Kỷ yếu Hội nghị Quốc gia lần thứ X về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR), Đà Nẵng, ngày 17-18/08/2017
DOI: 10.15625/vap.2017.00078

NGHIÊN CỨU PHƯƠNG PHÁP HÌNH THỨC DỰA TRÊN LOGIC TRONG PHÂN TÍCH AN NINH MẠNG

Nguyễn Thị Ánh Phượng, Nguyễn Trường Thắng, Trần Mạnh Đông, Bùi Thị Thư
Viện Công nghệ thông tin, Viện Hàn lâm Khoa học và Công nghệ Việt Nam
{ ntaphuong, ntthang, dongtm, btthu }@ioit.ac.vn

***TÓM TẮT**: Ngày nay với sự phát triển không ngừng của internet và các lĩnh vực trong công nghệ thông tin, các thiết bị phần cứng, các ứng dụng phần mềm và các dịch vụ trên internet trở nên rất phổ biến làm cho nền kinh tế của các nước trên toàn thế giới ngày càng phát triển.*

Tuy nhiên, cùng với sự phát triển nhanh chóng đó luôn tiềm ẩn những rủi ro, gây ra những hệ lụy tổn thất về tài chính, uy tín… cho các tổ chức hay cá nhân tham gia dịch vụ. Do đó, vấn đề đảm bảo an toàn, an ninh thông tin trong quá trình truyền tải thông tin trên môi trường mạng luôn là một vấn đề nóng trong nghiên cứu và ứng dụng thực tiễn. Đặc biệt là với một mạng doanh nghiệp, vấn đề đảm bảo an ninh, an toàn thông tin mạng ngày càng trở nên khó khăn. Hiện nay với sự lan tỏa của các lỗ hổng phần mềm, các nhà quản trị hệ thống đang phải đối mặt với rất nhiều thách thức và khó khăn trong việc đảm bảo an ninh không gian mạng. Bài báo này sẽ trình bày một cách tiếp cận của phương pháp hình thức dựa trên logic trong việc phân tích an ninh mạng nhằm phát hiện ra các lỗ hổng đang tồn tại trong hệ thống và những cuộc tấn công có thể có nhằm vào hệ thống, giúp cho các chuyên gia bảo mật và các nhà quản trị hệ thống quản lý tốt hơn cấu hình của một mạng doanh nghiệp và có thể kiểm soát được các rủi ro trong an ninh mạng.

***Từ khóa**: Enterprise network security, logic-programming, attack graphs.*

I. GIỚI THIỆU

Lĩnh vực đảm bảo an toàn thông tin trên môi trường mạng cũng như việc tăng cường khả năng phòng ngừa/khắc phục của hệ thống trước các cuộc tấn công trên mạng đang thu hút sự nghiên cứu trên thế giới, ở cả giới học thuật và giới công nghiệp cũng như sự đầu tư của các chính phủ. Tuy nhiên, một thực tế là cách tiếp cận hiện nay của những chuyên gia quản trị mạng thường phi hình thức, mang tính chất tạm thời, xử lý tình huống, chắp vá. Dựa trên các cảnh báo lỗ hổng từ nhiều nguồn như CERT, BugTraq,... họ tự đánh giá khả năng ảnh hưởng những lỗ hổng này trong hệ thống của mình và tự xây dựng cách xử lý: vá lỗi và khởi động lại hệ thống, tái thiết lập lại cấu hình tường lửa (firewall),... [1]. Cách tiếp cận này thường phụ thuộc trình độ/kinh nghiệm của từng cá nhân quản trị nên tiềm ẩn nhiều rủi ro. Với sự trưởng thành về cơ sở khoa học của các phương pháp hình thức đồng thời sự phát triển vượt trội về tốc độ tính toán, việc ứng dụng các phương pháp hình thức vào kiểm chứng sự an toàn của các giao thức truyền tin trên mạng (kể cả internet, mạng viễn thông cố định và không dây như IEEE 802.x [2]) cũng như sự suy diễn các tổn thương của hệ thống do các lỗ hổng tiềm năng phát hiện được từ lỗi mã nguồn [3], [4] hoặc lỗi thiết lập cấu hình [1],... đã trở thành hướng nghiên cứu chủ đạo trong lĩnh vực an ninh mạng trên thế giới trong khoảng 10 năm trở lại đây.

Tại Việt Nam, an toàn và an ninh thông tin cũng đang là một vấn đề nóng và ngày càng trở nên cấp thiết nhất là trong bối cảnh liên tiếp xảy ra các vụ tấn công mạng như sự kiện tin tặc tấn công cổng thông tin ngành hàng không Việt Nam (Vietnam Airlines) và hệ thống thông tin hiển thị bay tại 2 cụm cảng hàng không Nội Bài và Tân Sơn Nhất vào ngày 29/7/2016. Có thể nói cuộc tấn công website và hệ thống thông tin sân bay này được đánh giá là lớn nhất từ trước đến nay vào hệ thống thông tin hàng không của Việt Nam.

Việc đối phó với các lỗ hổng phần mềm trên các máy chủ mạng đặt ra một thách thức lớn cho các nhà quản trị mạng nhất là khi số lượng các máy chủ mạng tiếp tục phát triển cả về quy mô lẫn độ phức tạp. Do vậy việc lựa chọn các biện pháp đối phó với các lỗ hổng đó là rất quan trọng. Hiện nay đã có một số công cụ có thể tự động quét, phát hiện và báo cáo các lỗ hổng tồn tại trên các máy chủ riêng biệt như: COPS [5], Nessus [6], OVAL [7]. Tuy nhiên cần phải phân tích và đánh giá những ảnh hưởng và sự tương tác của các lỗ hổng này đến vấn đề an ninh an toàn mạng như thế nào. Do vậy việc nghiên cứu sử dụng phương pháp nào cho hiệu quả và có thể áp dụng được trên thực thế là rất quan trọng.

Trong đề tài này chúng tôi đề xuất lựa chọn phương pháp hình thức dựa trên logic. Sử dụng Datalog, một tập hợp con các cú pháp của Prolog để tìm ra phần lớn các cuộc tấn công có thể xảy ra trong một mạng. So với các đồ thị khai thác phụ thuộc (*exploit-dependency graph*), Datalog là một ngôn ngữ logic khai báo hình thức, trong đó cung cấp một đặc tả kỹ thuật rõ ràng. Thời gian thực hiện của một chương trình Datalog là thời gian đa thức phụ thuộc vào kích thước của dữ liệu đầu vào. Các công cụ logic đã được tối ưu hóa trong nhiều thập kỷ để xử lý các tập dữ liệu lớn một cách hiệu quả, vì vậy Datalog đặc biệt thích hợp cho việc phân tích an ninh của các mạng lớn và phức tạp.

Trong phần tiếp theo sẽ trình bày một số kiến thức nền tảng về an ninh mạng và các luật suy diễn Datalog cơ bản dùng trong phân tích an ninh mạng. Phần III sẽ đi vào phân tích cơ sở dữ liệu, đặc tả hình thức của các thông tin cấu hình mạng, máy chủ, lỗ hổng,… Phần IV trình bày thuật toán sinh đồ thị tấn công. Phần V là thực nghiệm và cuối cùng phần VI là phần kết luận.

II. MỘT SỐ KIẾN THỨC NỀN TẢNG

A. Lỗ hổng bảo mật và vấn đề quản lý an ninh mạng

Các lỗ hổng bảo mật trên một hệ thống là các điểm yếu có thể tạo ra sự ngưng trệ của dịch vụ, thêm quyền đối với người sử dụng hoặc cho phép các truy nhập không hợp pháp vào hệ thống. Các lỗ hổng cũng có thể nằm ngay các dịch vụ cung cấp như sendmail, web, ftp,... Ngoài ra các lỗ hổng còn tồn tại ngay chính tại hệ điều hành như trong Windows NT, Windows 95, UNIX; hoặc trong các ứng dụng mà người sử dụng thường xuyên sử dụng như Word processing, Các hệ databases,…

Lỗi logic (Logic error): là loại lỗ hổng thường sinh ra khi người lập trình thực hiện thiết kế các chương trình ứng dụng. Đây là loại lỗ hổng thường được mọi người quan tâm và nghĩ đến đầu tiên. Nó bao gồm các loại lỗ hổng sau:

Lỗ hổng hệ điều hành: Hệ điều hành là phần mềm hệ thống đóng vai trò trung gian trong giao tiếp giữa người dùng và phần cứng máy tính cho phép người sử dụng phát triển và thực hiện các ứng dụng của họ một cách dễ dàng. Với góc độ xử lý ta có các hệ điều hành đơn nhiệm MS-DOS, hệ điều hành đa nhiệm,… Với góc độ máy tính ta có hệ điều hành Window, Linux, MacOS. Nếu một người quản trị kém có thể là nguyên nhân dẫn tới hệ điều hành bị tin tặc tấn công. Nhưng đôi khi chính bản thân hệ điều hành cũng chứa đựng nhiều lỗ hổng, như tràn bộ đệm "buffer overflow".

Lỗ hổng trong các ứng dụng: Khác với lỗ hổng hệ điều hành, để tấn công thông qua các lỗ hổng trong các ứng dụng, người tấn công phải xác định được một cách chính xác máy tính nạn nhân đang sử dụng các chương trình ứng dụng tồn tại lỗ hổng bảo mật đó. Tuy nhiên, lỗ hổng trong các ứng dụng đôi khi còn tùy thuộc vào cách thức cài đặt, cấu hình các ứng dụng đang có của người sử dụng. Các hình thức tấn công phổ biến thông qua lỗ hổng ứng dụng hiện nay là: liên kết các địa chỉ web, khai thác Cookie, làm tràn bộ nhớ đệm, lỗ hổng trong các giao thức.

Lỗ hổng trong chính sách bảo mật (*Policy oversight*)**:** Là loại lỗ hổng sinh ra trong quá trình thiết lập các chính sách bảo mật cho một hệ thống như là thiếu cơ chế backup dữ liệu, dự phòng các sự cố phần cứng, nâng cấp phần mềm,… Hacker có thể lợi dụng những lỗ hổng này để tạo ra chuỗi lỗ hổng mới.

Trong nhiều trường hợp sau khi báo cáo các lỗ hổng sẽ có các bản vá lỗi ngay sau đó, người ta thường sẽ không vội vàng áp dụng ngay các bản vá lỗi này bởi việc vá lỗi vội vàng đó có thể sẽ tạo thêm nhiều lỗi mới hơn và việc áp dụng nó sẽ gây ra nhiều thiệt hại hơn. Do vậy, việc các nhà quản trị mạng vẫn tiếp tục cho chạy phần mềm lỗi đó trong một khoảng thời gian là khá phổ biến [7]. Các nhà quản trị mạng phải đảm bảo tiềm năng khai thác các lỗ hổng đó sẽ không xảy ra và nếu đã xảy ra sẽ không gây bất kỳ thiệt hại nào. Một trong các công việc hàng ngày của quản trị mạng là đọc các báo cáo lỗ hổng từ các nguồn khác nhau và hiểu được thực sự các lỗ hổng đó có ảnh hưởng đến an ninh mạng của họ hay không. Để khám phá ra một cuộc tấn công tiềm ẩn có thể xảy ra trong một mạng, không chỉ kiểm tra các thông số cấu hình trên mỗi phần tử mạng, máy móc, tường lửa, các bộ định tuyến,… mà còn phải xem xét tất cả các tương tác có thể xảy ra giữa chúng. Công việc này gọi là phân tích các lỗ hổng đa máy chủ, đa trạng thái, nếu công việc này chỉ được thực hiện bởi con người thì sẽ tốn rất nhiều thời gian, công sức và rất dễ bị nhầm lẫn sai sót. Do vậy việc tự động hóa công việc này là rất quan trọng và cần thiết. Một công cụ phân tích an ninh mạng tự động sẽ rất khả thi khi mà các mối đe dọa liên tục gia tăng và thay đổi, các suy luận logic cũng cần phải được đặc tả một cách hình thức để vừa có thể thực hiện trên lý thuyết vừa có hiệu quả trên thực tế.

Phần tiếp theo sẽ trình bày chi tiết hơn việc sử dụng đặc tả hình thức để mô hình hóa mạng nhằm phục vụ cho các công cụ phân tích an ninh mạng tự động như MulVAL, một công cụ phân tích an ninh mạng tự động cho một mạng doanh nghiệp sẽ được sử dụng để thực nghiệm trong chương cuối của bài báo.

B. Một số luật Datalog cơ bản

Datalog là một tập con của Prolog [8]. Một sự khác biệt đáng kể giữa Datalog và Prolog là Datalog có ngữ nghĩa tường thuật thuần túy. Thứ tự các mệnh đề trong một chương trình Datalog không liên quan đến ý nghĩa logic và giá trị kết quả của nó. Trong khi đó, trong Prolog lệnh như vậy là rất quan trọng và ảnh hưởng đến kết quả đánh giá [8]. Datalog cũng đã được sử dụng như là một ngôn ngữ bảo mật để thể hiện các chính sách kiểm soát truy cập [9, 10].

Có rất nhiều lợi ích của việc sử dụng Datalog như là một mô hình hóa hình thức của lý luận trong phân tích an ninh mạng đã thảo luận trong bài báo này. So với các đồ thị khai thác phụ thuộc (*exploit-dependency graph*), Datalog là một ngôn ngữ logic khai báo hình thức, trong đó cung cấp một đặc tả kỹ thuật rõ ràng. Giống như trong cách tiếp cận kiểm chứng mô hình, người ta có thể tận dụng một công cụ logic *off-the-shelf* để tiến hành phân tích. Nhưng không giống như kiểm chứng mô hình, thời gian thực hiện của một chương trình Datalog là đa thức theo kích thước của dữ liệu đầu vào. Các công cụ logic đã được tối ưu hóa trong nhiều thập kỷ để xử lý các tập dữ liệu lớn một cách hiệu quả, làm cho Datalog đặc biệt thích hợp cho việc phân tích an ninh của các mạng lớn và phức tạp.

Một *literal*, $p(t_1, ..., t_k)$ là một vị từ, mỗi thành phần trong đó có thể là một hằng số hoặc là một biến. Trong hình thức của Prolog, một biến là một định danh bắt đầu với chữ in hoa, một hằng số sẽ bắt đầu với một chữ cái thường. Đặt $L_0, L_1, …., L_n$ là các *literals*, một mệnh đề Horn trong Datalog có dạng như sau:

L_0 :- $L_1, …., L_n$

Có nghĩa là, nếu L_1, …., L_n là đúng thì L_0 cũng đúng. Phía trái được gọi là phần đầu (*head*), bên phải biểu thức được gọi là phần thân (*body*). Một mệnh đề với một *body* trống được gọi là một *fact*. Một mệnh đề với một *body* không trống được gọi là một *rule*.

Một số luật chỉ sự tồn tại của các lỗ hổng mạng:

• `vulExists(Host,Program,ExploitRange,ExploitConsequence)` là một vị từ có nguồn gốc xác định từ một `Program` trên một `Host` và nó có `ExploitRange` và `ExploitConsequence` riêng biệt. Đây là một vị từ có nguồn gốc. `Program` là đường dẫn đầy đủ của file thực thi có chứa các lỗi bảo mật. `ExploitRange` là `local` hoặc `remote`, chỉ ra lỗi là khai thác tại địa phương hoặc khai thác từ xa. Hai giá trị phổ biến cho `ExploitConsequence` là `privilege Escalation`, có nghĩa là một khai thác thành công sẽ cho phép kẻ tấn công thực thi mã tùy ý, và `dos`, có nghĩa là những kẻ tấn công có thể làm sụp đổ chương trình (từ chối dịch vụ).

• `vulExists(Host,ID,Program)` là một vị từ nguyên thủy chỉ ra rằng đó là một lỗ hổng với định danh `ID` tồn tại trong `Program` trên `Host`.

• `bugHyp(Host,Program,Range,Consequenc)` là một vị từ động giới thiệu một lỗi giả định trong một `Program` trên `Host` mà có `ExploitRange` và `ExploitConsequence`.

Một số luật khai thác lỗ hổng:

• `execCode(P,H,UserPriv)` là một vị từ có nguồn gốc xác định rằng `P` có thể thực thi mã nhị phân với đặc quyền `UserPriv` trên máy `H`.

• `netAccess(P,Src,Dst,Protocol,Port)` là một vị từ có nguồn gốc xác định rằng `P` có thể gửi các gói tin từ máy `Src` đến `Port` trên máy `Dst` thông qua `Protocol`.

• `setuidProgram(H,Prog)` là một vị từ nguyên thủy chỉ ra rằng `Prog` là một `setuid`[1]

III. PHÂN TÍCH CƠ SỞ DỮ LIỆU

A. Đặc tả hình thức cấu hình máy chủ và cấu hình mạng

Một công cụ phân tích an ninh mạng tự động thường bao gồm một máy quét (Scanner) chạy không đồng bộ trên từng máy chủ và thích nghi với các công cụ có sẵn như OVAL hoặc Nessus và một bộ phân tích chạy trên một máy chủ khi có thông tin mới đến từ máy quét.

Cấu hình máy chủ (Host configuration):

Thông tin cấu hình khác nhau trên một máy chủ là rất cần thiết cho các công cụ phân tích an ninh mạng như MulVAL. Các thông tin này có thể dễ dàng nhận được bằng cách thực hiện các câu lệnh hệ điều hành hoặc xem trong các file cấu hình. Một máy quét OVAL / Nessus có thể trích xuất các thông số cấu hình trên một máy chủ. Ví dụ kết quả chuyển đổi đầu ra của máy quét dưới dạng mệnh đề Datalog như sau:

`networkService(webServer,httpd,TCP,80,apache).`

Có nghĩa là chương trình `httpd` chạy trên máy `webServer` sử dụng `apache`, nghe trên cổng `80` và sử dụng giao thức `TCP`.

Cấu hình mạng (**Network configuration**): MulVAL mô hình hóa cấu hình mạng (router và tường lửa) như là một danh sách kiểm soát truy cập máy chủ trừu tượng (host access-control lists - HACL). Đó là một danh sách có cấu trúc như sau:

hacl(Source,Dest,Protocol,Port).

Có nghĩa là một máy trong `Source` có thể đạt đến đích `Dest` thông qua một dịch vụ mạng được đặc tả bởi `Protocol` và `Port`.

Thông tin này có thể được cung cấp bởi một công cụ quản lý tường lửa như Smart Firewall [11]. Dưới đây là một ví dụ HACL cho phép lưu lượng TCP chảy từ cổng 80 đến webServer:

hacl(internet,webServer,TCP,80).

B. Đặc tả hình thức các lỗ hổng

Phần lớn các lỗ hổng phần mềm liên quan tới bảo mật, do vậy các cơ quan báo cáo lỗ hổng đã cung cấp một đặc tả hình thức cho các lỗ hổng đã được phát hiện, định dạng mà máy có thể đọc được. Việc đặc tả hình thức các lỗ hổng cùng với các công cụ xử lý tự động đã giúp cho việc trao đổi báo cáo các lỗ hổng trở nên dễ dàng hơn và kịp thời đối phó với chúng. Đặc tả lỗ hổng bao gồm 2 phần: phần thứ nhất là đặc tả nhận dạng lỗ hổng (giúp nhận biết sự tồn tại

[1] Trong hệ điều hành Unix, một chương trình setuid sẽ có đặc quyền gốc (root) khi nó được thực hiện

của lỗ hổng đó trên hệ thống) và phần thứ hai là đặc tả ngữ nghĩa (ảnh hưởng của lỗ hổng đó trên hệ thống như thế nào). Trong 2 phần sau chúng tôi mô tả ngắn gọn về OVAL, một ngôn ngữ đặc tả hình thức cho việc ghi nhận các lỗ hổng và ICAT một cơ sở dữ liệu cung cấp các ảnh hưởng của lỗ hổng.

1. Ngôn ngữ OVAL và máy quét

OVAL là một ngôn ngữ dựa trên XML nhằm xác định kiểm tra cấu hình máy. Khi một lỗ hổng phần mềm mới được phát hiện, một định nghĩa OVAL có thể xác định làm thế nào để kiểm tra sự tồn tại của lỗ hổng đó trên máy tính. Sau đó định nghĩa OVAL được đưa vào một máy quét OVAL tương thích, máy này sẽ tiến hành kiểm tra theo quy định và báo cáo kết quả. Hiện nay định nghĩa lỗ hổng OVAL có sẵn cho Windows, Red Hat Linux và nền tảng Solaris. Máy quét OVAL-compliant có sẵn cho nền tảng Windows và Red Hat Linux. Các định nghĩa lỗ hổng OVAL đã được tạo ra từ năm 2002 và định nghĩa mới đang được đệ trình và xem xét trên một cơ sở hàng ngày.

Đầu ra của một máy quét OVAL được chuyển đổi vào các mệnh đề Datalog như sau:

```
vulExists(webServer,'CVE-2002-0392', httpd).
```

2. Ảnh hưởng của các lỗ hổng

Ta có thể tìm thấy thông tin chi tiết về các lỗ hổng từ trang web của OVAL. Ví dụ OVAL mô tả lỗi OVAL296 là:

Multiple unknown vulnerabilities in Linux kernel 2.4 and 2.6 allow local users to gain privileges or access kernel memory, ...

Mô tả ngắn dưới dạng không hình thức này nhấn mạnh ảnh hưởng của các lỗ hổng như là: các lỗ hổng này có thể bị khai thác như thế nào và hậu quả nó gây ra là gì? Nếu một cơ sở dữ liệu máy có thể đọc được cung cấp thông tin về ảnh hưởng của một lỗi như: *bug 2961 is only locally exploitable, one could formally prove properties like if all local users are trusted, then the network is safe from remote attacker*.

Tuy nhiên OVAL không biểu diễn được thông tin về ảnh hưởng của một lỗ hổng trong một định dạng mà máy tính có thể đọc được. Trong khi đó, cơ sở dữ liệu ICAT phân loại ảnh hưởng của một lỗ hổng thành hai phần: phạm vi khai thác và hậu quả.

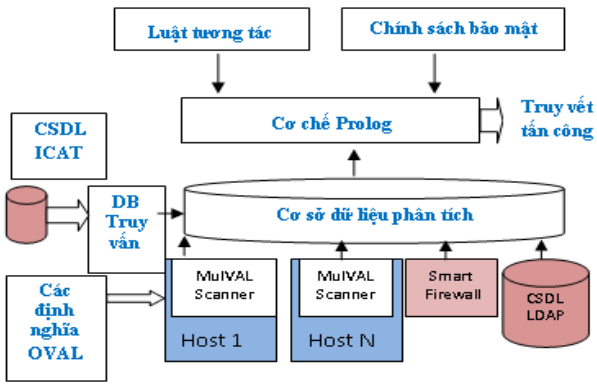
- Phạm vi khai thác: *local, remote.*
- Hậu quả: mất tính bảo mật (*confidentiality loss*), mất tính toàn vẹn (*integrity loss*), từ chối dịch vụ (*denial of service*) và leo thang đặc quyền (*privilege escalation*).

Một khai thác (tấn công) địa phương (*local*) yêu cầu kẻ tấn công đã có một vài truy cập địa phương trên máy chủ. Một khai thác từ xa (*remote*) thì không yêu cầu như vậy. Hậu quả của hai cách tấn công phổ biến đó là leo thang đặc quyền và từ chối dịch vụ. Hiện tại tất cả các định nghĩa OVAL đã có mục ICAT tương ứng (cả hai có thể được tham chiếu bởi CVEId). Nếu OVAL và ICAT được so khớp với nhau vào một cơ sở dữ liệu mà cung cấp được cả hai thông tin.

Thông tin trong cơ sở dữ liệu ICAT được chuyển vào các mệnh đề Datalog như sau:

```
vulProperty('CVE-2004-00495',localExploit,privEscalation).
```

C. Kết hợp các thông tin cấu hình



Hình 1. Cấu trúc của MulVAL

Hình 1 minh họa kiến trúc của MulVAL với các nguồn dữ liệu của cơ sở dữ liệu đã phân tích. Máy quét MulVAL, chạy trên mỗi máy chủ cá nhân, cung cấp thông tin cấu hình máy. *Smart Firewall* [11] cung cấp cấu hình mạng về các bộ HACL. Cơ sở dữ liệu LDAP cung cấp thông tin ràng buộc chính. Các chính sách bảo mật được xác

định bởi người quản trị hệ thống và người quản trị hệ thống cũng cần phải xác định ràng buộc dữ liệu như là một phần của chính sách bảo mật. Các nguồn dữ liệu bên trái đến từ các cơ quan báo cáo lỗ hổng (*bug-reporting*) của bên thứ ba. Họ cung cấp các đặc tả hình thức của các lỗi phần mềm, trong định nghĩa OVAL và từ các cơ sở dữ liệu NVD.

Phần tiếp theo ta sẽ tìm hiểu làm thế nào từ các lỗ hổng được báo cáo bởi các máy quét có thể phát hiện được và sinh đồ thị tấn công có thể có trong một hệ thống mạng.

IV. THUẬT TOÁN SINH ĐỒ THỊ TẤN CÔNG

A. Đồ thị tấn công

Đồ thị tấn công logic là một đồ thị có hướng và có thể được biểu diễn dưới dạng một cây với các liên kết chéo có thể có giữa các nút. Hình 2 cho thấy biểu diễn dạng cây của đồ thị tấn công logic. Có hai loại nút trong đồ thị: nút dẫn xuất (*derivation node*) và nút sự kiện (*fact node*). Một nút dẫn xuất được biểu diễn là một hình chữ nhật và một nút sự kiện được biểu diễn bởi một vòng tròn. Ngoài ra còn có hai loại nút sự kiện: nút sự kiện nguyên thủy (biểu diễn bởi vòng tròn) và nút sự kiện dẫn xuất (biểu diễn bởi một vòng tròn với một số trong đó).

Mỗi nút sự kiện trong một đồ thị logic được gán nhãn với một biểu diễn logic dưới dạng một vị ngữ áp dụng cho đối số của nó. Nút gốc là mục tiêu của kẻ tấn công; ví dụ trong hình 2: `execCode (attacker, Workstation, root)`, có nghĩa là "*những kẻ tấn công có thể thực thi mã tùy ý như là người sử dụng root trên máy Workstation* ". Mỗi nút dẫn xuất được gán nhãn với một luật tương tác, các luật này được sử dụng cho bước khai thác. Trong biểu diễn cây, tất cả các nút trong được bắt đầu với một nút được đánh số trong một dấu ngoặc (< >), tiếp theo là nhãn của nút. Một nút lá không có số và được đánh dấu bởi một cặp ngoặc vuông ([]). Các cạnh trong đồ thị biểu diễn mối quan hệ phụ thuộc. Một nút sự kiện phụ thuộc vào một hoặc nhiều nút dẫn xuất, mỗi trong số đó biểu diễn một ứng dụng của một luật tương tác tạo ra sự kiện; Một nút dẫn xuất phụ thuộc vào một hay nhiều nút sự kiện, cùng với nhau chúng đáp ứng các điều kiện tiên quyết của luật (*rule*). Vì vậy, một đồ thị tấn công logic là một đồ thị có hướng hai phía. Các nút dẫn xuất phục vụ như là một trung gian giữa một sự kiện và "nguyên nhân" của nó, tức là cách thức "sự kiện" trở thành sự thật. Từ một sự kiện có thể có nhiều cách khác nhau để trở thành hiện thực, các nút dẫn xuất được dẫn từ một nút sự kiện tạo thành một phép tuyển (*disjunction*). Một nút dẫn xuất đại diện cho viêc áp dụng thành công một quy tắc tương tác, tại đó tất cả các điều kiện tiên quyết của nó được thỏa mãn bởi các nút con của nó. Do đó các nút sự kiện dẫn từ một nút dẫn xuất hình thành một phép hội (*conjunction*).

Trong ví dụ ở hình 2 nút <2> có hai nút dẫn xuất (cũng chính là nút con của nó): r2a và r2b (lưu ý rằng biểu diễn cây sử dụng ký hiệu '||' để biểu thị rằng một nút sự kiện có nhiều hơn một dẫn xuất). Đó là, có hai cách để kẻ tấn công có thể sửa đổi tập tin trên *fileserver*. Cách thứ nhất là để có được quyền *root* trên máy chủ tập tin bằng cách khai thác lỗi *CVE-2003-0252* trong chương trình *mountd* và cách thứ hai là sử dụng chương trình *NFS Shell*. Cả hai phụ thuộc vào các điều kiện mà kẻ tấn công đã đạt được một số quyền truy cập vào *webserver* (nút 5). Trong biểu diễn cây, có một liên kết chéo (==> <5>) trỏ đến nút 5 trong nhánh dẫn xuất thứ hai (<r2b>).

```
<0>|--execCode(attacker,workStation,root)
   <r0>Rule5: Trojan horse installation
     <1>|--accessFile(attacker,workStation,write,/usr/local/share)
        <r1>Rule14: NFS semantics
             []-nfsMounted(workStation,/usr/local/share,fileServer,/export,read)
          <2>||--accessFile(attacker,fileServer,write,/export)
             <r2a>Rule10: execCode implies file access
                  []-fileSystemACL(fileServer,root,write,/export)
               <3>|--execCode(attacker,fileServer,root)
                  <r3>Rule3: remote exploit of a server program
                       []-networkServiceInfo(fileServer,mountd,rpc,100005,root)
                       []-vulExists(fileServer,CVE-2003-0252,mountd,
                                    remoteExploit,privEscalation)
                    <4>|--netAccess(attacker,fileServer,rpc,100005)
                       <r4>Rule6: multi-hop access
                            []-hacl(webServer,fileServer,rpc,100005)
                         <5>|--execCode(attacker,webServer,apache)
                            <r5>Rule3: remote exploit of a server program
                                 []-networkServiceInfo(webServer,httpd,tcp,80,apache)
                                 []-vulExists(webServer,CAN-2002-0392,httpd,
                                                        remoteExploit,privEscalation)
                              <6>|--netAccess(attacker,webServer,tcp,80)
                                 <r6>Rule7: direct network access
                                      []-hacl(internet,webServer,tcp,80)
                                      []-located(attacker,internet)
             <r2b>Rule15: NFS shell
                  []-hacl(webServer,fileServer,rpc,100003)
                  []-nfsExportInfo(fileServer,/export,write,webServer)
                  |--execCode(attacker,webServer,apache)==> <5>
```

Hình 2. Ví dụ đồ thị tấn công logic biểu diễn dưới dạng cây

Một đồ thị tấn công logic có thể được xem như là một đồ thị dẫn xuất cho một truy vấn Datalog thành công. Có thể có nhiều cách khác nhau để lấy được một sự kiện trong Datalog (tương ứng với nhiều đường để đột nhập vào một

mạng), ở đây các nút dẫn xuất biểu diễn cho một bước dẫn xuất có thể. Theo một cách logic, một nút dẫn xuất là một nút "and", tại đó tất cả các nút con của nó là các đối số của một phép hội; một nút sự kiện dẫn xuất là một nút "or", tại đó tất cả các nút con của nó biểu diễn những cách khác nhau để dẫn xuất chúng. Một nút sự kiện nguyên thủy là một nút lá trong đồ thị. Nó biểu diễn cho một phần của thông tin cấu hình. Sau đây là định nghĩa chính thức của đồ thị tấn công logic.

Định nghĩa 1. (N_r, N_p, N_d, E, L, G) là một đồ thị tấn công logic, trong đó N_r, N_p và N_d là ba tập hợp của các nút rời nhau trong đồ thị, $E \subset (N_r \times (N_p \cup N_d)) \cup (N_d \times N_r)$, L là ánh xạ từ một nút tới nhãn của nó, và $G \in N_d$ là mục tiêu của kẻ tấn công.

N_r, N_p và N_d tương ứng là tập hợp nút dẫn xuất, các nút sự kiện nguyên thủy và các nút sự kiện dẫn xuất. Một sự kiện là nguyên thủy nếu nó đến từ đầu vào tới công cụ suy diễn MulVAL. Một sự kiện dẫn xuất là kết quả của việc áp dụng quy tắc tương tác lặp đi lặp lại trên các sự kiện đầu vào. Các cạnh trong một đồ thị tấn công logic chỉ có thể đi từ một nút sự kiện dẫn xuất đến một nút dẫn xuất, hoặc từ một nút dẫn xuất đến một nút sự kiện. Các chức năng ghi nhãn ánh xạ một nút sự kiện tới sự kiện nó biểu diễn và một nút dẫn xuất tới các quy tắc được sử dụng cho dẫn xuất. Một cách hình thức, ngữ nghĩa của một đồ thị tấn công logic được định nghĩa như sau:

Tính chất 1. Với mỗi nút dẫn xuất R, đặt P là nút cha của nút R và C là tập hợp các nút con của R.

Khi đó $(\wedge L(C)) \Rightarrow L(P)$ là một thể hiện của luật tương tác L(R).

Trong đó $\wedge$ là toán tử hội.

Định nghĩa 2. Truy vết mô phỏng tấn công.

*TraceStep : : = **because** (interactionRule, Fact, Conjunct)*

Fact : : = predicate(list of constant)

Conjunct : : = [list of Fact]

interactionRule là một chuỗi duy nhất liên kết với một luật tương tác MulVAL. Một danh sách (*list*) được biểu diễn như một loạt các mục (*item*) cách nhau bởi dấu phẩy. Ngữ nghĩa của một bước truy vết là:" *Conjunct* $\Rightarrow$ *Fact* là một thể hiện của của *interactionRule*". Nó ghi lại lý do tại sao một mục tiêu là đúng (*true*) trong khi đánh giá Datalog.

B. Thuật toán sinh đồ thị tấn công

Đầu vào: Tập τ chứa tất cả các *TraceStep*, và $\mathcal{G}$ là mục tiêu của kẻ tấn công
Đầu ra: đồ thị tấn công logic (N_r, N_p, N_d, E, L, $\mathcal{G}$).

1. $N_r, N_p, N_d, E, L \leftarrow \emptyset$
2. Vòng lặp với mỗi $t \in \tau$ {

   Đặt t = **because**(*interactionRule*, *Fact*, *Conjunct*)
3. Tạo một nút dẫn xuất r
   $N_r \leftarrow N_r \cup \{r\}$
   $L \leftarrow L \cup \{r \rightarrow interactionRule\}$
4. Tìm $n \in N_d$ sao cho $L(n) = Fact$,
5. Nếu không tồn tại n {
   Tạo một nút *fact* mới n
   $L \leftarrow L \cup \{n \rightarrow Fact\}$
   $N_d \leftarrow N_d \cup \{n\}$
   }
6. $E \leftarrow E \cup \{(r,n)\}$
7. Vòng lặp với mỗi *fact* f trong *Conjunct* {
8. Tìm nút *fact* $c \in (N_p \cup N_d)$ sao cho
   $L(c) = f$,
9. Nếu c không tồn tại {
   Tạo một nút *fact* mới c
   $L \leftarrow L \cup \{c \rightarrow f\}$
   Nếu f là nguyên thủy { $N_p \leftarrow N_p \cup \{c\}$}
   Ngược lại {$N_d \leftarrow N_d \cup \{c\}$}
   }
10. $E \leftarrow E \cup \{(r,c)\}$
    }
   }

Hình 3. Thuật toán sinh đồ thị tấn công logic

Một đồ thị tấn công logic được xây dựng từ những thông tin có được ở các bước truy vết. Thuật toán sinh đồ thị tấn công logic được mô tả trong Hình 3. Nói một cách đơn giản, mỗi thuật ngữ TraceStep trở thành một nút dẫn xuất

trong đồ thị tấn công. Các trường Fact trong các bước truy vết trở thành các nút cha và các trường Conjunct là các nút con của nó. Số lần lặp tối đa cho các vòng lặp trong tại dòng 7 giống như số lượng lớn nhất các điều kiện tiên đề trong số tất cả các quy tắc tương tác, nó là hằng số cho một bộ quy tắc tương tác cố định. Vì vậy, nếu các vòng lặp trên dòng 4 và dòng 8 là thời gian hằng số, thì các thuật toán xây dựng đồ thị cần có thời gian tuyến tính theo số lượng các bước truy vết.

C. Phân tích độ phức tạp

Quá trình tính toán một đồ thị tấn công logic bao gồm hai giai đoạn. Giai đoạn đầu tiên tính mô phỏng dấu vết tấn công thông qua Datalog đánh giá trong XSB; giai đoạn thứ hai là xây dựng cấu trúc dữ liệu đồ thị tấn công bằng cách sử dụng thuật toán trong Hình 3.

1. Độ phức tạp của tính toán truy vết tấn công

Mỗi thế hệ truy vết tấn công chỉ đưa ra một hằng số thời gian cận trên cho mọi dẫn xuất Datalog thành công. Vì vậy, sự phức tạp của giai đoạn đầu tiên là giống như sự phức tạp của việc đánh giá các chương trình MulVAL Datalog trong XSB. Sự phức tạp của việc đánh giá một chương trình Datalog cố định đối với đầu vào kích thước biến phụ thuộc vào các chi tiết cụ thể của chương trình. Các tài liệu về XSB có một số thảo luận về làm thế nào để xác định sự phức tạp của việc đánh giá một chương trình Datalog trong XSB [13]. Để dễ hiểu hơn, chúng ta hãy xem xét các quy tắc tương tác Datalog sau trong MulVAL:

```
netAccess(Attacker,H2,Protocol,Port):-
execCode(Attacker,H1,_User),
hacl(H1,H2,Protocol,Port).
```

Ý nghĩa của các quy tắc là: nếu một kẻ tấn công có thể trở thành một người dùng cục bộ trên máy H1 và mạng cho phép `H1`, `H2` truy cập thông qua `Protocol` và `Port` , sau đó kẻ tấn công có thể truy cập `H2` thông qua giao thức và cổng. Quy luật này cho thấy truy cập *multi-hop* trong một mạng: một kẻ tấn công có thể chiếm quyền, hoặc sử dụng một máy để làm bước đệm cho việc thỏa hiệp các máy khác.

Khi XSB đánh giá luật này đầu tiên nó sẽ tính toán tất cả các máy có thể có một kẻ tấn công thực thi mã tùy ý trên (các mục tiêu phụ đầu tiên) và sau đó nó sẽ tìm kiếm đầy đủ tất cả H1và H2 giữa các mạng có thể truy cập (các mục tiêu phụ thứ hai H2). Khi tất cả các bộ dữ liệu được tính toán, vị từ mục tiêu *netAccess* sẽ được tính bằng cách kết hợp các kết quả của hai mục tiêu phụ. Sử dụng mô hình so khớp mẫu trong XSB rất hiệu quả do việc sử dụng bảng băm và thử nghiệm. Vì vậy, chi phí thời gian bị chi phối bởi số lượng các bộ dữ liệu trung gian cần phải được tính toán. Việc tính toán trung gian có thể gọi quy tắc tương tác khác. Trong bảng thực thi của XSB, một lời gọi sẽ tính toán tất cả các kết quả của mục tiêu đó, do đó có thể được tái sử dụng sau này.

Định lý 1. *Đánh giá các luật tương tác của MulVAL với bộ dữ liệu cấu hình biểu diễn N máy mất $O(N^2)$ bước dẫn xuất.*

Nếu mô hình so khớp mẫu XSB là thời gian hằng số, từng bước dẫn xuất cần có thời gian hằng số để kết thúc và thời gian chạy tổng thể để đánh giá MulVAL Datalog sẽ là bậc hai. Vì mỗi bước truy vết đã được tạo ra bởi một bước dẫn xuất trong đánh giá Datalog.

Hệ quả 1. Số lượng của các term truy vết được tạo ra trong mô phỏng tấn công là $O(N^2)$.

2. Độ phức tạp của việc xây dựng đồ thị

Định lý 2. *Đồ thị tấn công logic của mạng gồm N máy có kích thước lớn nhất là $O(N^2)$.*

Chứng minh. Có sự tương ứng một-một giữa các term *TraceStep* và nút dẫn xuất. Cho D là số lượng các bước truy vết, sau đó có D nút dẫn xuất trong đồ thị. Nếu số lượng tối đa điều kiện tiên quyết cho một quy tắc tương tác là *m,* số cạnh tối đa của đồ thị là *mD* và số lượng tối đa các nút sự kiện là *m*D + 1. Theo Hệ quả 1, chúng ta đã biết D là O (N^2) và do đó là *m*D+1.

Định lý 3. *Các thuật toán xây dựng đồ thị trong Hình 3 mất độ phức tạp thời gian $O(\delta N^2)$ để hoàn thành, trong đó N là số lượng host trong mạng và δ là chi phí thời gian tối đa cho việc tra cứu trong bảng tại dòng 4 và 8 của thuật toán.*

Chứng minh. Các vòng lặp trong thuật toán xây dựng đồ thị trong hình 3 đi qua tất cả các *term TraceStep*. Theo hệ quả 1, chúng ta biết có O (N^2) *term* như vậy. Trong mỗi lần lặp, thuật toán tạo ra một nút dẫn xuất cho *term TraceStep* và tạo liên kết từ nút cha đến các nút con của nó. Mỗi hành động có chi phí thời gian là hằng số, ngoại trừ việc tìm kiếm trên bảng ở dòng 4 và 8.

Do đó, thời gian cần thiết để xây dựng các cấu trúc dữ liệu đồ thị là bậc hai theo số các máy chủ, tạo ra một bảng tra cứu với thời gian hằng số để lưu trữ các nút đồ thị. Để thực hiện, chỉ đơn giản là sử dụng cấu trúc dữ liệu

"*map*" trong thư viện chuẩn của C++, trong đó thời gian tra cứu là log(*n*). Từ định lý 2, chúng ta biết rằng kích thước bảng là $O(N^2)$. Vì vậy, $\delta = \log(N^2)$ và thời gian chạy để sinh ra đồ thị sẽ là $O(N^2 \log(N))$.

V. THỰC NGHIỆM

Về mặt hình thức, phương pháp tiếp cận dựa trên logic để sinh đồ thị tấn công cũng giống như bởi Sheyner đã sử dụng các phương pháp ad-hoc để sinh đồ thị tấn công. Tuy nhiên việc sử dụng các kĩ thuật logic hợp lý sẽ ít gặp phải các lỗi hơn là so với các thuật toán tùy chỉnh thiết kế, đặc biệt là đối với các vấn đề phức tạp của phân tích an ninh. Một ngữ nghĩa logic rõ ràng cho các biểu đồ tấn công cũng làm cho việc phân tích sẽ dễ dàng hơn dựa trên các cấu trúc dữ liệu của đồ thị.

Để xác định các lỗ hổng phần mềm ảnh hưởng đến an ninh của một mạng xác định ta phải xem xét sự tương tác giữa các thành phần mạng. Đối với một công cụ phân tích lỗ hổng để có được tính hữu dụng trong thực tế cần phải có hai tính năng quan trọng. Đầu tiên đó là mô hình sử dụng trong phân tích phải có khả năng tự động tích hợp các đặc tả hình thức của lỗ hổng từ *bug-reporting*. Thứ hai, các phân tích phải có khả năng mở rộng quy mô đến các mạng với hàng ngàn máy. MulVAL đáp ứng được cả hai điều kiện trên.

Khi sử dụng công cụ Sheyner để phân tích an ninh của một mạng thực tế kết quả cho thấy rằng thời gian sinh đồ thị và kích thước đồ thị là rất lớn. Ví dụ: một mạng lưới gồm 10 máy chủ với 5 lỗ hổng trên máy chủ mất khoảng 15 phút để tạo ra đồ thị và kết quả là trong một đồ thị gồm 10 triệu cạnh. Trong đồ thị tấn công của Sheyner mỗi nút là tập hợp các biến Boolean mã hóa toàn bộ trạng thái mạng ở giai đoạn tấn công. Trong khi số lượng các biến là đa thức theo kích thước của mạng, số lượng có thể có của các trạng thái là số mũ.

Do vậy sử dụng MulVAL (Multihost, multistage Vulnerability Analysis), một framework cho việc mô hình hóa sự tương tác của các lỗi phần mềm với hệ thống và các cấu hình mạng để phân tích ảnh hưởng của các lỗ hổng tồn tại trong hệ thống mạng sẽ cho kết quả tốt hơn. MulVAL sử dụng ngôn ngữ mô hình hóa Datalog. Thông tin về lỗi cơ sở dữ liệu được cung cấp bởi cộng đồng *bug-reporting*. Thông tin cấu hình của từng máy, mạng và các thông tin liên quan khác tất cả đều được mã hóa như là một sự kiện của Datalog. Đây là một Framework được thực hiện trên nền tảng Red Hat Linux.

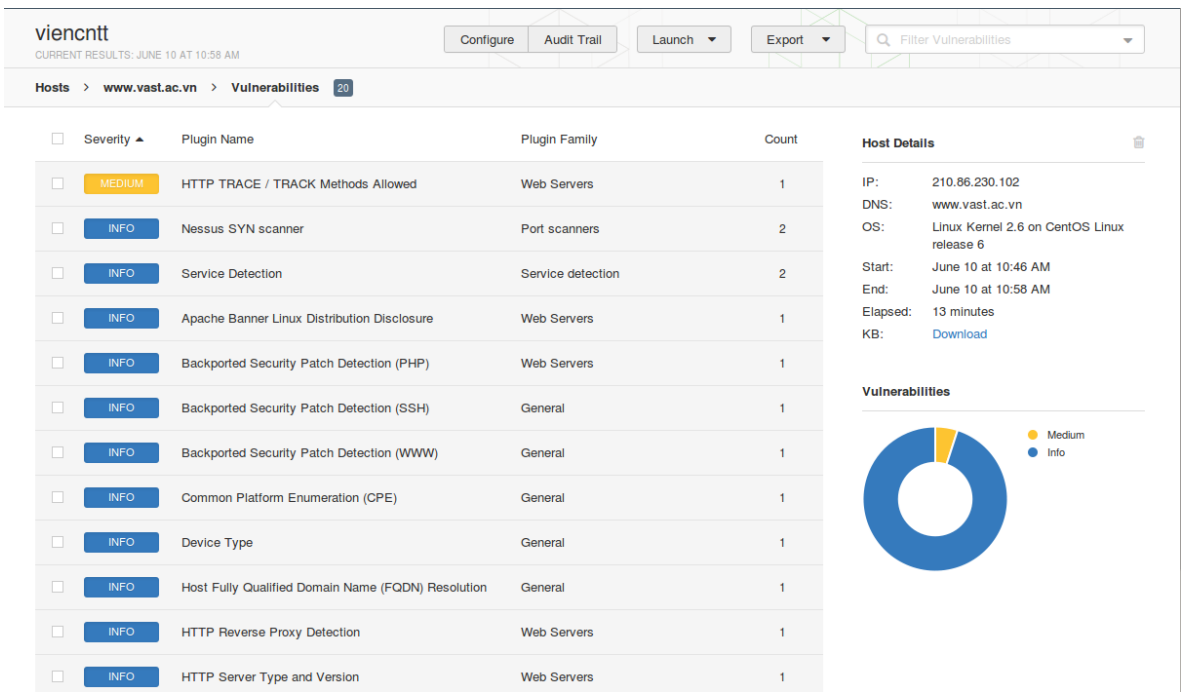
Kết quả thử nghiệm sử dụng MulVAL để phân tích an ninh trên một mạng thực tế:

Nhóm thực nghiệm phân tích an ninh trên máy chủ mạng của Viện Hàn lâm Khoa học và Công nghệ Việt Nam [16]. Các bước thực hiện như sau:

Bước 1: Sử dụng máy quét OVAL để quét tìm lỗ hổng trong mạng và tạo file đầu vào cho MulVAL.

- Thời gian quét: 13 phút
- Tổng số lỗ hổng phát hiện: 20

Hình sau minh họa một số lỗ hổng phát hiện được:



Hình 4. Danh sách các lỗ hổng

File đầu ra của máy quét được xuất ra dưới định dạng XML chứa thông tin các lỗi phát hiện được.

Bước 2: Tạo file đầu vào cho MulVAL

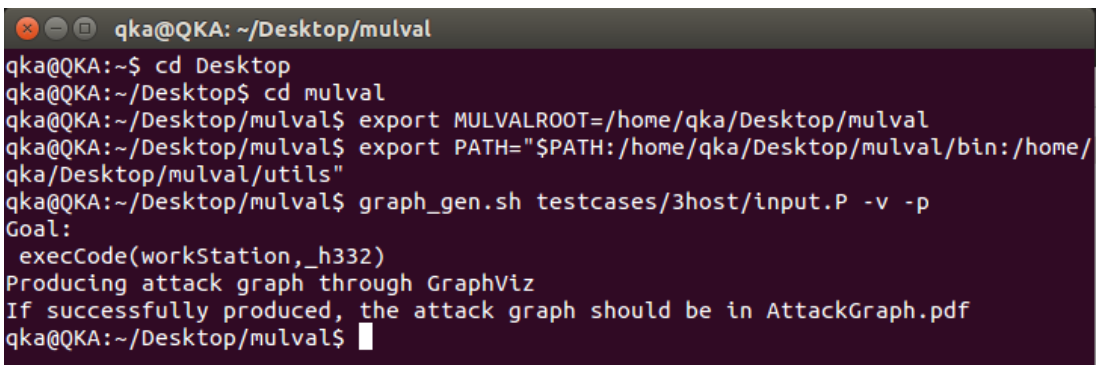
Chuyển File kết quả của máy quét từ định dạng XML sang các mệnh đề Datalog:

```
inCompetent('www.vast.ac.vn_victim').
hasAccount('www.vast.ac.vn_victim', 'www.vast.ac.vn', user).
attackerLocated(internet).
attackGoal(execCode('www.vast.ac.vn', _)).
hacl(_,_,_,_).
```

Bước 3: Chạy thử nghiệm trên MulVAL để xem ảnh hưởng của các lỗi quét được có tạo nên 1 cuộc tấn công không?

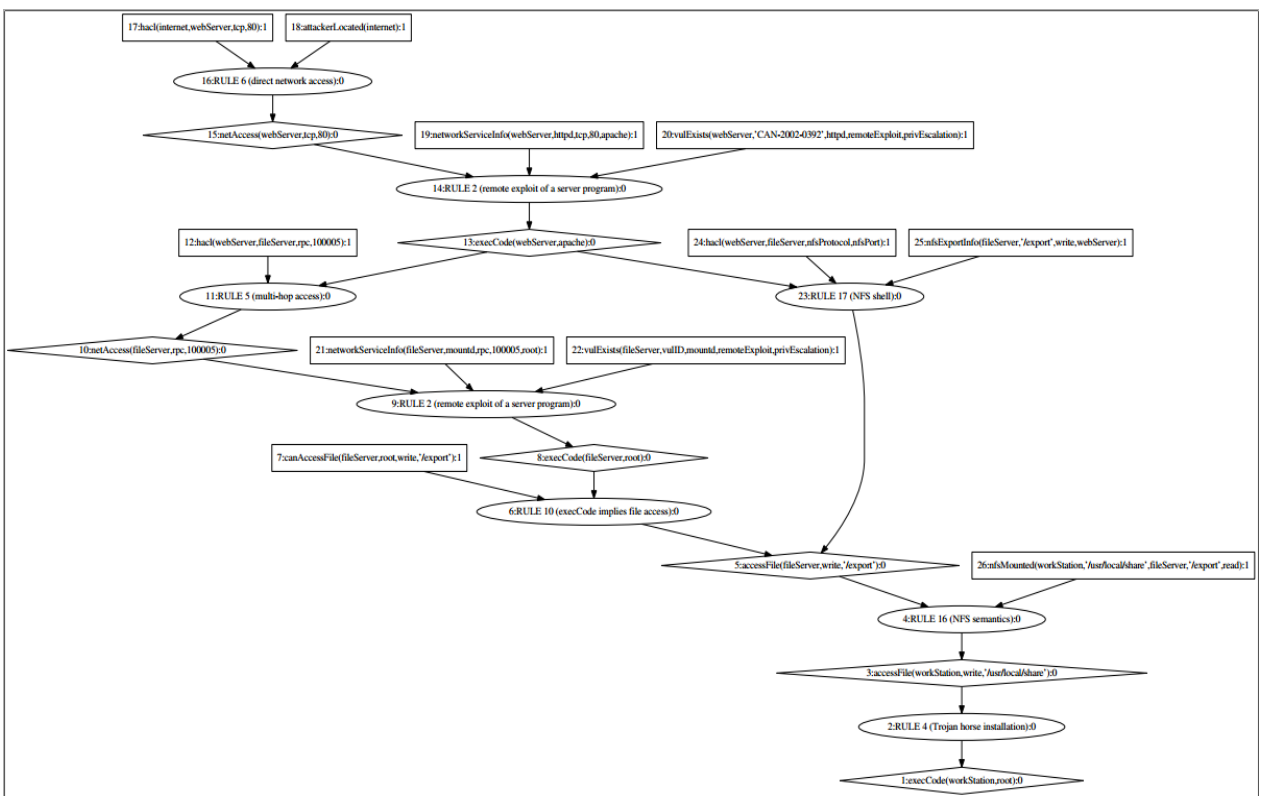
Kết quả với các lỗi mà máy quét phát hiện ra trên hệ thống mạng chưa đủ để tạo nên một cuộc tấn công tiềm ẩn.

Do giới hạn nguồn tài nguyên để thử nghiệm thực tế với một mạng có thể bị tấn công nên nhóm đã chạy trên file ví dụ đầu vào của MulVAL. Kết quả thông báo sinh được đồ thị tấn công trong file AttackGraph.pdf (Hình 5).



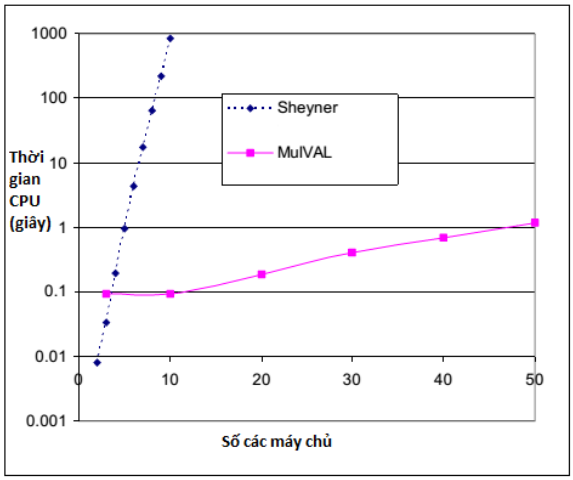
Hình 5. Kết quả chạy file ví dụ mẫu

Hình 6 là đồ thị tấn công logic.



Hình 6. Đồ thị tấn công logic

Dưới đây là biểu đồ so sánh thời gian chạy của CPU khi sinh đồ thị tấn công của hai công cụ phân tích an ninh mạng tự động là MulVAL và Sheyner. Kết quả cho thấy thời gian chạy của MulVAL nhanh hơn rất nhiều so với Sheyner.



Hình 7. Biểu đồ so sánh thời gian chạy của CPU khi sinh đồ thị tấn công của hai công cụ MulVAL và Sheyner

VI. KẾT LUẬN

Trong bài báo này chúng tôi đã trình bày một phương pháp tiếp cận hình thức dựa trên logic trong phân tích an ninh an toàn mạng. Phương pháp này có khả năng ứng dụng và thực tế nhằm đảm bảo an ninh không gian mạng cho đất nước ta hiện nay. Đó là việc sử dụng lập trình logic và các mệnh đề Datalog để mô hình hóa thông tin cấu hình mạng, cấu hình máy chủ và việc đặc tả thông tin lỗ hổng mạng. Tiếp đó là tìm hiểu thuật toán sinh đồ thị tấn công, chạy thực nghiệm trên một mạng thực tế để phát hiện ra các lỗ hổng mạng và sinh đồ thị tấn công có thể có. Hướng nghiên cứu tiếp theo là phối hợp cùng các nhà quản trị mạng trong việc nghiên cứu mô hình hóa hình thức các chính sách bảo mật, các tập luật tương tác do các nhà quản trị mạng đặt ra, để bổ sung thêm vào các bộ luật suy diễn trong MulVAL nhằm giúp cho việc quản lý an ninh mạng tốt hơn, hiệu quả hơn.

TÀI LIỆU THAM KHẢO

[1] O. Udrea, C. Lumezanu, and J. S. Foster, "Rule-based static analysis of network protocol implementations," in *Proceedings of the 15th Conference on USENIX Security Symposium – Volume* 15, ser. USENIX-SS'06. Berkeley, CA, USA: USENIX Association, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267336.1267350.

[2] C. He and J. C. Mitchell, "Analysis of the 802.11i 4-way handshake," in *Proceedings of the 3rd ACM Workshop on Wireless Security, ser*. WiSe '04. New York, NY, USA: ACM, 2004, pp. 43–50. [Online]. Available: http://doi.acm.org/10.1145/1023646.1023655.

[3] N. Jovanovic, C. Kruegel, and E. Kirda, "Pixy: A static analysis tool for detecting web application vulnerabilities (short paper)," in *Proceedings of the 2006 IEEE Symposium on Security and Privacy, ser*. SP '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 258–263. [Online]. Available: http://dx.doi.org/10.1109/SP.2006.29

[4] V. B. Livshits and M. S. Lam, "Finding security vulnerabilities in java applications with static analysis," in *Proceedings of the 14th Conference on USENIX Security Symposium - Volume* 14, ser. SSYM'05. Berkeley, CA, USA: USENIX Association, 2005, pp.18-18.[Online].Available: http://dl.acm.org/citation.cfm?id=1251398.1251416.

[5] D. Farmer and E. Spafford. The COPS security checker system. *In Proceedings of the Summer Usenix Conference*, 1990.

[6] R. Deraison. Nessus Scanner. *http://www .nessus.org*.

[7] Ou, X., Appel, A. W.: A logic-programming approach to network security analysis.Phd, Princeton University Princeton (2005).

[8] W. F. Clocksin and C.S. Mellish. *Programming in Prolog*. Springer-Verlag New York, Inc., 1987.

[9] John DeTreville. Binder, a logic-based security language. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 105. IEEE Computer Society, 2002.

[10] Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. Delegation Logic: A logic-based approach to distributed authorization. *ACM Transaction on Information and System Security (TISSEC)*, February 2003.

[11] James Burns, Aileen Cheng, Proveen Gurung, David Martin, Jr., S. Raj Rajagopalan, Prasad Rao, and Alathurai V. Surendran. Automatic management of network security policy. In DARPA *Information Survivability Conference and Exposition* (DISCEX II'01), volume 2, Anaheim, California, June 2001.

[12] D. S. Warren. *Programming in Tabled Prolog*. Department of Computer Science SUNY @ Stony Brook, July 1999.

[13] D. S. Warren. On the Complexity of Tabled Datalog Programs. Department of Computer Science, SUNY @ Stony Brook, Stony Brook, NY 11794-4400, U.S.A., July 1999.

[14] Xinming Ou, Sudhakar Govindavajhala, and Andrew W. Appel. Mulval: A logic-based network security analyzer. In 14th USENIX Security Symposium, Baltimore, MD, USA, August 2005.

[15] http://vast.ac.vn.

RESEARCH FORMAL METHOD BASED ON LOGIC IN NETWORK SECURITY ANALYSIS

Nguyen Thi Anh Phuong, Nguyen Truong Thang, Tran Manh Dong, Bui Thi Thu

***ABSTRACT**: Nowadays, with the continuous development of the Internet and the fields of information technology, hardware devices, software applications and Internet services have become very popular making the economy Of countries around the world growing.*

However, along with the rapid development there is always potential risks, causing the financial loss, reputation... for organizations or individuals to participate in the service. Therefore, the problem of ensuring safety and security of information in the process of transmitting information in the network environment is always a hot issue in research and practical application. Especially with an enterprise network, the problem of ensuring network security and information security is becoming increasingly difficult. Now with the spread of software vulnerabilities, system administrators are facing many challenges and difficulties in securing cybersecurity. This paper will present an approach of logic-based methodology in network security analysis to identify existing vulnerabilities in the system and possible attacks on the system. It helps security professionals and system administrators better manage the configuration of an enterprise network and can control the risks of network security.