉ nguồn và đích, nhận dạng giao thức, phân tích phiên và trích xuất tệp. Lớp Policy Script Interpreter là nơi tiến hành phân tích ngữ nghĩa. Nó chịu trách nhiệm mô tả các mối tương quan của sự kiện bằng cách sử dụng tập lệnh Zeek.

- Trình thông dịch kịch bản của Zeek: Trong khi bản thân công cụ sự kiện là trung lập về chính sách, thì lớp trên cùng của Zeek xác định chính sách an ninh mạng theo môi trường cụ thể. Bằng cách viết các trình xử lý cho các sự kiện có thể do cỗ máy sự kiện nâng lên, người dùng có thể xác định chính xác các ràng buộc trong mạng. Nếu một vi phạm bảo mật được phát hiện, lớp chính sách sẽ tạo ra một cảnh báo. Có thể tạo trình xử lý sự kiện mới bằng ngôn ngữ kịch bản riêng của Zeek.

- Các bộ phận phân tích của Zeek: Phần lớn các bộ phân tích của Zeek nằm trong công cụ sự kiện của nó với các tập lệnh chính sách đi kèm có thể được tùy chỉnh bởi người dùng. Tuy nhiên, đôi khi, bộ phân tích chỉ là một chính sách tập lệnh triển khai nhiều trình xử lý sự kiện. Các bộ phân tích thực hiện lớp ứng dụng giải mã, phát hiện bất thường, đối sánh chữ ký và phân tích kết nối. Zeek đã được được thiết kế để có thể dễ dàng thêm các bộ phân tích bổ sung.

- Chữ ký: hầu hết các hệ thống phát hiện xâm nhập mạng (NIDS) phù hợp với một tập hợp lớn các chữ ký chống lại lưu lượng mạng. Ở đây, chữ ký là một mẫu byte mà NIDS cố gắng xác định vị trí tải trọng của các gói mạng. Ngay sau khi tìm thấy một kết quả phù hợp, hệ thống sẽ tạo ra một báo động. Một hệ thống IDS nổi tiếng là Snort; ngược lại, cách tiếp cận chung của Zeek để xâm nhập phát hiện có phạm vi rộng hơn nhiều so với đối sánh chữ ký truyền thống, nhưng vẫn chứa một công cụ chữ ký cung cấp một chức năng tương tự như của các hệ thống khác.

- ZeekControl: trước đây được gọi là BroControl, là một trình bao tương tác để dễ dàng vận hành và quản lý các cài đặt Zeek trên một hệ thống hoặc trên nhiều hệ thống trong một cụm điều khiển lưu lượng.

3. Cách hoạt động của Zeek.

- Quản lý trạng thái: Cấu trúc dữ liệu chính của Zeek là một kết nối tuân theo nhận dạng luồng điển hình các cơ chế, chẳng hạn như phương pháp tiếp cận 5-tuple. Cấu trúc 5 tuple bao gồm IP nguồn địa chỉ / số cổng, địa chỉ IP đích / số cổng và giao thức đang sử dụng. Cho

một giao thức hướng kết nối như TCP, định nghĩa về kết nối rõ ràng hơn, tuy nhiên đối với những người khác như UDP và ICMP, Zeek triển khai một sự trừu tượng giống như luồng để tổng hợp các gói tin. Mỗi gói thuộc về chính xác một kết nối.

- Phân tích tầng giao vận: Trên lớp giao vận, Zeek phân tích các gói TCP, UDP. Trong TCP, trình phân tích liên kết của Zeek theo dõi chặt chẽ các thay đổi trạng thái khác nhau, theo dõi các xác nhận, xử lý truyền lại và nhiều hơn nữa.

- Phân tích tầng ứng dụng: Việc phân tích dữ liệu lớp ứng dụng của một kết nối phụ thuộc vào dịch vụ. Ở đó là máy phân tích cho nhiều loại giao thức khác nhau, ví dụ: HTTP, SMTP hoặc DNS, thường tiến hành phân tích chi tiết luồng dữ liệu.

III. Zeek Scripting.

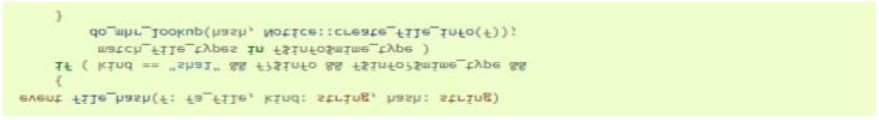
Zeek bao gồm một ZeekScripts hướng sự kiện cung cấp phương tiện chính cho một tổ chức để mở rộng và tùy chỉnh chức năng của Zeek. Trên thực tế, hầu như tất cả đầu ra do Zeek tạo ra đều được tạo bởi Zeek scripts. Gần như dễ dàng hơn khi coi Zeek là một thực thể xử lý các kết nối hậu trường và tạo ra các sự kiện trong khi ZeekScripts là phương tiện mà thông qua đó chúng ta bình thường có thể giao tiếp. Các tập lệnh Zeek thông báo một cách hiệu quả cho Zeek rằng nếu có một sự kiện thuộc loại mà chúng tôi xác định, sau đó cho chúng tôi biết thông tin về kết nối để chúng tôi có thể thực hiện một số chức năng trên đó.

1. Tập lệnh.

Thông thường, dễ hiểu nhất về ngôn ngữ ZeekScripts bằng cách xem một tập lệnh hoàn chỉnh và chia nhỏ thành các thành phần có thể nhận dạng được.

- Trong ví dụ này, chúng ta sẽ xem xét cách Zeek kiểm tra hàm băm SHA1 của các tệp khác nhau được trích xuất từ lưu lượng mạng dựa trên modul Team Cymru Malware hash registry. Team Cymru cũng điền vào bản ghi TXT của các phản hồi DNS của họ với cả dấu thời gian “được nhìn thấy lần đầu” và “tỷ lệ phát hiện” bằng số. Khía cạnh quan trọng cần hiểu là Zeek đã tạo hàm băm cho các tệp thông qua Files , framework, nhưng đó là policy/frameworks/files/detect-MHR.zeek chịu trách nhiệm tạo bản tra cứu DNS thích hợp, phân tích cú pháp phản hồi và tạo thông báo thích hợp.

```
@load base/frameworks/notice
@load frameworks/files/hash-all-files

module TeamCymruMalwareHashRegistry;

export {
    redef enum Notice::Type += {
        ## The hash value of a file transferred over HTTP matched in the
        ## malware hash registry.
        Match
    };

    ## File types to attempt matching against the Malware Hash Registry.
    option match_file_types = /application\/x-dosexec/ |
                              /application\/vnd.ms-cab-compressed/ |
                              /application\/pdf/ |
                              /application\/x-shockwave-flash/ |
                              /application\/x-java-applet/ |
                              /application\/jar/ |
                              /video\/mp4/;

    ## The Match notice has a sub message with a URL where you can get more
    ## information about the file. The %s will be replaced with the SHA-1
    ## hash of the file.
    option match_sub_url = "https://www.virustotal.com/en/search/?query=%s";

    ## The malware hash registry runs each malware sample through several
    ## A/V engines.  Team Cymru returns a percentage to indicate how
    ## many A/V engines flagged the sample as malicious. This threshold
    ## allows you to require a minimum detection rate.
    option notice_threshold = 10;
}

function do_mhr_lookup(hash: string, fi: Notice::FileInfo)
    {
    local hash_domain = fmt("%s.malware.hash.cymru.com", hash);

    when ( local MHR_result = lookup_hostname_txt(hash_domain) )
        {
        # Data is returned as "<dateFirstDetected> <detectionRate>"
        local MHR_answer = split_string1(MHR_result, / /);

        if ( |MHR_answer| == 2 )
            {
            local mhr_detect_rate = to_count(MHR_answer[1]);

            if ( mhr_detect_rate >= notice_threshold )
                {
                local mhr_first_detected = double_to_time(to_double(MHR_answer[0]));
                local readable_first_detected = strftime("%Y-%m-%d %H:%M:%S", mhr_first_detected);
                local message = fmt("Malware Hash Registry Detection rate: %d%%  Last seen: %s", mhr_de
                local virustotal_url = fmt(match_sub_url, hash);
                # We don't have the full fa_file record here in order to
                # avoid the "when" statement cloning it (expensive!).
                local n: Notice::Info = Notice::Info($note=Match, $msg=message, $sub=virustotal_url);
                Notice::populate_file_info2(fi, n);
                NOTICE(n);
                }
            }
        }
    }
```



Hình ảnh 2. Tập lệnh.

- Có ba phần riêng biệt của ZeekScripting. Đầu tiên, gồm các thư viện được bao gồm trong tập lệnh thông qua **@load** và một không gian tên được xác định với **module**. Tiếp theo là phần được định dạng và giải thích các biến tùy chỉnh đang được cung cấp **(export)** như một phần của không gian tên của tập lệnh. Cuối cùng, có một phần được định dạng và thụt lề thứ hai mô tả các hướng dẫn cần thực hiện cho một sự kiện cụ thể event file_hash( ).

Phần đầu tiên của script bao gồm các lệnh **@load** xử lý **__load__.zeek** script trong các thư mục tương ứng đang được tải. Các @load chỉ thị đảm bảo Files framework, Notice framework và tập lệnh băm tất cả các tệp đã được tải bởi Zeek.

```
@load base/frameworks/files
@load base/frameworks/notice
@load frameworks/files/hash-all-files
```

Hình ảnh 3. Thư viện.

Phần export xác định lại một hằng số mô tả loại thông báo sẽ tạo với Notice framework. Bằng cách mở rộng **Notice::Type** như được hiển thị, điều này cho phép hàm **NOTICE** tạo ra các thông báo với trường $note được chỉ định là TeamCymruMalwareHashRegistry:: Match. Notices cho phép Zeek tạo một số loại thông báo bổ sung ngoài các loại nhật ký mặc định của nó.Với phần tiếp theo, tập lệnh bắt đầu xác định các hướng dẫn để thực hiện trong một sự kiện nhất định.

```
export {
    redef enum Notice::Type += {
        ## The hash value of a file transferred over HTTP matched in the
        ## malware hash registry.
        Match
    };

    ## File types to attempt matching against the Malware Hash Registry.
    option match_file_types = /application\/x-dosexec/ |
                              /application\/vnd.ms-cab-compressed/ |
                              /application\/pdf/ |
                              /application\/x-shockwave-flash/ |
                              /application\/x-java-applet/ |
                              /application\/jar/ |
                              /video\/mp4/;

    ## The Match notice has a sub message with a URL where you can get more
    ## information about the file. The %s will be replaced with the SHA-1
    ## hash of the file.
    option match_sub_url = "https://www.virustotal.com/en/search/?query=%s";

    ## The malware hash registry runs each malware sample through several
    ## A/V engines.  Team Cymru returns a percentage to indicate how
    ## many A/V engines flagged the sample as malicious. This threshold
    ## allows you to require a minimum detection rate.
    option notice_threshold = 10;
}
```

Hình ảnh 4. Phần export.

- Đặc tính của tập lệnh được chứa trong trình xử lý event cho **file_hash**. Sự kiện này file_hash cho phép các tập lệnh truy cập thông tin được liên kết với một tệp mà khung phân tích tệp của Zeek đã tạo ra hàm băm.

- Trong trình xử lý sự kiện **file_hash**, có một câu lệnh if được sử dụng để kiểm tra loại băm chính xác, trong trường hợp này là băm SHA1.Nó cũng kiểm tra kiểu mime type mà đã được xác định là được chú ý như được định nghĩa trong constant **match_file_types** . So sánh được thực hiện dựa trên biểu thức **f$info$mime_type**, sử dụng toán tử **$** tham chiếu để kiểm tra giá trị **mime_type** bên trong biến **f$info**. Nếu toàn bộ biểu thức đánh giá là true, thì một hàm phụ

được gọi để thực hiện phần còn lại của công việc. Trong hàm đó, một biến cục bộ được định nghĩa để chứa một chuỗi bao gồm hàm băm SHA1 được nối với **.malware.hash.cymru.com**; giá trị này sẽ là miền được truy vấn trong the malware hash registry (sổ đăng ký băm phần mềm độc hại)

- Thời gian phát hiện được xử lý thành một chuỗi biểu diễn và được lưu trữ trong **readable_first_detected**. Sau đó, tập lệnh so sánh tỷ lệ phát hiện với tỷ lệ **notice_threshold** đã được xác định trước đó. Nếu tỷ lệ phát hiện đủ cao, tập lệnh sẽ tạo mô tả ngắn gọn về thông báo và lưu trữ trong biến **message**. Nó cũng tạo ra một URL khả thi để kiểm tra mẫu dựa **virustotal.com** trên cơ sở dữ liệu của và thực hiện lệnh gọi để **NOTICE** chuyển thông tin liên quan vào khung Thông báo.

```
function do_mhr_lookup(hash: string, fi: Notice::FileInfo)
    {
    local hash_domain = fmt("%s.malware.hash.cymru.com", hash);

    when ( local MHR_result = lookup_hostname_txt(hash_domain) )
        {
        # Data is returned as "<dateFirstDetected> <detectionRate>"
        local MHR_answer = split_string1(MHR_result, / /);

        if ( |MHR_answer| == 2 )
            {
            local mhr_detect_rate = to_count(MHR_answer[1]);

            if ( mhr_detect_rate >= notice_threshold )
                {
                local mhr_first_detected = double_to_time(to_double(MHR_answer[0]));
                local readable_first_detected = strftime("%Y-%m-%d %H:%M:%S", mhr_first_detected);
                local message = fmt("Malware Hash Registry Detection rate: %d%%  Last seen: %s", mhr_det
                local virustotal_url = fmt(match_sub_url, hash);
                # We don't have the full fa_file record here in order to
                # avoid the "when" statement cloning it (expensive!).
                local n: Notice::Info = Notice::Info($note=Match, $msg=message, $sub=virustotal_url);
                Notice::populate_file_info2(fi, n);
                NOTICE(n);
                }
            }
        }
    }

event file_hash(f: fa_file, kind: string, hash: string)
    {
    if ( kind == "sha1" && f?$info && f$info?$mime_type &&
         match_file_types in f$info$mime_type )
        do_mhr_lookup(hash, Notice::create_file_info(f));
```

Hình ảnh 5. function và event.

2. Trình xử lý sự kiện.

ZeekScripts được điều khiển bởi event, đây là một sự thay đổi cần thiết so với phần lớn các ngôn ngữ lập trình kịch bản mà hầu hết người dùng sẽ có kinh nghiệm trước đó. Việc viết Scripting trong Zeek phụ thuộc vào việc xử lý các sự kiện do Zeek tạo ra khi nó xử lý lưu lượng mạng, thay đổi trạng thái của cấu trúc dữ liệu thông qua các sự kiện đó và đưa ra quyết định về thông tin được cung cấp. Cách tiếp cận tập lệnh này thường có thể gây nhầm lẫn cho người dùng đến với Zeek từ ngôn ngữ thủ tục hoặc chức năng, nhưng một khi cú sốc ban đầu hết, nó sẽ trở nên rõ ràng hơn với mỗi lần tiếp xúc.

Cốt lõi của Zeek hoạt động để đặt các sự kiện vào một “event queue” (hàng đợi sự kiện) có thứ tự, cho phép người xử lý sự kiện xử lý chúng trên cơ sở ai đến trước được phục vụ trước. Trên thực tế, đây là chức năng cốt lõi của Zeek vì nếu không có các tập lệnh được viết để thực hiện các hành động rời rạc trên các sự kiện, sẽ có rất ít hoặc không có đầu ra có thể sử dụng được. Như vậy, hiểu biết cơ bản về hàng đợi sự kiện, các sự kiện đang được tạo và cách mà trình xử lý sự kiện xử lý các sự kiện đó là cơ sở để không chỉ học viết script cho Zeek mà còn để hiểu về chính Zeek.

```
## Generated for DNS requests. For requests with multiple queries, this event
## is raised once for each.
##
## See `Wikipedia <http://en.wikipedia.org/wiki/Domain_Name_System>`__ for more
## information about the DNS protocol. Zeek analyzes both UDP and TCP DNS
## sessions.
##
## c: The connection, which may be UDP or TCP depending on the type of the
##    transport-layer session being analyzed.
##
## msg: The parsed DNS message header.
##
## query: The queried name.
##
## qtype: The queried resource record type.
##
## qclass: The queried resource record class.
##
## .. zeek:see:: dns_AAAA_reply dns_A_reply dns_CNAME_reply dns_EDNS_addl
##    dns_HINFO_reply dns_MX_reply dns_NS_reply dns_PTR_reply dns_SOA_reply
##    dns_SRV_reply dns_TSIG_addl dns_TXT_reply dns_WKS_reply dns_end
##    dns_full_request dns_mapping_altered dns_mapping_lost_name dns_mapping_new_name
##    dns_mapping_unverified dns_mapping_valid dns_message dns_query_reply
##    dns_rejected non_dns_request dns_max_queries dns_session_timeout dns_skip_addl
##    dns_skip_all_addl dns_skip_all_auth dns_skip_auth
event dns_request%(c: connection, msg: dns_msg, query: string, qtype: count, qclass: count%);
```

Hình ảnh 6. Tài liệu cho sự kiện dns_request.

Trên đây là một đoạn của tài liệu cho sự kiện dns_request. Nó được tổ chức sao cho documentation, commentary và list các đối số đứng trước định nghĩa sự kiện thực tế được Zeek sử dụng. Khi Zeek phát hiện các yêu cầu DNS được cấp bởi một người khởi tạo, nó sẽ đưa ra sự kiện này và bất kỳ số lượng tập lệnh nào sau đó có quyền truy cập vào dữ liệu mà Zeek chuyển cùng với sự kiện. Trong ví dụ này, Zeek không chỉ chuyển thông báo, truy vấn, loại truy vấn và lớp truy vấn cho yêu cầu DNS mà còn chuyển một bản ghi được sử dụng cho chính kết nối.

3. Frameworks.

Zeek bao gồm một số frameworks cung cấp chức năng thường được sử dụng cho lớp scripting. Ngoài những thứ khác, các khung công tác này nâng cao khả năng của Zeek trong việc nhập dữ liệu, cấu trúc và lọc đầu ra của nó, điều chỉnh cài đặt trong thời gian chạy và tương tác với các thành phần khác trong mạng của bạn. Hầu hết các khuôn khổ bao gồm chức năng được triển khai trong lõi của Zeek, với các cấu trúc dữ liệu và API tương ứng được hiển thị trên lớp tập lệnh.

Zeek có một số framworks để cung cấp chức năng mở rộng trong lớp tập lệnh. Các khung này nâng cao tính linh hoạt và khả năng tương thích của Zeek với các thành phần mạng khác. Mỗi khung tập trung vào trường hợp sử dụng cụ thể và dễ dàng chạy khi cài đặt Zeek.

- Các framework cũng được xây dựng dựa trên nhau, ta có một số Framework sau:

- Logging Framework
- Notice Framework
- Input Framework
- Signature Framework
- Configuration Framework
- Intelligence Framework
- Configuration Framework

- Management Framework

a, Logging Framework.

Zeek đi kèm với giao diện ghi nhật ký dựa trên khóa-giá trị linh hoạt cho phép kiểm soát chi tiết những gì được ghi lại và cách nó được ghi lại.

- Giao diện ghi nhật ký của Zeek được xây dựng dựa trên ba điểm trừu tượng chính:
  - Streams: Một dòng nhật ký tương ứng với một nhật ký duy nhất. Nó xác định tập hợp các trường mà nhật ký bao gồm với tên và kiểu của chúng. Ví dụ như luồng kết nối để ghi tóm tắt kết nối và luồng http để ghi lại hoạt động HTTP.
  - Filters: Mỗi luồng có một tập hợp các bộ lọc được đính kèm để xác định thông tin nào được ghi ra ngoài và cách thức. Theo mặc định, mỗi luồng có một bộ lọc mặc định chỉ ghi mọi thứ trực tiếp vào đĩa. Tuy nhiên, các bộ lọc bổ sung có thể được thêm vào để chỉ ghi lại một tập hợp con của các bản ghi nhật ký, ghi vào các đầu ra khác nhau hoặc đặt khoảng thời gian xoay vòng tùy chỉnh. Nếu tất cả các bộ lọc bị xóa khỏi một luồng, thì đầu ra sẽ bị tắt cho luồng đó.
  - Writers: Xác định định dạng đầu ra thực tế cho thông tin đang được ghi. Trình ghi mặc định là trình ghi ASCII, tạo ra các tệp ASCII được phân tách bằng tab.

b, Notice Framework.

Zeek cung cấp một số lượng lớn các kịch bản chính sách thực hiện nhiều phân tích khác nhau. Hầu hết các tập lệnh này đều giám sát hoạt động mà người dùng có thể quan tâm. Tuy nhiên, không có tập lệnh nào trong số này xác định tầm quan trọng của những gì nó tìm thấy. Thay vào đó, các tập lệnh chỉ gắn cờ các tình huống có khả năng thú vị, để nó ở cấu hình cục bộ để xác định tình huống nào trong số đó trên thực tế có thể xử lý được. Việc tách rời hoạt động phát hiện và báo cáo này cho phép Zeek giải quyết các nhu cầu khác nhau mà các trang web khác nhau có. Các định nghĩa về những gì cấu thành một cuộc tấn công hoặc thậm chí là một sự thỏa hiệp khác nhau khá nhiều giữa các môi trường và hoạt động được coi là độc hại ở một trang web có thể hoàn toàn được chấp nhận ở một trang web khác.

Bất cứ khi nào một trong các tập lệnh phân tích của Zeek thấy điều gì đó, nó sẽ gắn cờ tình huống bằng cách gọi **NOTICE** hàm và tạo cho nó một **Notice::Info** bản ghi duy nhất. Thông báo có một **Notice::Type** , phản ánh loại hoạt động đã được nhìn thấy, và nó cũng thường được bổ sung thêm bối cảnh về tình huống. Có thể tìm thêm thông tin về việc nâng cao thông báo trong phần Nâng cao thông báo.

Hoạt động	Sự miêu tả
`Notice::ACTION_LOG`	Viết thông báo vào `Notice::LOG` luồng ghi nhật ký.
`Notice::ACTION_ALARM`	Đăng nhập vào `Notice::ALARM_LOG` luồng sẽ luân chuyển hàng giờ và gửi nội dung qua email đến địa chỉ email hoặc các địa chỉ trong trường *email_dest* của `Notice::Info` bản ghi thông báo đó.
`Notice::ACTION_EMAIL`	Gửi thông báo bằng email đến địa chỉ email hoặc các địa chỉ trong trường *email_dest* của `Notice::Info` bản ghi thông báo đó.
`Notice::ACTION_PAGE`	Gửi email đến địa chỉ email hoặc các địa chỉ trong trường *email_dest* của `Notice::Info` bản ghi thông báo đó.

Hình ảnh 7. Thông báo nâng cao.

- **Thông báo xử lý**:

Chính sách Thông báo

  - Hook **Notice::policy** cung cấp cơ chế để áp dụng các hành động và thường

sửa đổi thông báo trước khi nó được gửi tới các plugin hành động. Hook có thể được coi là các chức năng đa thân và việc sử dụng chúng trông rất giống với việc xử lý các sự kiện. Sự khác biệt là chúng không đi qua hàng đợi sự kiện như các sự kiện. Người dùng có thể thay đổi quá trình xử lý thông báo bằng cách sửa đổi trực tiếp các trường trong **Notice::Info** bản ghi được cung cấp làm đối số cho hook.

Các phím tắt chính sách thông báo

- Mặc dù khung thông báo cung cấp rất nhiều tính linh hoạt và khả năng cấu hình, nhưng nhiều khi không cần đến sự thể hiện đầy đủ và thực sự trở thành một trở ngại cho việc đạt được kết quả. Khung công tác cung cấp một nội dung Notice::policy hook mặc định như một cách cung cấp cho người dùng các phím tắt để dễ dàng áp dụng nhiều hành động phổ biến cho các thông báo. Chúng được triển khai dưới dạng các tập hợp và bảng được lập chỉ mục với một Notice::Type giá trị enum. Sau đây hiển thị và mô tả tất cả các biến có sẵn cho cấu hình phím tắt của khung thông báo.

Tên biến	Sự miêu tả
`Notice::ignored_types`	Việc thêm a `Notice::Type` vào bộ này sẽ dẫn đến thông báo bị bỏ qua. Nó sẽ không có bất kỳ hành động nào khác được áp dụng cho nó, thậm chí cả `Notice::ACTION_LOG`.
`Notice::emailed_types`	Việc thêm a `Notice::Type` vào bộ này sẽ dẫn đến `Notice::ACTION_EMAIL` việc áp dụng cho các thông báo thuộc loại đó.
`Notice::alarmed_types`	Việc thêm a `Notice::Type` vào bộ này sẽ dẫn đến `Notice::ACTION_ALARM` việc áp dụng cho các thông báo thuộc loại đó.
`Notice::not_suppressed_types`	Việc thêm a `Notice::Type` vào nhóm này sẽ dẫn đến thông báo đó không còn trải qua quá trình chặn thông báo thông thường sẽ diễn ra nữa. Hãy cẩn thận khi sử dụng tính năng này trong sản xuất, nó có thể làm tăng đáng kể số lượng thông báo được xử lý.
`Notice::type_suppression_intervals`	Đây là một bảng được lập chỉ mục `Notice::Type` và mang lại một khoảng thời gian. Nó có thể được sử dụng như một cách dễ dàng để kéo dài khoảng thời gian loại bỏ mặc định cho toàn bộ `Notice::Type` mà không cần phải tạo toàn bộ `Notice::policy` mục nhập và thiết lập `$suppress_for` trường.

Hình ảnh 8. Các biến cấu hình phím tắt.

Đưa ra thông báo

- NOTICE là một hàm bình thường trong không gian tên chung bao bọc một hàm trong không gian tên Thông báo. Phải mất một đối số duy nhất của Notice::Infoloại bản ghi. Các trường phổ biến nhất được sử dụng khi đưa ra thông báo được mô tả trong bảng sau:

Tên trường	Sự miêu tả
`$note`	Trường này là bắt buộc và là giá trị enum đại diện cho loại thông báo.
`$msg`	Đây là thông báo con người có thể đọc được nhằm cung cấp thêm thông tin về trường hợp cụ thể này của loại thông báo.
`$sub`	Đây là một tin nhắn phụ dành cho con người có thể đọc được nhưng cũng sẽ thường xuyên được sử dụng để chứa dữ liệu phù hợp với phần mở rộng `Notice::policy` .
`$conn`	Nếu bản ghi kết nối có sẵn khi thông báo được đưa ra và thông báo thể hiện một số thuộc tính của kết nối thì bản ghi kết nối có thể được cung cấp tại đây. Các trường khác như $id và $src sẽ tự động được điền từ giá trị này.
`$id`	Nếu một `conn_id` bản ghi có sẵn khi thông báo được đưa ra và thông báo thể hiện một số thuộc tính của kết nối thì kết nối có thể được đưa ra ở đây. Các trường khác như `$src` sẽ tự động được điền từ giá trị này.
`$src`	Nếu thông báo đại diện cho một thuộc tính của một máy chủ thì có thể chỉ điền vào trường này để thể hiện máy chủ đang được "chú ý".
`$n`	Điều này thường đại diện cho một con số nếu thông báo liên quan đến một con số nào đó. Nó được sử dụng thường xuyên nhất cho các bài kiểm tra số trong việc `Notice::policy` đưa ra các quyết định chính sách.
`$identifier`	Điều này đại diện cho một mã định danh duy nhất cho thông báo này. Trường này được mô tả chi tiết hơn trong phần Loại bỏ tự động .
`$suppress_for`	Trường này có thể được đặt nếu có khoảng thời gian loại bỏ tự nhiên cho thông báo có thể khác với giá trị mặc định. Giá trị được đặt cho trường này cũng có thể được người dùng sửa đổi `Notice::policy` để giá trị không được đặt vĩnh viễn và không thể thay đổi.

Hình ảnh 9. Các trường phổ biến nhất được sử dụng khi đưa ra thông báo.

Ức chế tự động

- Khung thông báo hỗ trợ loại bỏ các thông báo nếu tác giả của tập lệnh tạo thông báo đã chỉ ra cho khung thông báo cách xác định các thông báo về bản chất giống nhau. Việc xác định các thông báo "trùng lặp nội tại" này được triển khai bằng một trường tùy chọn trong Notice::Infocác bản ghi có tên $identifierlà một chuỗi đơn giản. Nếu các trường $identifiervà $notegiống nhau cho hai thông báo thì khung thông báo thực sự coi chúng là giống nhau và có thể sử dụng thông tin đó để loại bỏ các bản sao trong một khoảng thời gian có thể định cấu hình.

- **Mở rộng khung thông báo:**

Định cấu hình email thông báo

- Định cấu hình email thông báo Nếu Notice::mail_dest được đặt, các thông báo với một hành động e-mail liên quan sẽ được gửi đến địa chỉ đó. Để có thêm tùy chỉnh, người dùng có thể sử dụng Notice::policy hook để sửa đổi trường email_dest . Ví dụ sau sẽ tạo ra 3 email riêng biệt.

```
hook Notice::policy(n: Notice::Info)
  {
  n$email_dest = set(
      "snow.white@example.net",
      "doc@example.net",
      "happy@example.net,sleepy@example.net,bashful@example.net"
  );
  }
```

Hình ảnh 10. Ví dụ cấu hình email.

- Nếu có thêm thông tin mà bạn muốn thêm vào email, bạn có thể thêm thông tin đó bằng cách viết Notice::policy hook. Có một trường trong Notice::Info bản ghi có tên $ email_body_section sẽ được bao gồm nguyên văn khi email được gửi. Dưới đây là một ví dụ về việc bao gồm một số thông tin từ một yêu cầu HTTP.

```
hook Notice::policy(n: Notice::Info)
  {
  if ( n?$conn && n$conn?$http && n$conn$http?$host )
    n$email_body_sections[|n$email_body_sections|] = fmt("HTTP host header: %s", n$conn$http$host);
  }
```

Hình ảnh 11. Một số thông tin từ một yêu cầu HTTP.

Xem xét cụm

- Khi chạy Zeek trong một cụm, hầu hết thông tin ở trên được giữ nguyên. Thông báo được tạo, Notice::policy hook được đánh giá và mọi hành động được chạy trên nút tạo ra thông báo (thường là nút worker). Lưu ý đối với người dùng / nhà phát triển của Zeek là mọi tệp hoặc quyền truy cập cần thiết để chạy các hành động thông báo phải có sẵn cho (các) nút tương ứng.
- Vai trò của người quản lý là nhận và phân phối thông tin ngăn chặn thông báo, để các thông báo trùng lặp không được tạo ra. Hãy nhớ rằng có một số độ trễ nội tại trong quá trình đồng bộ hóa này, vì vậy có thể các thông báo tạo nhanh sẽ được lặp lại (và trong trường hợp này, bất kỳ hành động nào sẽ được thực hiện nhiều lần, một lần bởi mỗi nhân viên tạo thông báo).

Nhật ký kỳ lạ

- Một loạt các hoạt động "kỳ lạ" được Zeek phát hiện có thể kích hoạt các sự kiện tương ứng thông báo cho lớp kịch bản về hoạt động này. Các sự kiện này tồn tại ở nhiều mức độ chi tiết khác nhau, bao gồm , và **conn_weird** các sự kiện khác. Được xây dựng trên đỉnh khung thông báo, mô-đun Weird triển khai các trình xử lý sự kiện có chức năng chuyển các "điểm bất thường" khác nhau vào các trình xử lý khung thông báo thông thường. Để có ý tưởng về các kiểu kỳ lạ có sẵn, hãy xem **flow_weirdnet_weirdfile_weirdWeird::actions** bảng, xác định các hành động mặc định cho các loại hoạt động khác nhau. Weirds thường không chỉ ra hoạt động liên quan đến bảo mật - chúng chỉ là những điều kỳ lạ mà bạn thường không mong đợi xảy ra, chẳng hạn như vi phạm máy trạng thái TCP kỳ lạ, chòm sao tiêu đề HTTP không mong muốn hoặc thuộc tính thông báo DNS nằm ngoài các thông số kỹ thuật RFC có liên quan. Đó là, đừng coi chúng là những phát hiện có thể hành động theo nghĩa IDS, mặc dù chúng cũng có thể cung cấp manh mối bổ sung có ý nghĩa cho một sự cố bảo mật.
- Loại thông báo cho các điểm bất thường là Activity . Bạn có nhiều hành động theo ý mình để biết cách xử lý các điểm bất thường: bạn có thể bỏ qua chúng,

ghi lại chúng hoặc yêu cầu chúng kích hoạt thông báo, tất cả ở nhiều mức độ giảm / lọc khác nhau (xem các Weird::Action giá trị enum để biết thêm chi tiết). Đối với lọc động, bộ Weird::ignore_hosts và Weird::weird_ignore cho phép loại trừ hoạt động khỏi báo cáo.

c, Signature Framework.

Zeek chủ yếu dựa vào ngôn ngữ kịch bản mở rộng của mình để xác định và phân tích các chính sách phát hiện, nhưng nó cũng cung cấp một ngôn ngữ chữ ký độc lập để thực hiện khớp mẫu kiểu Snort ở cấp độ thấp. Mặc dù chữ ký không phải là công cụ phát hiện ưa thích của Zeek nhưng đôi khi chúng có ích và gần với những gì mà nhiều người quen thuộc khi sử dụng NIDS khác. Trang này cung cấp một cái nhìn tổng quan ngắn gọn về chữ ký của Zeek và đề cập đến một số chi tiết kỹ thuật của họ.

- **Khái niệm cơ bản.**

Trước tiên, hãy xem xét một chữ ký ví dụ:

```
signature my-first-sig {
    ip-proto == tcp
    dst-port == 80
    payload /.*root/
    event "Found root!"
}
```

Hình ảnh 12. Ví dụ về chữ ký.

- Chữ ký này yêu cầu Zeek so khớp biểu thức chính quy .*root trên tất cả các kết nối TCP đi đến cổng 80. Khi chữ ký kích hoạt, Zeek sẽ đưa ra một sự kiện signature_match có dạng:

```
event signature_match(state: signature_state, msg: string, data: string)
```

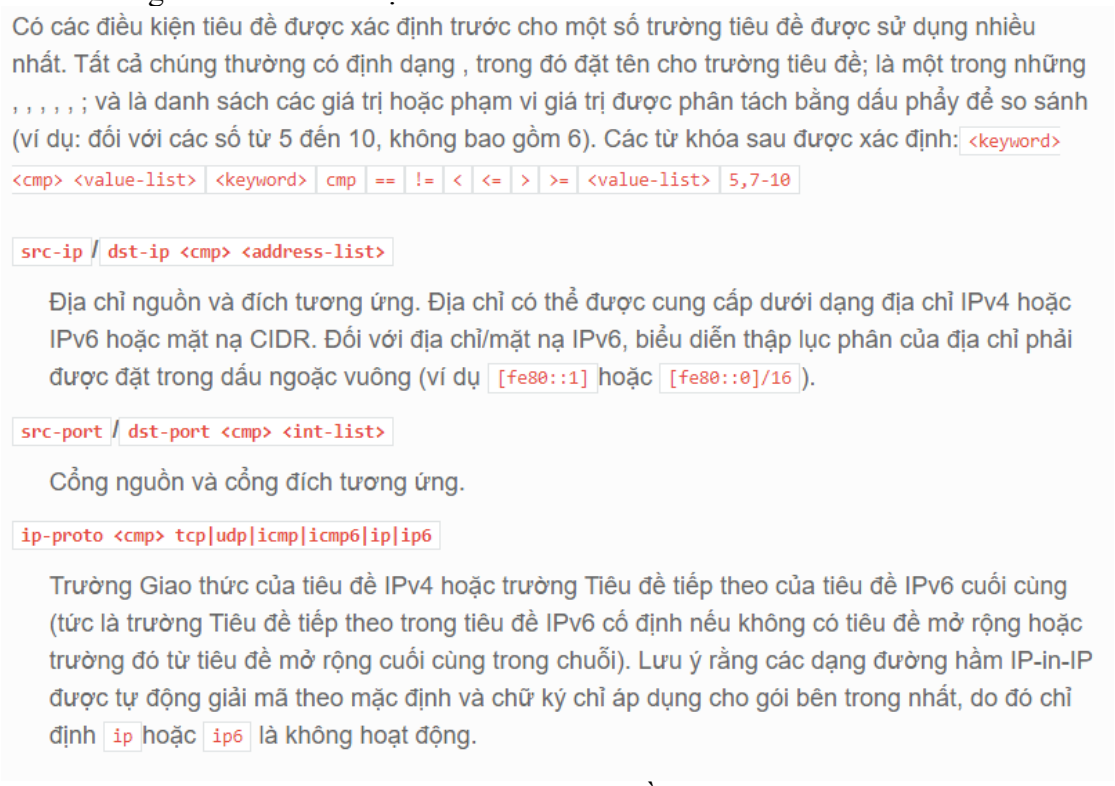
Hình ảnh 13. Event.

- Ở đây, state chứa thêm thông tin về kết nối đã kích hoạt đối sánh, msg là chuỗi được chỉ định bởi câu lệnh sự kiện của chữ ký ( ) và dữ liệu là phần trọng tải cuối cùng đã kích hoạt đối sánh mẫu. Found root! Vì chữ ký độc lập với các tập lệnh của Zeek, chúng được đưa vào (các) tệp của riêng họ. Có ba cách để chỉ định tệp nào chứa chữ ký: Bằng cách sử dụng -s cờ khi bạn gọi Zeek hoặc bằng cách mở rộng biến Zeek signature_files bằng += toán tử hoặc bằng cách sử dụng @load-sigs chỉ thị bên trong tập lệnh Zeek. Nếu một tệp chữ ký được cung cấp mà không có đường dẫn đầy đủ, nó sẽ được tìm kiếm theo cách bình thường ZEEKPATH . Ngoài ra, @load-sigs chỉ thị có thể được sử dụng để tải các tệp chữ ký trong một đường dẫn liên quan đến tập lệnh Zeek mà nó được đặt, ví dụ: sẽ mong đợi tệp chữ ký đó trong cùng thư mục với tập lệnh Zeek. Phần mở rộng mặc định của tên tệp là và Zeek sẽ tự động thêm phần mở rộng đó khi cần thiết. @load-sigs ./mysigs.sig.sig.

- Ngôn ngữ Chữ ký cho Lưu lượng Mạng:
  - Chúng ta hãy xem xét định dạng của một chữ ký kỹ hơn. Mỗi chữ ký cá nhân có định dạng, đâu là nhãn duy nhất cho chữ ký. Có hai loại thuộc tính: điều kiện và hành động. Các điều kiện xác định thời điểm chữ ký khớp, trong khi các hành động khai báo những việc cần làm trong trường hợp khớp. Các điều kiện có thể được chia thành bốn loại: tiêu đề, nội dung , phụ thuộc và ngữ cảnh.

Các điều kiện

- Điều kiện tiêu đề giới hạn khả năng áp dụng của chữ ký cho một tập con lưu lượng có chứa tiêu đề gói phù hợp. Loại đối sánh này chỉ được thực hiện cho gói đầu tiên của một kết nối.


Có các điều kiện tiêu đề được xác định trước cho một số trường tiêu đề được sử dụng nhiều nhất. Tất cả chúng thường có định dạng , trong đó đặt tên cho trường tiêu đề; là một trong những , , , , , ; và là danh sách các giá trị hoặc phạm vi giá trị được phân tách bằng dấu phẩy để so sánh (ví dụ: đối với các số từ 5 đến 10, không bao gồm 6). Các từ khóa sau được xác định: <keyword>

<cmp> <value-list> | <keyword> | cmp | == | != | < | <= | > | >= | <value-list> | 5,7-10

src-ip / dst-ip <cmp> <address-list>

Địa chỉ nguồn và đích tương ứng. Địa chỉ có thể được cung cấp dưới dạng địa chỉ IPv4 hoặc IPv6 hoặc mặt nạ CIDR. Đối với địa chỉ/mặt nạ IPv6, biểu diễn thập lục phân của địa chỉ phải được đặt trong dấu ngoặc vuông (ví dụ [fe80::1] hoặc [fe80::0]/16).

src-port / dst-port <cmp> <int-list>

Cổng nguồn và cổng đích tương ứng.

ip-proto <cmp> tcp|udp|icmp|icmp6|ip|ip6

Trường Giao thức của tiêu đề IPv4 hoặc trường Tiêu đề tiếp theo của tiêu đề IPv6 cuối cùng (tức là trường Tiêu đề tiếp theo trong tiêu đề IPv6 cố định nếu không có tiêu đề mở rộng hoặc trường đó từ tiêu đề mở rộng cuối cùng trong chuỗi). Lưu ý rằng các dạng đường hầm IP-in-IP được tự động giải mã theo mặc định và chữ ký chỉ áp dụng cho gói bên trong nhất, do đó chỉ định ip hoặc ip6 là không hoạt động.

Hình ảnh 14. Các điều kiện.

Điều kiện nội dung

- Điều kiện nội dung được xác định bởi biểu thức chính quy. Chúng tôi phân biệt hai loại điều kiện nội dung: thứ nhất, biểu thức có thể được khai báo bằng câu payloadlệnh, trong trường hợp đó nó được khớp với tải trọng thô của một kết nối (đối với các luồng TCP được tập hợp lại) hoặc của mỗi gói (đối với ICMP, UDP và không phải -TCP được tập hợp lại). Thứ hai, nó có thể được đặt trước một nhãn dành riêng cho máy phân tích, trong trường hợp đó biểu thức được so khớp với dữ liệu được trích xuất bởi máy phân tích tương ứng.


payload /<regular expression>/

Hiện tại, các điều kiện nội dung dành riêng cho máy phân tích sau đây đã được xác định (lưu ý rằng máy phân tích tương ứng phải được kích hoạt bằng cách tải tập lệnh chính sách của nó):

http-request /<regular expression>/

Biểu thức chính quy được so khớp với các URI đã giải mã của các yêu cầu HTTP. Bí danh lỗi thời: http .

http-request-header /<regular expression>/

Biểu thức chính quy được so khớp với các tiêu đề HTTP phía máy khách.

http-request-body /<regular expression>/

Biểu thức chính quy được so khớp với nội dung yêu cầu HTTP phía máy khách.

http-reply-header /<regular expression>/

Biểu thức chính quy được so khớp với tiêu đề HTTP phía máy chủ.

http-reply-body /<regular expression>/

Biểu thức chính quy được so khớp với nội dung phản hồi HTTP phía máy chủ.

ftp /<regular expression>/

Biểu thức chính quy được khớp với đầu vào dòng lệnh của phiên FTP.

finger /<regular expression>/

Hình ảnh 15. Các điều kiện nội dung.

Điều kiện phụ thuộc

- Để xác định sự phụ thuộc giữa các chữ ký, có hai điều kiện:

`requires-signature [!] <id>`

Xác định chữ ký hiện tại chỉ khớp nếu chữ ký được cung cấp bởi `id` khớp với cùng một kết nối. Việc sử dụng `!` phủ định điều kiện: Chữ ký hiện tại chỉ khớp nếu `id` không khớp với cùng một kết nối (sử dụng điều này sẽ trì hoãn quyết định khớp cho đến khi kết nối chấm dứt).

`requires-reverse-signature [!] <id>`

Tương tự như `requires-signature`, nhưng `id` phải khớp với hướng ngược lại của cùng một kết nối, so với chữ ký hiện tại. Điều này cho phép mô hình hóa khái niệm yêu cầu và phản hồi.

Hình ảnh 16. Các điều kiện phụ thuộc.

Điều kiện bối cảnh

- Các điều kiện bối cảnh chuyển quyết định trận đấu sang các thành phần khác của Zeek. Chúng chỉ được đánh giá nếu tất cả các điều kiện khác đã phù hợp. Các điều kiện ngữ cảnh sau đây được xác định:

`eval <policy-function>`

Hàm chính sách đã cho được gọi và phải trả về một boolean xác nhận kết quả khớp. Nếu trả về sai, sẽ không có kết quả khớp chữ ký nào được kích hoạt. Hàm phải thuộc loại . Ở đây, có thể chứa đoạn nội dung gần đây nhất có sẵn tại thời điểm chữ ký được khớp. Nếu không có đoạn nào như vậy thì sẽ là chuỗi trống. Xem định nghĩa của nó. `function cond(state: signature_state, data: string): bool` `data` `data` `signature_state`

`payload-size <cmp> <integer>`

So sánh số nguyên với kích thước tải trọng của gói. Đối với các luồng TCP được tập hợp lại, số nguyên được so sánh với kích thước của đoạn tải trọng theo thứ tự đầu tiên. Lưu ý rằng cái sau không được xác định rõ ràng.

`same-ip`

Được đánh giá là đúng nếu địa chỉ nguồn của gói IP bằng địa chỉ đích của nó.

`tcp-state <state-list>`

Áp đặt các hạn chế đối với trạng thái TCP hiện tại của kết nối. `state-list` là danh sách các từ khóa được phân tách bằng dấu phẩy `established` (bắt tay ba chiều đã được thực hiện), `originator` (dữ liệu hiện tại được gửi bởi người khởi tạo kết nối) và `responder` (dữ liệu hiện tại được gửi bởi người phản hồi kết nối).

`udp-state <state-list>`

Áp đặt các hạn chế về hướng luồng UDP phù hợp. `state-list` là danh sách được phân tách bằng dấu phẩy của một trong hai `originator` (dữ liệu hiện tại được gửi bởi người khởi tạo kết nối) hoặc `responder` (dữ liệu hiện tại được gửi bởi người phản hồi kết nối). Trạng `established` thái bị từ chối do lỗi trong chữ ký vì nó không có ý nghĩa hữu ích như đối với TCP.

Hình ảnh 17. Các điều kiện bối cảnh.

Actions

- Hành động xác định phải làm gì nếu chữ ký khớp. Hiện tại, có hai hành động được xác định event và enable.
- Các thao tác xác định việc cần làm nếu một chữ ký khớp. Hiện tại, có hai hành động được xác định:

+ Event <string>: Nâng cao một signature_match sự kiện. Trình xử lý sự kiện có loại sau: event signature_match(state: signature_state, msg: string, data: string) Chuỗi đã cho được chuyển vào dưới dạng msg và dữ liệu là phần hiện tại của trọng tải cuối cùng dẫn đến khớp chữ ký (điều này có thể trống đối với chữ ký không có điều kiện nội dung).
+ Enable <string>: Bật trình phân tích giao thức cho kết nối phù hợp ("http" , "ftp" v.v.). Điều này được sử dụng bởi tính năng phát hiện giao thức động của Zeek để kích hoạt các bộ phân tích khi đang di chuyển.

- **Ngôn ngữ chữ ký cho nội dung tệp.**

Khung chữ ký cũng có thể được sử dụng để xác định các loại tệp MIME bất kể giao thức/kết nối mạng mà tệp được truyền qua đó. Một loại chữ ký đặc biệt có thể được viết cho mục đích này và sẽ được sử dụng tự động bởi Files Framework hoặc bởi các tập lệnh Zeek sử dụng file_magic chức năng tích hợp sẵn.

Điều kiện

- Chữ ký tệp sử dụng một loại điều kiện nội dung duy nhất ở dạng biểu thức chính quy:
- file-magic /<regular expression>/. Điều này tương tự với payloadđiều kiện nội dung cho ngôn ngữ chữ ký lưu lượng mạng được mô tả ở trên. Sự khác biệt là payloadchữ ký được áp dụng cho tải trọng của các kết nối mạng, nhưng file-magic có thể được áp dụng cho bất kỳ dữ liệu tùy ý nào, nó không nhất thiết phải gắn với giao thức/kết nối mạng.

Actions

- Khi khớp một đoạn dữ liệu, chữ ký tệp sẽ sử dụng hành động sau để lấy thông tin về loại MIME của dữ liệu đó: file-mime <string> [, <integer>].
- Các đối số bao gồm chuỗi loại MIME được liên kết với biểu thức chính quy ma thuật của tệp và "độ mạnh" tùy chọn dưới dạng số nguyên có dấu. Vì nhiều chữ ký ma thuật tệp có thể khớp với một đoạn dữ liệu nhất định nên giá trị cường độ có thể được sử dụng để giúp chọn "người chiến thắng". Giá trị cao hơn được coi là mạnh hơn.

d, Input Framework.

Zeek có khung nhập liệu linh hoạt cho phép người dùng nhập dữ liệu tùy ý vào Zeek. Dữ liệu được đọc vào các bảng Zeek hoặc được chuyển đổi trực tiếp thành các sự kiện để các tập lệnh xử lý khi chúng thấy phù hợp. Kiến trúc trình đọc mô-đun cho phép đọc từ tệp, cơ sở dữ liệu hoặc các nguồn dữ liệu khác.

- **Đọc dữ liệu thành bảng:**
  - Tệp đầu vào được đọc vào bảng với lệnh gọi hàm:Input::add_table

```
global denylist: table[addr] of Val = table();

event zeek_init() {
    Input::add_table([$source="denylist.file", $name="denylist",
                      $idx=Idx, $val=Val, $destination=denylist]);
    Input::remove("denylist");
}
```

Hình ảnh. 18. Tệp đầu vào được đọc vào bảng với lệnh gọi hàm:Input::add_table.

- Với ba dòng này, trước tiên, chúng ta tạo một bảng trống sẽ nhận dữ liệu danh sách từ chối và sau đó hướng dẫn khung công tác đầu vào mở một luồng đầu vào có tên "danh sách từ chối" để đọc dữ liệu vào bảng. Dòng thứ ba lại loại bỏ luồng đầu vào, vì chúng ta không cần nó nữa sau khi dữ liệu đã được đọc.

Xử lý không đồng bộ

- Vì một số tệp dữ liệu có thể khá lớn, khung đầu vào hoạt động không đồng bộ. Một luồng mới được tạo cho mỗi luồng đầu vào mới. Luồng này mở tệp dữ liệu đầu vào, chuyển đổi dữ liệu thành định dạng bên trong và gửi nó trở lại luồng Zeek chính. Do đó, dữ liệu không thể truy cập ngay lập tức. Tùy thuộc vào kích thước của nguồn dữ liệu, có thể mất từ vài mili giây đến vài giây cho đến khi tất cả dữ liệu có trong bảng.
- Các lệnh gọi tiếp theo đến một nguồn đầu vào được xếp hàng đợi cho đến khi hoàn thành hành động trước đó. Vì điều này, ví dụ, nó có thể được gọi Input::add_table và Input::remove trong hai dòng tiếp theo: hành động loại bỏ sẽ vẫn được xếp hàng đợi cho đến khi hoàn thành lần đọc đầu tiên.
- Khi khung công tác đầu vào đọc xong từ một nguồn dữ liệu, nó sẽ kích hoạt Input::end_of_data sự kiện. Khi sự kiện này đã được nhận, tất cả dữ liệu từ tệp đầu vào sẽ có sẵn trong bảng.

```
event Input::end_of_data(name: string, source: string) {
    # now all data is in the table
    print denylist;
}
```

```
if ( 192.168.18.12 in denylist )
    # take action
```

Hình ảnh 19. Đầu vào đọc từ một nguồn dữ liệu.

Bộ thay vì bảng

- Đối với một số trường hợp sử dụng, khái niệm khóa / giá trị thúc đẩy dữ liệu dạng bảng không áp dụng, chẳng hạn như khi mục đích chính của dữ liệu là để kiểm tra tư cách thành viên trong một tập hợp. Khung đầu vào hỗ trợ phương pháp này bằng cách sử dụng các tập hợp làm kiểu dữ liệu đích và bỏ $val qua Input::add_table :

```
type Idx: record {
    ip: addr;
};

global denylist: set[addr] = set();

event zeek_init() {
    Input::add_table([$source="denylist.file", $name="denylist",
                      $idx=Idx, $destination=denylist]);
    Input::remove("denylist");
}
```

Hình ảnh 20. Tập dữ liệu đích.

Đọc lại và phát trực tuyến dữ liệu

- Đối với một số nguồn dữ liệu (chẳng hạn như nhiều danh sách từ chối), dữ liệu đầu vào thay đổi liên tục. Khung đầu vào hỗ trợ các kỹ thuật bổ sung để quản lý đầu vào luôn thay đổi như vậy.
- Phương pháp đầu tiên, rất cơ bản là làm mới một cách rõ ràng một luồng đầu vào. Khi một luồng đầu vào đang mở (nghĩa là nó vẫn chưa bị loại bỏ bởi một lệnh gọi đến Input::remove), hàm Input::force_update có thể được gọi. Điều này sẽ kích hoạt làm mới hoàn toàn bảng: mọi phần tử đã thay đổi từ tệp sẽ được cập nhật, những phần tử mới được thêm vào và bất kỳ phần tử nào không còn trong dữ liệu đầu vào sẽ bị xóa. Sau khi cập nhật xong, Input::end_of_data sự kiện sẽ được nâng lên.

Nhận các sự kiện thay đổi

- Khi đọc lại tệp, có thể thú vị khi biết chính xác dòng nào trong tệp nguồn đã thay đổi. Vì lý do này, khuôn khổ đầu vào có thể đưa ra một sự kiện mỗi khi một mục dữ liệu được thêm vào, xóa khỏi hoặc thay đổi trong bảng.

Lọc dữ liệu trong quá trình nhập

- Khung nhập liệu cũng cho phép người dùng lọc dữ liệu trong quá trình nhập. Để đạt được mục đích này, các chức năng vị ngữ được sử dụng. Một hàm vị từ được gọi trước khi một phần tử mới được thêm / thay đổi / xóa khỏi bảng. Vị từ có thể chấp nhận hoặc phủ quyết thay đổi bằng cách trả về true cho một thay đổi được chấp nhận và false cho một thay đổi bị từ chối. Hơn nữa, nó có thể thay đổi dữ liệu trước khi nó được ghi vào bảng.

Đọc dữ liệu cho sự kiện

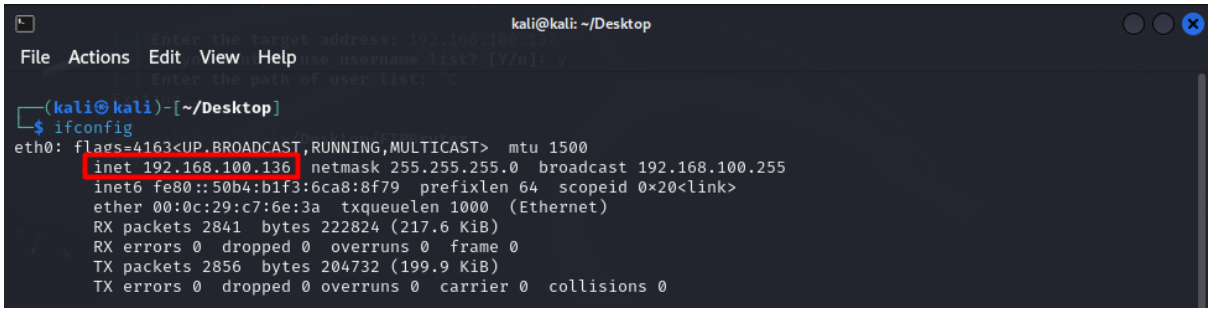
- Chế độ nhập dữ liệu thứ hai của khung nhập liệu trực tiếp tạo ra các sự kiện Zeek từ dữ liệu đã nhập thay vì chèn chúng vào bảng. Luồng sự kiện hoạt động rất giống với các luồng bảng được thảo luận ở trên và hầu hết các tính năng được thảo luận (chẳng hạn như các vị từ để lọc) cũng hoạt động cho các luồng sự kiện.
- Luồng sự kiện khác với luồng bảng theo hai cách:
  + Luồng sự kiện không cần khai báo chỉ mục và giá trị riêng biệt - thay vào đó, tất cả các kiểu dữ liệu nguồn được cung cấp trong một định nghĩa bản ghi duy nhất.
  + Vì khung công tác nhận thức một dòng sự kiện liên tục, nên nó không có khái niệm về đường cơ sở dữ liệu (ví dụ: một bảng) để so sánh dữ liệu đến. Do đó, loại sự kiện thay đổi (một Input::Event trường hợp, tpe ở trên) hiện luôn luôn Input::EVENT_NEW .

Người đọc dữ liệu

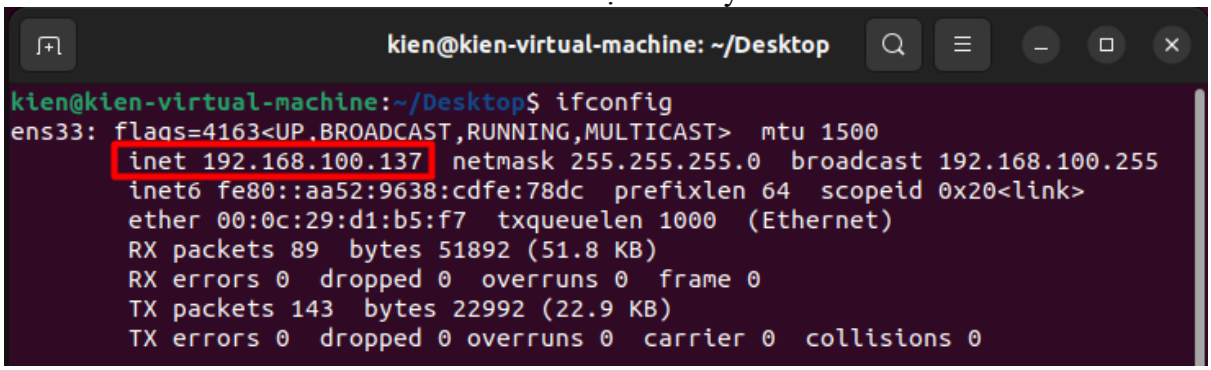
- Khung đầu vào hỗ trợ các loại trình đọc khác nhau cho các loại tệp dữ liệu nguồn khác nhau. Hiện tại, khuôn khổ mặc định nhập các tệp ASCII được định dạng ở định dạng tệp nhật ký Zeek (các giá trị được phân tách bằng tab bằng #fields dòng tiêu đề). Một số trình đọc khác được bao gồm trong Zeek và các gói / plugin của Zeek có thể cung cấp các trình đọc bổ sung.

IV. Demo.

- Kịch bản sử dụng máy kali để tấn công brute force dịch vụ ftp tại cổng 21. Sử dụng Zeek Scripting để phát hiện Brute-force.

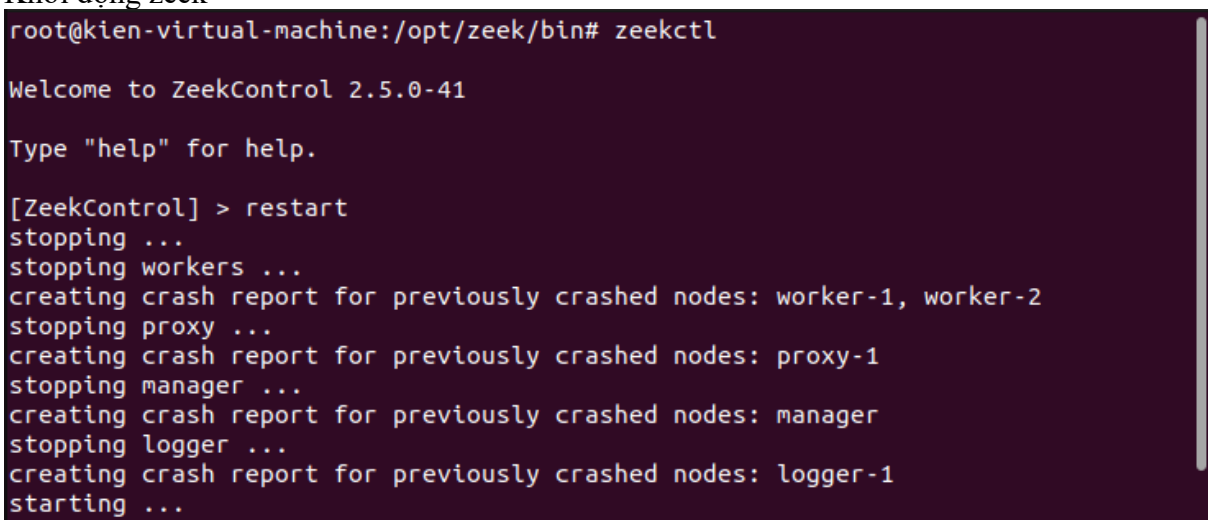


Hình ảnh 21. Địa chỉ máy Kali

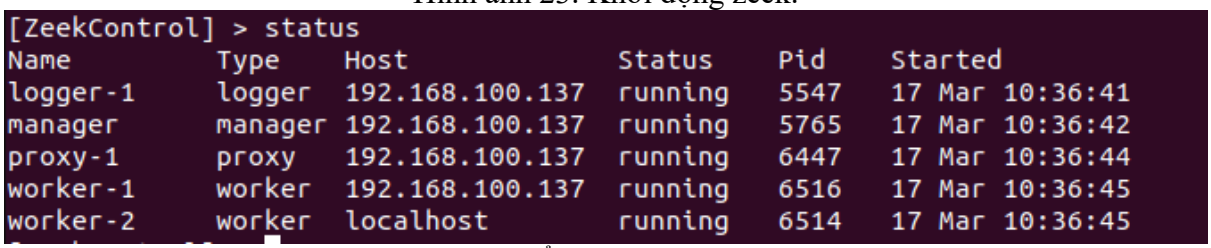


Hình ảnh 22. Địa chỉ máy Ubuntu.

Khởi động zeek



Hình ảnh 23. Khởi động zeek.

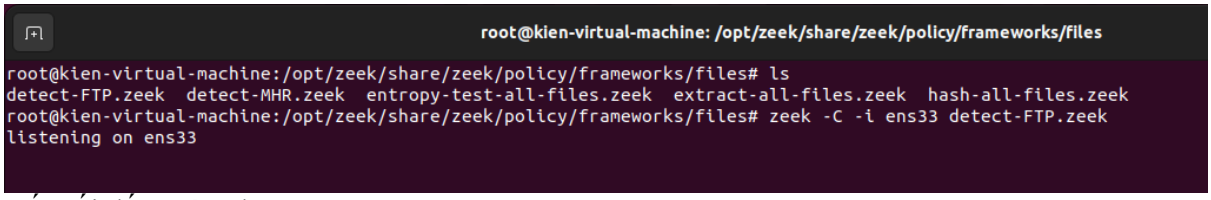


Hình ảnh 24. Kiểm tra trạng thái của zeek.

Vào đường dẫn /opt/zeek/share/zeek/policy/frameworks/files chạy file detect-FTP.zeek bằng câu lệnh:

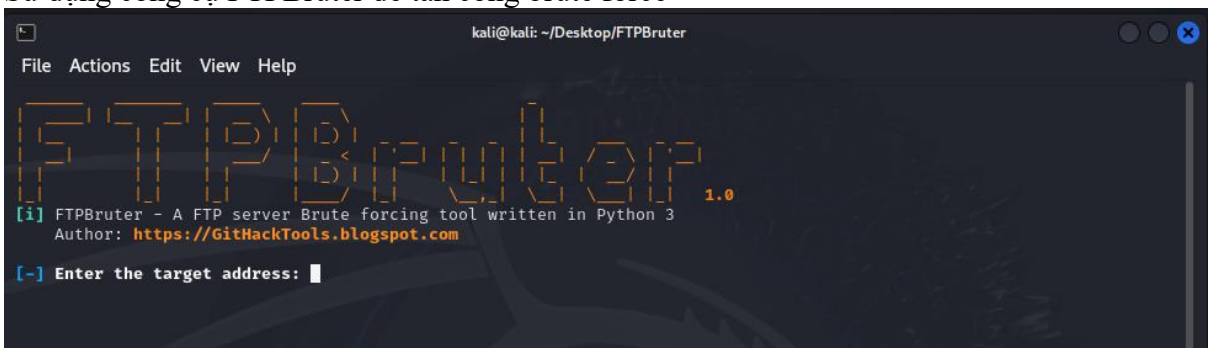
zeek -C -i ens33 detect-FTP.zeek

câu lệnh "zeek -C -i ens33 detect-FTP.zeek" sẽ chạy Zeek trên giao diện mạng "ens33", cho phép Zeek phân tích các gói tin có checksum không hợp lệ và sử dụng tệp script "detect-FTP.zeek" để phát hiện các hoạt động FTP trong gói tin mạng.



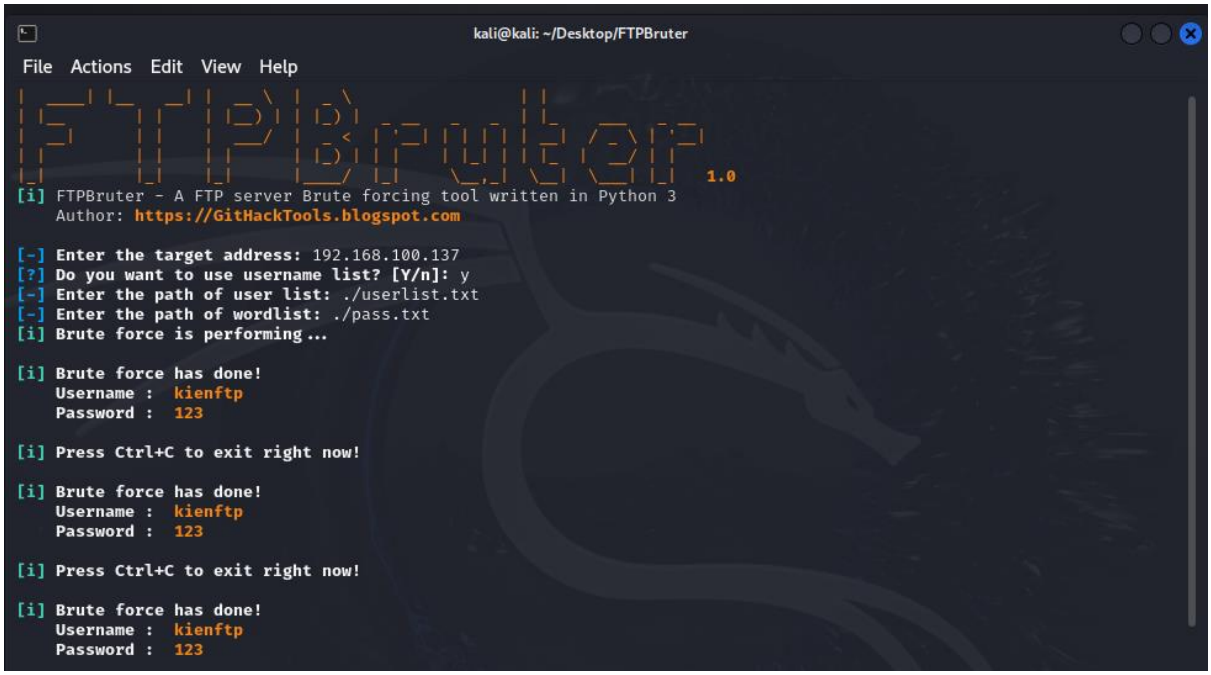
Kết nối đến máy ubuntu

Sử dụng công cụ FTPBruter để tấn công brute force

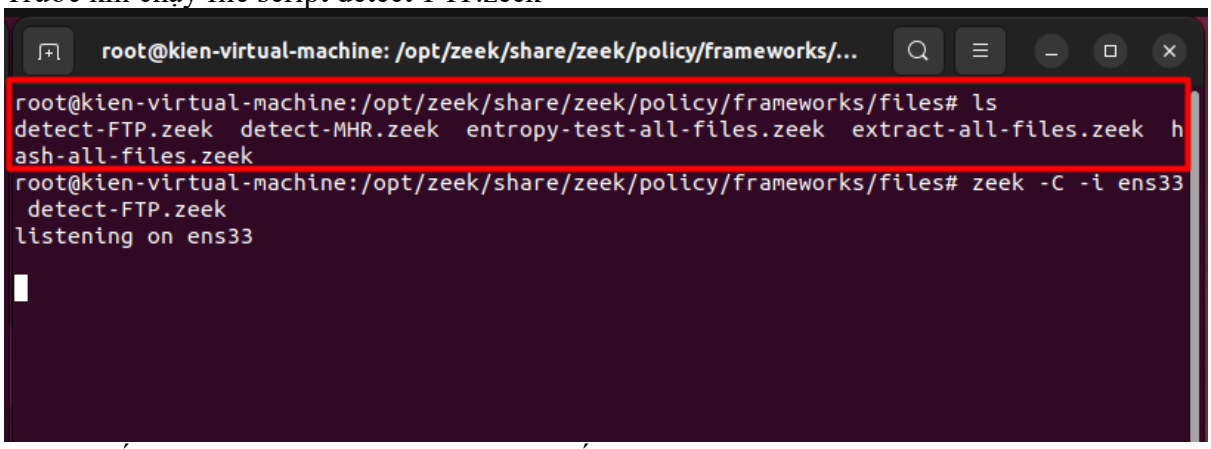


Địa chỉ Ip mục tiêu là địa chỉ Ip máy ubuntu

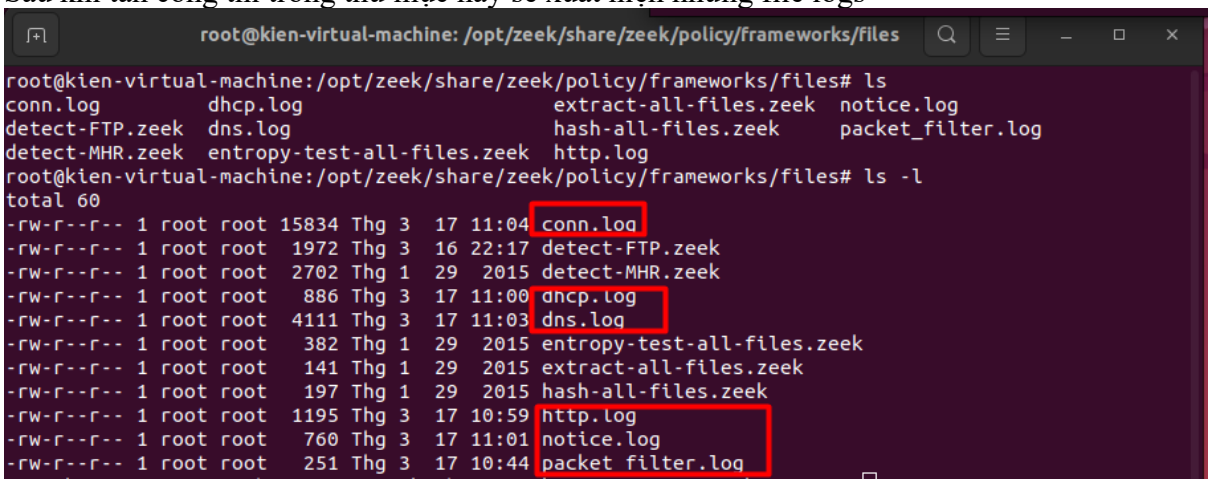
Khai thác thành công



Truy cập thư mục chứa file detect-FTP.zeek:
/opt/zeek/share/zeek/policy/frameworks/files
Trước khi chạy file script detect-FTP.zeek

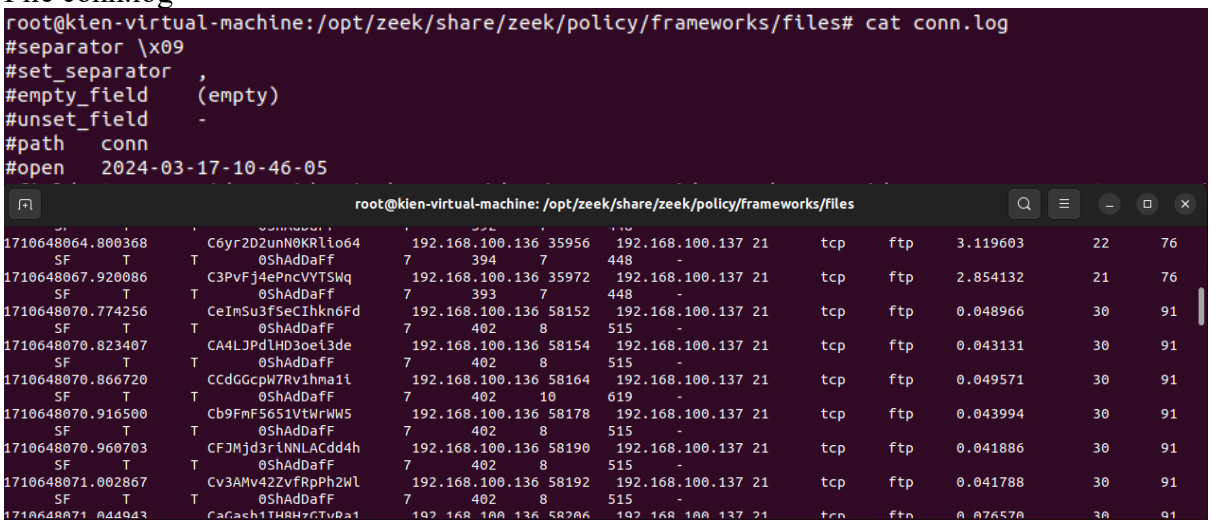


Sau khi tấn công thì trong thư mục này sẽ xuất hiện những file logs

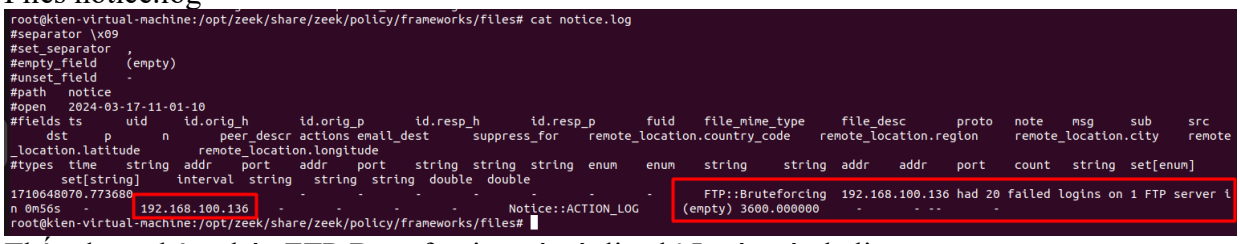


Ta xem các file logs:

File conn.log



Files notice.log



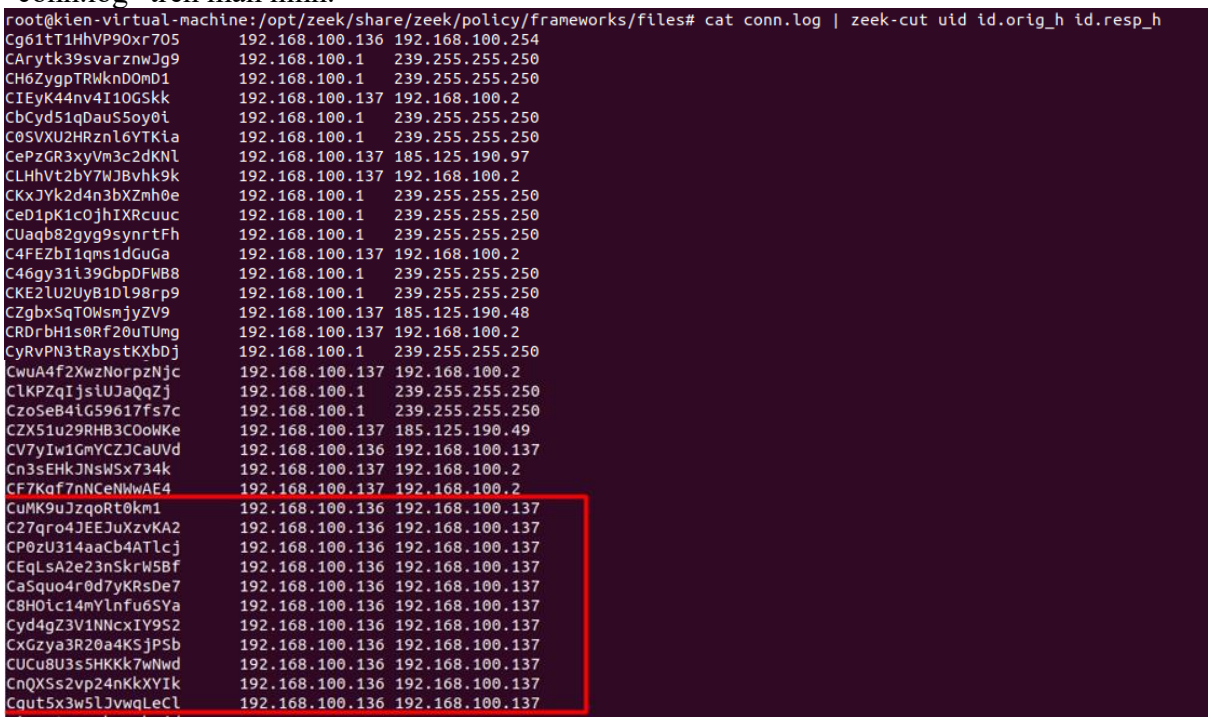
Thấy được thông báo FTP:Bruteforcing và có địa chỉ Ip từ máy kali

→ Zeek đã phát hiện được xâm nhập bất thường từ máy kali

Sử dụng câu lệnh: **cat conn.log | zeek-cut uid id.orig_h id.resp_h**

("zeek-cut": Đây là một công cụ dòng lệnh Zeek để cắt (cắt giảm) các trường dữ liệu từ đầu ra của Zeek logs.)

Câu lệnh **"cat conn.log | zeek-cut uid id.orig_h(địa chỉ Ip nguồn) id.resp_h(địa chỉ ip đích)"** sẽ hiển thị các giá trị của các trường "uid", "id.orig_h" và "id.resp_h" từ tệp tin "conn.log" trên màn hình.



File Script: detect-FTP.zeek

```
@load base/protocols/ftp
@load base/frameworks/sumstats

@load base/utils/time

module FTP;

export {
        redef enum Notice::Type += {
                ## Indicates a host bruteforcing FTP logins by watching for too
                ## many rejected usernames or failed passwords.
                Bruteforcing
        };

        ## How many rejected usernames or passwords are required before being
        ## considered to be bruteforcing.
        const bruteforce_threshold: double = 20 &redef;

        ## The time period in which the threshold needs to be crossed before
        ## being reset.
        const bruteforce_measurement_interval = 15mins &redef;
}

event zeek_init()
        {
        local r1: SumStats::Reducer = [$stream="ftp.failed_auth", $apply=set(SumStats::UNIQUE), $unique_max=double_to_count(bruteforce_threshold+2)];
        SumStats::create([$name="ftp-detect-bruteforcing",
                          $epoch=bruteforce_measurement_interval,
                          $reducers=set(r1),
                          $threshold_val(key: SumStats::Key, result: SumStats::Result) =
                                {
                                return result["ftp.failed_auth"]$num+0.0;
                                },
                          $threshold=bruteforce_threshold,
                          $threshold_crossed(key: SumStats::Key, result: SumStats::Result) =
                                {
                                local r = result["ftp.failed_auth"];
                                local dur = duration_to_mins_secs(r$end-r$begin);
                                local plural = r$unique>1 ? "s" : "";
                                local message = fmt("%s had %d failed logins on %d FTP server%s in %s", key$host, r$num, r$unique, plural, dur);
                                NOTICE([$note=FTP::Bruteforcing,
                                        $src=key$host,
                                        $msg=message,
                                        $identifier=cat(key$host)]);
                                }]);
        }

 event ftp_reply(c: connection, code: count, msg: string, cont_resp: bool)
        {
        local cmd = c$ftp$cmdarg$cmd;
        if ( cmd == "USER" || cmd == "PASS" )
                {
                if ( FTP::parse_ftp_reply_code(code)$x == 5 )
                        SumStats::observe("ftp.failed_auth", [$host=c$id$orig_h], [$str=cat(c$id$resp_h)]);
                }
        }
```

Tài liệu tham khảo

zeek. (n.d.). *Bro (Zeek) Network Monitoring Project*. Retrieved from https://github.com/zeek/zeek.

Zeek. (n.d.). *Try Zeek*. Retrieved from https://try.zeek.org/.

Zeek. (n.d.). *Zeek Network Monitoring*. Retrieved from https://docs.zeek.org/en/master/.

Zeek. (n.d.). *Zeek Network Mornitoring Project*. Retrieved from https://github.com/bro/bro.