Tạp chí Khoa học Trường Đại học Cần Thơ
website: sj.ctu.edu.vn

NHẬN DẠNG MÃ ĐỘC SỬ DỤNG CƠ CHẾ BĂM THEO CHỈ MỤC TRÊN KHÔNG GIAN DỮ LIỆU PHÂN HOẠCH

Trương Minh Nhật Quang[1]
[1] *Khoa Công nghệ Thông tin, Trường Đại học Kỹ thuật - Công nghệ Cần Thơ*

Thông tin chung:
Ngày nhận: 03/09/2013
Ngày chấp nhận: 21/10/2013

Title:
Recognizing malicious code using hash indexing mechanism in categorical data space

Từ khóa:
Mã độc, nhận dạng mã độc, băm theo chỉ mục, không gian dữ liệu phân hoạch

Keywords:
Malicious code, recognize malicious code, hash indexing, categorical data space

ABSTRACT

To protect a computer system from threat infections, an anti-virus system needs to scan for malicious codes which may appear in target systems. In this paper, we present a technique to recognize malicious codes quickly by using hash indexing mechanism in categorical space. First, in training phase, the dataset of malicious codes will be separated into clusters which have the same characteristics. After that, we build a special rule set in a hash indexing format of ordered cluster-buckets. Next, in recognition phase, we extract object's characteristics and code them into a checksum value by using some popular hash algorithms. Then this value is used as a key to search for the same rule in indexed rule space. Finally, the system returns the scanning process results.

We have built this technique for D2 Anti-virus 2013 running on Windows XP SP3 in a computer with Intel Core 2 Duo E7200 – 2.53 GHz. Using a dataset of 615,880 malicious signatures, D2 needs only 05 seconds to examine 105,330 MB of 8,696 executable files. Its average scanning speed is about 21,651MBps. The experimental results denote that this is an effective technique to improve the scanning speed of anti-virus systems nowadays.*

TÓM TẮT

Để bảo vệ máy tính khỏi các đe dọa lây nhiễm, hệ phòng chống virus máy tính cần quét kiểm tra mã độc trong hệ thống đích. Trong bài viết này, chúng tôi trình bày kỹ thuật nhận dạng nhanh mã độc sử dụng cơ chế băm theo chỉ mục trên không gian phân hoạch. Đầu tiên, trong giai đoạn luyện, tập mẫu chữ ký mã độc được phân thành các cụm có cùng đặc điểm. Sau đó, chúng tôi xây dựng một tập luật đặc biệt dưới dạng bảng băm các bucket luật được sắp xếp thứ tự theo cụm. Tiếp theo, ở giai đoạn nhận dạng, chúng tôi tiến hành trích chọn đặc trưng và biến đổi thành một giá trị tổng kiểm đại diện cho đối tượng bằng các thuật giải băm phổ biến. Giá trị này sau đó được dùng làm khóa tìm kiếm luật nhận dạng của đối tượng trong không gian luật đã được sắp xếp. Cuối cùng, hệ trả về kết quả quá trình duyệt quét.

Chúng tôi đã cài đặt kỹ thuật này cho hệ D2 Anti-virus 2013 chạy hệ điều hành Windows XP SP3 trên máy tính Intel Core 2 Duo E7200 – 2.53 GHz. Sử dụng tập 615,880 mẫu mã độc, D2 chỉ tốn 5 giây để kiểm tra 105,330 MB dữ liệu của 8,696 tập tin thực thi. Tốc độ quét trung bình của D2 đạt 21,651 MB/giây. Kết quả thực nghiệm chứng tỏ đây là kỹ thuật hiệu quả nhằm tăng tốc duyệt quét cho các hệ phòng chống virus máy tính ngày nay.*

1 GIỚI THIỆU

Trong bối cảnh ngày càng gia tăng các cuộc tấn công thâm nhập mạng, các hệ phòng chống virus máy tính không ngừng cải tiến nhằm sớm phát hiện các loại mã độc hại, kịp thời loại trừ các tác nhân thâm nhập, tăng cường công tác bảo vệ an toàn hệ thống và an ninh mạng. Hiện nay, hiệu quả hoạt động của các hệ phòng chống virus máy tính rất được người dùng quan tâm. Để cải tiến tốc độ truy vấn chữ ký mã độc trên cơ sở dữ liệu (CSDL) lớn, chúng tôi nghiên cứu kỹ thuật nhận dạng nhanh mã độc sử dụng cơ chế băm theo chỉ mục trên không gian phân hoạch. Đầu tiên, tập mẫu chữ ký mã độc được phân thành các cụm có cùng đặc điểm. Sau đó, chúng tôi xây dựng tập luật dưới dạng bảng băm các bucket được sắp xếp thứ tự theo cụm. Tiếp theo, chúng tôi tiến hành trích chọn đặc trưng và biến đổi thành một giá trị số đại diện cho đối tượng bằng các hàm tổng kiểm (checksum) phổ biến. Giá trị này được dùng làm khóa tìm kiếm luật nhận dạng của đối tượng trong không gian luật đã sắp xếp. Kết quả truy vấn sẽ cho biết tình trạng an ninh của đối tượng chẩn đoán có phải là mã độc hay không.

Để đánh giá hiệu quả kỹ thuật, chúng tôi xây dựng tập 615,880 mẫu mã độc và động cơ chẩn đoán của hệ D2 Anti-virus* 2013 (phiên bản dành cho Windows XP). Máy tính sử dụng bộ vi xử lý Intel Core 2 Duo E7200 – 2.53GHz. Kiểm tra 8,696 tập tin thực thi, hệ D2 tốn 5 giây, tốc độ quét trung bình 21,651MB/giây. Kết quả thực nghiệm chứng tỏ đây là kỹ thuật hiệu quả tăng tốc duyệt quét cho các hệ phòng chống virus máy tính ngày nay.

2 CÁC VẤN ĐỀ LIÊN QUAN

2.1 Mã độc trong tấn công an ninh mạng

2.1.1 Vai trò của mã độc trong kịch bản tấn công an ninh mạng

Trong các cuộc tấn công mạng, đặc biệt là hình thức tấn công từ chối dịch vụ lan tràn (DDoS – Distributed Denial of Service), hacker thường cài đặt các đoạn mã độc hại (malicious code) vào các phần mềm ác ý (malware) rồi tìm cách cấy vào các máy trạm (zombie) để hình thành mạng botnet, chuẩn bị cho cuộc tấn công. Dưới sự điều khiển của hacker, zombie sẽ thu thập, đánh cắp thông tin, liên lạc hacker... Khi lượng zombie đủ lớn, hacker sẽ phát lệnh điều khiển botnet đồng loạt tấn công vào một mục tiêu trên mạng [3].

Có nhiều dạng thức thi hành mã lệnh. Dạng mã lệnh thực thi thuận tiện cho việc truyền tải, thi hành trên máy đích hiện nay là định dạng tập thực thi (executable files). Mỗi mã độc có đoạn mã đặc trưng nhận dạng gọi là chữ ký (signature) mã độc [5]. Có hai hình thức thiết kế mã độc: thi hành phụ thuộc và thi hành độc lập.

2.1.2 Mã độc thi hành phụ thuộc

Tiêu biểu cho dạng mã độc này là các loại virus máy tính (gọi tắt là virus), trojan horse (gọi tắt là trojan), spyware, adware... Virus là loại mã độc có khả năng ký sinh mã vào các tập thực thi khác và nắm quyền thực thi khi ứng dụng chủ thi hành. Mã virus được thiết kế bằng Hợp ngữ (Assembly Language), độc lập với ngôn ngữ lập trình của vật chủ. Các thủ tục thi hành của trojan (và spyware, adware...) được hacker cài đặt trong lúc thiết kế mã độc. Virus ký sinh vào các tập thực thi tạo hiệu ứng lây lan, trong khi trojan (và spyware, adware...) không tự lây lan; chúng được ngụy trang dưới vỏ bọc các phần mềm hiền lành, lừa người sử dụng tải về máy và thi hành chúng [8].

2.1.3 Mã độc thi hành độc lập

Việc cài đặt malware vào máy người dùng dưới dạng một phần mềm hoàn chỉnh có thể gây nhiều bất lợi: kích thước lớn, thay đổi registry, lộ diện tiến trình... Để khắc phục, hacker biên dịch mã độc dưới dạng file thực thi rút gọn rồi lan truyền trên mạng thông qua các lỗ hổng bảo mật hệ thống. Tiêu biểu cho dạng mã độc này là các loại sâu mạng (networm), cửa hậu (backdoor), intruder, dropper, rootkit... Chúng được hacker ưa chuộng vì kích thước nhỏ, lây lan nhanh, hoạt động độc lập (stand alone), dễ ẩn náu [4].

Hiệu quả của kế hoạch tấn công mạng phụ thuộc vào số lượng zombie. Để gia tăng khả năng lây lan, kéo dài thời gian ẩn náu..., hacker thường kết hợp nhiều hình thức mã độc: virus chứa sâu mạng, trojan kết xuất dropper... Do đó, phân loại mã độc như trên chỉ là tương đối.

2.2 Các hệ phòng chống virus máy tính

Lịch sử phát triển mã độc bắt nguồn từ virus máy tính (sau đây gọi tắt là virus). Thời kỳ đầu các phần mềm phòng chống virus được gọi là anti-virus. Ngày nay thuật ngữ anti-virus (AV) dùng để chỉ loại phần mềm bảo vệ máy tính khỏi sự xâm nhập của các loại mã độc hại (virus, trojan, worm, backdoor, rootkit, spyware...) bằng cách đối chiếu dữ liệu hệ thống với thông tin mô tả các loại mã độc đã biết trong một CSDL được cập nhật thường xuyên [8]. Các AV hoạt động dựa vào tập mẫu thường áp dụng các tiếp cận chuỗi mã, tiếp cận hành vi, tiếp cận máy học.

2.2.1 Tiếp cận chuỗi mã

Hoạt động theo nguyên lý so khớp mẫu (signature matching), các AV đối chiếu thông tin thi hành, mã lệnh hoặc/và dữ liệu của đối tượng chẩn đoán với thông tin mã độc được tổ chức, cập nhật trong CSDL [7]. Tiếp cận chuỗi mã giúp AV nhận dạng mã độc đã biết với độ chính xác cao. Tuy nhiên hoạt động của AV sẽ kém hiệu quả khi kiểm tra các mã độc chưa được cập nhật thông tin vào CSDL mẫu.

2.2.2 Tiếp cận hành vi

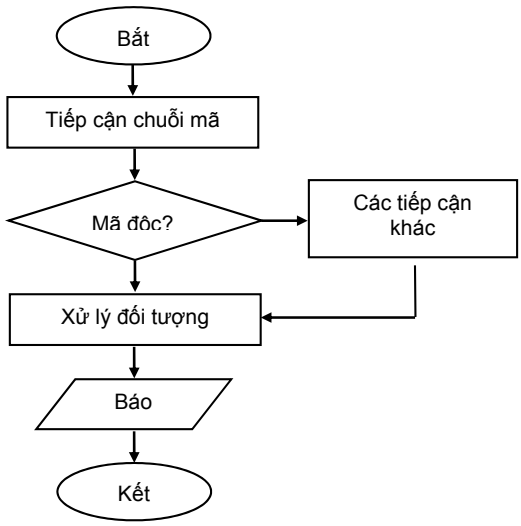
Các mã độc giống nhau thường có các hành vi giống nhau. Nghiên cứu mã độc dưới góc độ thi hành, tiếp cận này dựa vào khái niệm *hành vi* (behavior checking) để nhận dạng mã độc bằng cách tải và thử thi hành (heuristic) mã lệnh nghi ngờ trong môi trường mô phỏng (emulated environment) nhằm giải mã các hành vi lạ [7]. Tiếp cận hành vi (còn gọi là tiếp cận heuristic) giải quyết khá tốt các loại mã độc đa hình (polymorphism). Tuy nhiên khi xử lý các hành vi nghi ngờ của đối tượng, các AV heuristic cần hỗ trợ quyết định nên thường gây lúng túng cho người dùng ít kinh nghiệm.

2.2.3 Tiếp cận máy học

Tiếp cận máy học nhận dạng virus máy tính (Machine Learning Approach to Anti-virus System) xây dựng các mô hình học dựa vào cơ sở tri thức (CSTT) gồm tập mẫu chữ ký mã độc và luật nhận dạng. Tiếp cận này chia bài toán chẩn đoán mã độc thành nhiều lớp bài toán con rồi áp dụng các hình thức học phù hợp cho từng bài toán cụ thể [2][10][11]. Sử dụng kiến thức chuyên gia tổ chức trong CSTT, các hệ học 'dạy' cho máy tính xử lý các trường hợp tương tự bằng cách khái quát hóa các tình huống chẩn đoán thông qua các luật dẫn xuất (production rule) suy luận từ CSTT của hệ. Tiếp cận máy học kết hợp hệ chuyên gia giúp hệ thường xuyên cập nhật và tự tăng trưởng tri thức. Nhận dạng hướng luật giúp AV có khả năng dự báo các mẫu mã độc tương đồng. Tuy nhiên, hiệu quả chẩn đoán của AV phụ thuộc vào chất lượng CSTT của hệ.

2.2.4 Vai trò của tiếp cận chuỗi mã trong kiến trúc anti-virus

Dù áp dụng tiếp cận nào, nhiệm vụ của các AV là sớm đưa ra bằng chứng xuất hiện nếu có của một loại mã độc đã biết với tên gọi cụ thể. Vấn đề này chỉ có thể giải quyết bằng tiếp cận chuỗi mã ở giai đoạn tiền xử lý (Hình 1).



Hình 1: Lưu đồ xử lý tổng quát của AV

2.3 Cơ chế tìm kiếm, so khớp mẫu

Quét virus là quá trình tìm kiếm chữ ký mã độc trong đối tượng. Liên quan đến chủ đề bài viết có các kỹ thuật tìm kiếm tuyến tính, tìm kiếm bằng chỉ mục và tìm kiếm bằng bảng băm.

2.3.1 Tìm kiếm tuyến tính

Tìm kiếm tuyến tính (linear search) là kỹ thuật duyệt từng phần tử trong danh sách đến khi tìm thấy (so khớp) giá trị mong muốn hoặc đến cuối danh sách. Đơn giản trong thiết kế, tìm kiếm tuyến tính thích hợp cho các danh sách đủ nhỏ, danh sách chưa sắp thứ tự, các tổ chức lưu trữ tuần tự như danh sách liên kết, tập tin định kiểu… Tuy nhiên khi hoạt động trên danh sách đã sắp xếp hoặc danh sách lớn, phương pháp này không tối ưu và kém hiệu quả.

2.3.2 Tìm kiếm theo chỉ mục

Trong các hệ quản trị CSDL quan hệ, tìm kiếm theo chỉ mục (index search) là kỹ thuật tìm kiếm vị trí mẫu tin trên cấu trúc bảng dựa vào thông tin chỉ mục. Đầu tiên, thông tin đối tượng được tổ chức trong bảng 2 chiều; mỗi bảng có 1 trường khóa chính (primary key) lưu trị thuộc tính duy nhất (unique) của đối tượng. Tiếp theo, một bảng chỉ mục dạng *<primary key, #record>* được khởi tạo (*#record* chứa số hiệu mẫu tin của đối tượng trong bảng chính). Sau đó, bảng chỉ mục được sắp xếp theo (sort by) trường *primary key* rồi lưu vào tổ chức đĩa. Quá trình tìm kiếm mẫu tin có khóa *primary key* trong bảng chính được thực hiện bằng thuật toán tìm kiếm trên danh sách thứ tự (chẳng

hạn, tìm kiếm nhị phân), lấy giá trị *#record* rồi mở bảng chính, dịch đến vị trí mẫu tin tương ứng.

Sử dụng thuật toán tìm kiếm trên danh sách thứ tự nên tìm kiếm theo chỉ mục nhanh hơn tìm kiếm tuyến tính. Tuy nhiên, số mẫu tin trong bảng chính luôn bằng với số mẫu tin trong bảng chỉ mục nên kỹ thuật tìm kiếm theo chỉ mục nói chung không giảm không gian dữ liệu. Mục 2.3.3 sau đây sẽ trình bày kỹ thuật tìm kiếm hướng giảm không gian dữ liệu.

2.3.3 Tìm kiếm bằng bảng băm

Tìm kiếm bằng bảng băm (hash table search) là kỹ thuật tìm kiếm theo hướng giảm không gian dữ liệu. Ở giai đoạn thiết kế, một hàm băm (hash function) được sử dụng để đặc tả vị trí mẫu tin, sau đó lưu từng mẫu tin vào cấu trúc bảng theo mục (bucket) do hàm băm cung cấp. Ở giai đoạn truy xuất, hàm băm sẽ giúp định vị bucket phục vụ các thao tác cơ bản trên CSDL.

Hiệu quả tìm kiếm dữ liệu trên bảng băm phụ thuộc vào chất lượng hàm băm. Một hàm băm lý tưởng sẽ tạo k ánh xạ 1-1 từ tập mẫu sang bảng băm. Nếu băm 'không mịn', bảng băm thưa gây lãng phí ô nhớ (có nhiều bucket rỗi không dùng đến). Ngược lại, băm 'quá mịn' sẽ gây xung đột địa chỉ mẫu tin (hàm băm trả về một bucket cho nhiều mẫu tin khác nhau). Để giải quyết đụng độ trên bảng băm, chúng tôi sử dụng phương pháp kết nối dạng *<bucket, record list>*; trong đó, *record list* là danh sách liên kết các mẫu tin đụng độ *bucket* do hàm băm trả về.

2.4 Các nghiên cứu liên quan

2.4.1 Tình hình nghiên cứu cải tiến tốc độ quét AV trên máy cá nhân

Có nhiều yếu tố ảnh hưởng tốc độ quét của AV: tài nguyên khả dụng của hệ thống, mức ưu tiên tiến trình, kỹ thuật xử lý, tối ưu mã lệnh, tình trạng dữ liệu, số lượng mẫu trong CSDL... Đề tài này tập trung nghiên cứu các vấn đề liên quan đến dữ liệu ở hai khía cạnh: dữ liệu bên ngoài (dữ liệu chẩn đoán) và dữ liệu bên trong (CSDL mẫu của hệ).

Phân loại dữ liệu chẩn đoán, giảm không gian tìm kiếm để tránh quá tải là giải pháp được các AV sử dụng phổ biến [13]. Bên cạnh đó, một số giải pháp tăng tốc duyệt quét như lập hồ sơ tiền sử (history) lưu vết trạng thái hoạt động; ghi nhận dữ liệu an toàn [12]; đánh dấu dữ liệu nguyên trạng (not modified) [14]; tổ chức CSDL mẫu hợp lý [15]; kiểm soát các hoạt động thay đổi tập thi hành [17]; phân luồng xử lý (thread processing) [16]... cũng được các AV đầu tư nghiên cứu.

2.4.2 Tình hình nghiên cứu cải tiến tốc độ quét mã độc trên mạng theo hướng băm

Số lượng mã độc trên thị trường ngày càng nhiều khiến tập mẫu của AV gia tăng từ vài trăm ngàn đến hàng triệu chữ ký. Giải pháp cải tiến tốc độ quét mã độc theo hướng băm gần đây cũng được quan tâm nghiên cứu. Năm 2007, Ozgun Erdogan và Pei Cao đề xuất kỹ thuật Hash-AV sử dụng cơ chế đệm L2 trong CPU để quét nhanh chữ ký mã độc lan truyền trên mạng [7]. Thực nghiệm trên máy tính CPU Pentium 4 2.4GHz, sử dụng tập mẫu 120,000 chữ ký mã độc, kết quả Hash-AV cho phép tốc độ quét thông lượng trên mạng đạt 200Mb/s.

Năm 2011, Po-Ching Lin và *ctv* giới thiệu kỹ thuật Hybrid Algorithm of Backward Hashing And Automaton Tracking. Băm ngược chuỗi mẫu, ghi vết quá trình tìm kiếm, bỏ qua các chuỗi đã đối chiếu nhằm kéo dài vectơ khoảng cách trong thuật toán tìm kiếm của Aho-Corasick, kỹ thuật này giúp cải tiến 50% tốc độ thông lượng quét mạng bằng Hash-AV sử dụng CSDL mẫu của phần mềm nguồn mở Clam-AV [9].

3 NHẬN DẠNG MÃ ĐỘC SỬ DỤNG CƠ CHẾ BĂM THEO CHỈ MỤC TRÊN KHÔNG GIAN PHÂN HOẠCH

3.1 Tổ chức cơ sở tri thức

Trong các hệ học, CSTT chứa thông tin mô tả đối tượng và các luật nhận dạng đối tượng. Tiếp cận máy học và hệ chuyên gia tổ chức CSTT ở giai đoạn luyện qua bốn bước: (1) xây dựng tập mẫu, (2) phân hoạch tập mẫu, (3) rút luật nhận dạng và (4), tổ chức tập luật [2].

3.1.1 Xây dựng tập mẫu

Có nhiều dạng thức thi hành mã lệnh (và cũng từng ấy các mô tả mã độc tương ứng). Bài viết này tập trung vào các loại mã độc dạng thực thi EXE. Đầu tiên, các tập mã độc được thu thập cho chuyên gia phân tích kỹ thuật, phân loại. Sau đó, chuyên gia sẽ cập nhật các mẫu tin mô tả mã độc vào cấu trúc bảng dạng:

Malware = <Tên gọi, Tập thuộc tính, Chữ ký>

Trong đó:

- Tên gọi: tên mã độc, dùng cho giai đoạn báo cáo.
- Tập thuộc tính: các đặc trưng thi hành của mã độc, dùng phân hoạch tập mẫu.

– Chữ ký: đặc trưng 'nhân dạng' mã độc, dùng xây dựng luật nhận dạng.

3.1.2 Phân hoạch tập mẫu

Các phương pháp tìm kiếm tuyến tính trên tập mẫu lớn có nhiều hạn chế. Chúng tôi giải quyết vấn đề này bằng chiến lược tìm kiếm trên không gian tập mẫu được phân cụm.

Phân cụm (clustering) là một trong những phương pháp khai phá dữ liệu phổ biến trong các hệ học. Bằng các cấu trúc dữ liệu đặc biệt, quá trình phân cụm sử dụng các giải thuật thích hợp nhằm phát hiện sự giống nhau giữa các mục dữ liệu để gom (hoặc tách) thành từng cụm có cùng tính chất. Để phân hoạch không gian tập mẫu, chúng tôi chọn hình thức phân cụm phân cấp (*hierarchical clustering*) dạng tách nhóm. Không cần xác định số cụm từ đầu, phương pháp này sử dụng đồ thị cây các cụm (*dendogram*) với các nút biểu diễn cụm trung gian, nút lá biểu diễn các cụm kết quả.

Cây là mô hình thích hợp cho việc tổ chức dữ liệu phân cụm tách nhóm [1]. Chúng tôi chọn V-Tree để phân cụm tập mẫu [6]. Cho số nguyên m>1, cây m-phân V-Tree là một cây có:

– Các nút lá đều có số mức như nhau.

– Mỗi nút trung gian N có m/2 đến m nút con, nút gốc có từ 2 đến m con.

– Mỗi nút có nhãn chứa giá trị của nút và danh sách các nút thuộc cây con của nút.

Chúng tôi cài đặt mỗi nút V-Tree có 3 trường: (1) nhãn nút chứa trị thuộc tính, (2) danh sách các nút con và (3), danh sách các mẫu trong cụm (dùng lưu trữ thông tin chỉ mục dành cho giai đoạn xử lý, xem mục 3.1.4).

CSDL mẫu được tổ chức dưới dạng bảng 2 chiều *X(p,k)* có p cột (thuộc tính) k dòng (mẫu tin). Giải thuật phân cụm tập mẫu như sau:

```
Khởi tạo nút gốc
Đối với mỗi mẫu tin thứ j của X:
    Chọn nút gốc là Nút_hiện_tại
    Đối với mỗi thuộc tính thứ i:
        Nếu Nút_hiện_ tại chưa có nút con nhãn
        x[i,j] thì:
            Tạo nút mới có nhãn x[i,j]
            Cho nút nhãn x[i,j] là con của Nút_hiện
            tại
        Đặt nút con nhãn x[i,j] của Nút_hiện_tai làm
        Nút_hiện_tại
        Bổ sung mẫu tin j vào danh sách các mẫu
        thuộc cụm của Nút_hiện_tại
```

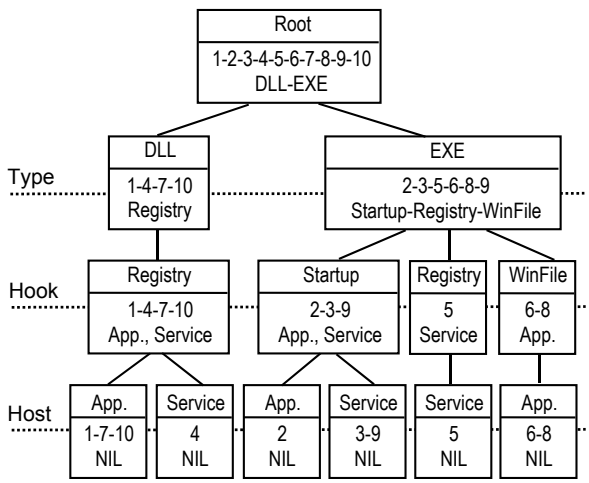
Xét CSDL mẫu gồm 5 thuộc tính (Bảng 1). Trong đó, các thuộc tính Type, Hook, Host được chọn tham gia phân cụm (p'= p–2).

Bảng 1: CSDL mẫu chứa 10 mã độc

Name	Type	Hook	Host	Signature
M1	DLL	Registry	App.	3F2D9A..
M2	EXE	StartUp	App.	E9124F..
M3	EXE	StartUp	Service	60E89F..
M4	DLL	Registry	Service	583E6F..
M5	EXE	Registry	Service	E8558B..
M6	EXE	WinFile	App.	C0317F..
M7	DLL	Registry	App.	902D3F..
M8	EXE	WinFile	App.	7E5D9B..
M9	EXE	StartUp	Service	347A9C..
M10	DLL	Registry	App.	4B80FF..

Cluster1: M1, M7, M10
Cluster2: M4
Cluster3: M5
Cluster4: M2
Cluster5: M3, M9
Cluster6: M6, M8

Hình 2: Mô tả kết quả phân cụm CSDL dựng bằng V-tree, kết quả thu được 6 nhóm mã độc



Hình 3: Kết quả phân cụm tập mẫu bằng V-tree

3.1.3 Rút luật nhận dạng

Sau khi phân hoạch tập mẫu, mỗi nhánh trên V-Tree sẽ chứa một luật theo mệnh đề Horn. Rút luật nhận dạng bằng cách duyệt đường đi từ nút gốc đến nút lá. Trong ví dụ trên, tập 10 mẫu tin được tách thành 6 cụm mã độc tương ứng với 6 luật nhận dạng như sau:

1. IF (Type=DLL) AND
   (Hook=Registry) AND
   (Host=App.) THEN Cluster1
2. IF (Type=DLL) AND

(Hook=Registry) AND
(Host=Service) THEN Cluster2
3. IF (Type=EXE) AND
(Hook=Startup) AND
(Host=App.) THEN Cluster3
4. IF (Type=EXE) AND
(Hook=Startup) AND
(Host=Sevice) THEN Cluster4
5. IF (Type=EXE) AND
(Hook=Registry) AND
(Host=Service) THEN Cluster5
6. IF (Type=EXE) AND
(Hook=WinFile) AND
(Host=App.) THEN Cluster6

3.1.4 Tổ chức tập luật băm theo chỉ mục

Mục đích của phân cụm V-Tree nhằm tách tập mẫu thành các nhóm có cùng đặc tính dữ liệu về mặt *giá trị*. Các cụm kết quả (bố trí ở nút lá của V-Tree) được tách rời và không phủ nhau nên số cụm thu được (m cụm) luôn nhỏ hơn số mẫu dữ liệu (k mẫu). Điều này luôn đúng vì luật có tính khái quát, do đó chỉ cần lập luận trên m luật là có thể bao quát toàn bộ k mẫu (m<k). Như vậy, ngoài chức năng cung cấp tri thức mã độc của chuyên gia, luật nhận dạng còn có thể sử dụng làm chỉ dẫn truy xuất dữ liệu. Chúng tôi bổ sung trường thứ ba vào mỗi nút trên V-Tree nhằm cung cấp giá trị trả về cho hàm băm $H(f(A),k)$ trong phép trích chọn đặc trưng thi hành $f(A)$ của đối tượng.

Đầu tiên, tập luật nhận dạng được tổ chức dạng bảng 2 chiều $R(\psi, £, r)$ có r phần tử luật; mỗi luật ψ liên kết với danh sách £ các thể hiện thỏa luật ψ. Trong ví dụ trên, CSDL luật R có 6 phần tử luật ($r = 6$) được minh họa như sau:

Luật 1: ψ_1 » {1,7,10}
Luật 2: ψ_2 » {4}
Luật 3: ψ_3 » {5}
Luật 4: ψ_4 » {2}
Luật 5: ψ_5 » {3,9}
Luật 6: ψ_6 » {6,8}

Ký hiệu '»' đặc tả phép 'liên kết' (link-to).

Tiếp theo, các bucket luật thành viên ψ_i được biến đổi thành các giá trị tổng kiểm bằng các thuật toán băm thông dụng như CRC (Cyclic Redundancy Check), MD5 (Message - Digest Algorithm 5) hoặc SHA-1 (Secure Hash Algorithm). Sau cùng, sắp xếp tập luật theo chỉ mục checksum (Bảng 2) rồi lưu trữ CSDL luật. Kết thúc giai đoạn luyện trên máy chuyên gia.

Bảng 2: CSDL luật chứa 6 luật thành viên

No.	Rule	Checksum	Link-to
1	ψ_2	58,487,876	4
2	ψ_5	76,455,645	3-9
3	ψ_4	156,496,857	2
4	ψ_1	325,474,326	1-7-10
5	ψ_6	437,665,473	6-8
6	ψ_3	758,355,475	5

3.2 Nhận dạng mã độc dựa vào tri thức

Giai đoạn nhận dạng - xử lý trên máy khách vận dụng các thủ tục suy diễn và lập luận trong động cơ quét (scan engine), căn cứ vào thông tin trong tập chữ ký mẫu và các tri thức mô tả trong CSDL luật để chẩn đoán đối tượng A. Quá trình này được thực hiện qua bốn bước: (1) trích chọn đặc trưng, (2) mã hóa nhân dạng, (3) truy vấn luật và (4), so khớp mẫu.

3.2.1 Trích chọn đặc trưng

Đầu vào của bước này là đối tượng A cần chẩn đoán. Đầu ra là một tập mô tả các đặc trưng thi hành của đối tượng dùng cho bước mã hóa nhân dạng tiếp theo. Để tương thích với khuôn dạng tri thức mô tả mã độc M ở giai đoạn luyện, bước này sử dụng lại hàm trích chọn đặc trưng $f(M)$ của chuyên gia dạng:

$$f(A) = \{\delta_i\} \ \forall \ (i < p)$$

Trong đó, $\delta(q)$ là phép trích chọn tự động tập thuộc tính mã độc ở giai đoạn trước. Ví dụ, áp dụng hàm f trích chọn đặc trưng trên các đối tượng bất kỳ U, V nhận được kết quả như sau:

$f(U)$ = {DLL, Registry, Driver}
$f(V)$ = {EXE, Startup, App.}

Ví dụ này sử dụng $i = 3$ ($i = p' < p$)

3.2.2 Mã hóa nhân dạng

Nhiệm vụ của bước trích chọn đặc trưng là tạo lập nhân dạng (identification) của đối tượng theo khuôn dạng luật trong CSTT. Khái niệm nhân dạng trong tiếp cận học chẩn đoán mã độc dựa vào lập luận: "nếu đối tượng có nhân dạng giống với nhân dạng tội phạm thì có thể đối tượng chính là tội phạm đang truy nã".

Sau khi trích chọn đặc trưng, nhân dạng đối tượng sẽ được mã hóa bằng thuật toán băm h của chuyên gia. Ví dụ, hồ sơ chẩn đoán các đối tượng

U, *V* nhận được mã số nhân dạng sau:

$ID(U) = h(f(U)) = 87,954,543$

$ID(V) = h(f(V)) = 758,355,475$

Các mã số này sau đó được sử dụng làm giá trị tìm kiếm theo trường khóa chính Checksum trong CSDL luật của hệ.

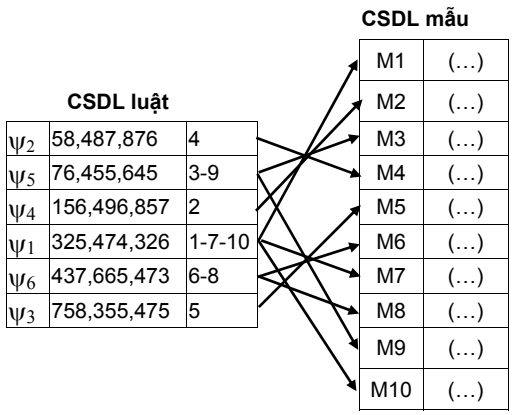
3.2.3 Truy vấn luật

Khi tra cứu mã số nhân dạng *ID*(*A*) trong CSDL luật, nếu không tìm thấy có thể kết luận *A* là đối tượng an toàn vì trong CSTT chưa có các mô tả mã độc nào có đặc trưng thi hành giống với *A*. Nếu tìm thấy, cần đưa *A* vào diện nghi vấn, sau đó tiếp tục xác minh các đặc điểm nhân dạng đặc thù để có kết luận chính xác. Trong ví dụ trên, *U* là an toàn vì trong tập luật không có giá trị checksum 87,954,543 nào. Tuy nhiên cần cảnh giác *V* vì mã số nhân dạng 758,355,475 của nó đang có trong CSDL luật.

Mỗi luật nhận dạng đại diện cho một cụm (nhóm) các mã độc cùng đặc trưng. Xác minh hồ sơ là quá trình đối chiếu thông tin nhân dạng của *A* với các mã độc cùng nhóm. Cho các số nguyên *i*, *j*; hàm băm *H* truy vấn danh sách mã độc cùng nhóm trên tập mẫu *X(p,k)* có dạng:

$$H(\psi_m) = \psi_i \gg \{X(p, j)\} \ \forall \ (i < m); (j < k)$$

Theo thiết kế này, hàm *H* chỉ đơn giản trả về danh sách ở trường thứ ba của nút lá trên nhánh V-Tree của luật ψ_i tương ứng. Chiến lược sử dụng tập luật nhận dạng làm chỉ mục băm tìm kiếm chữ ký mã độc được minh họa ở Hình 3.



Hình 3: Cơ chế băm tập luật theo chỉ mục

3.2.4 So khớp mẫu

Đây là công đoạn rà soát, đối chiếu thông tin lần cuối trước khi quyết định 'bắt giam' đối tượng nhằm hạn chế sai sót trong tác nghiệp. Có nhiều cách so khớp mẫu: dò tìm mã phổ biến, trích xuất các giá trị đặc trưng tại các địa chỉ nhạy cảm, trích chọn mã ngẫu nhiên…[5].

Trong bài viết này, so khớp mẫu là bước kiểm tra sự xuất hiện giá trị thuộc tính thứ *p* của quan hệ *X*(*p*,*k*) trong đối tượng chẩn đoán. Nếu đối chiếu đúng, có thể kết luận đối tượng *A* là mã độc. Ngược lại, tiếp tục đối chiếu các mẫu còn lại cho đến khi phát hiện. Gọi *M*(*p*,*n*) là tập mã độc có mã nhận dạng trùng với mã nhân dạng của *A*. Giải thuật so khớp mẫu như sau:

```
Tìm thấy ← false
j ← 1
Lặp lại
    Nếu Chữ ký(A) = M(p,j) thì
       Kết luận ('Phát hiện mã độc', M(1,j))
       Tìm thấy ← true;
    Ngược lại
       j ← j+1
Đến khi Tìm thấy hoặc (j>n)
```

4 KẾT QUẢ THỰC NGHIỆM

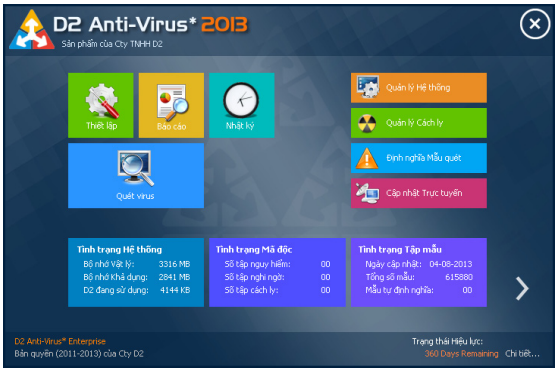
4.1 Thử nghiệm với D2 Anti-virus*

*4.1.1 Giới thiệu hệ D2 Anti-virus**

D2 Anti-virus* (Diagnose and Destroy Computer Viruses) là phần mềm quét virus hướng tiếp cận máy học của Việt Nam. Thiết kế theo mô hình của một hệ chuyên gia, hoạt động của hệ D2 gồm ba giai đoạn: giao tiếp chuyên gia (giai đoạn học), động cơ suy diễn (giai đoạn nhận dạng, xử lý) và giao tiếp người dùng (giai đoạn tổng kết, báo cáo).

Thực hiện trên máy chuyên gia, giai đoạn học xây dựng CSTT dưới dạng tập các mô tả tri thức mã độc và luật nhận dạng khẳng định dương trên tập mẫu. Giai đoạn xử lý và báo cáo thực hiện trên máy khách. Hoạt động hướng tác tử [10], động cơ quét của hệ chứa các thuật toán học phân loại, động cơ suy diễn, thực thi các lớp bài toán học và xử lý các trường hợp lây nhiễm. Cuối cùng, giai đoạn báo cáo tổng hợp kết quả các bài toán, phân tích quá trình, tạo các giao tiếp hội thoại, thu nhận ý kiến người dùng, báo cáo kết quả và tăng trưởng tri thức.

Phiên bản D2 Anti-virus* 2013 (Hình 4) có một số cải tiến quan trọng về giao diện, tăng tốc độ quét, dự báo mã độc hướng heuristic, ước lượng mã tương đồng, phát hiện hành vi lây nhiễm trên thiết bị lưu trữ cá nhân…



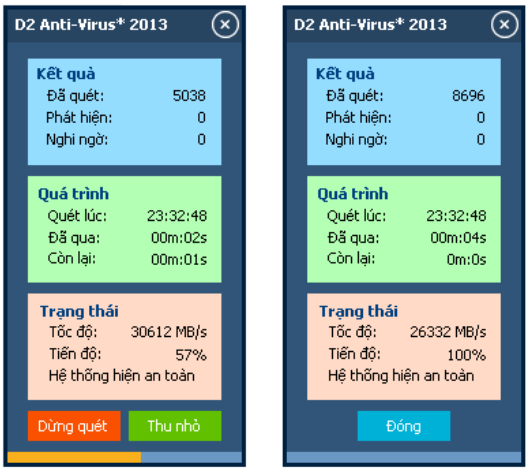
Hình 4: Phần mềm D2 Anti-Virus 2013*

*4.1.2 Kết quả thử nghiệm D2 Anti-virus**

Trong giai đoạn học, chúng tôi thiết kế gói công cụ D2 Expert Utilities hỗ trợ phân tích kỹ thuật tập mẫu, đặt tên mã độc, trích chọn đặc trưng thi hành, xây dựng CSDL, phân hoạch tập mẫu, rút luật nhận dạng, mã hóa chỉ mục, sắp xếp, lưu trữ và xuất bản CSTT. Để cực tiểu hóa kích thước CSDL luật, chúng tôi chọn thuật toán CRC32 nhằm tránh xung đột giá trị băm cho 4,294,967,296 luật thành viên khác nhau. Do số luật nhận dạng luôn nhỏ hơn số mẫu dữ liệu nên CRC32 là lựa chọn kinh tế, đảm bảo các luật thành viên hoạt động tốt trên CSDL mẫu có hơn 4.2 tỷ chữ ký mã độc.

Trong giai đoạn xử lý, phần mềm được thiết kế bằng ngôn ngữ Object Pascal sử dụng trình biên dịch Delphi XE3 (2012) của Embarcadero Technologies Inc. Gói D2 Anti-virus* 2013 dành cho máy chạy hệ điều hành Windows XP Service Pack 3. Máy tính thử nghiệm sử dụng Intel CPU Core 2 Duo E7200 – 2.53 GHz; 4 GB RAM vật lý; bộ nhớ khả dụng 2.8 GB.

Quá trình thử nghiệm D2 Anti-virus* 2013 (Hình 5) sử dụng CSDL 615,880 mã độc cập nhật ngày 04-08-2013. Mã động cơ 0013.06.01. Tập dữ liệu kiểm tra gồm 8,696 tập thực thi dung lượng 105,330 MB của Windows XP. Chúng tôi thực hiện thử nghiệm 3 lần, ghi nhận số liệu, tính toán, lấy giá trị trung bình. Kết quả tốc độ quét trung bình của D2 đạt 21,651 MB/giây. Kiểm tra 1 mẫu dữ liệu 12.1MB, D2 chỉ cần 0.574977 miligiây. Thời gian kiểm tra 1MB dữ liệu là 0.04747 miligiây (Bảng 3).



Hình 5: Thử nghiệm D2 Anti-Virus 2013*

Bảng 3: Số liệu thực nghiệm D2 Anti-virus 2013*

Lần kiểm tra	Thời gian kiểm tra (giây)	Tốc độ kiểm tra (MB/giây)	Thời gian kiểm tra trung bình (miligiây)	
			1 mẫu	1 MB
1	6	17,555	0.689972	0.05696
2	5	21,066	0.574977	0.04747
3	4	26,332	0.459982	0.03798
Trung bình	5	21,651	0.574977	0.04747

4.2 Tham khảo hệ anti-virus khác

Trong thực nghiệm, chúng tôi gặp khó khăn khi so sánh tốc độ quét của D2 với các hệ AV khác. Do mỗi hệ AV sở hữu một CSDL mẫu có số chữ ký mã độc khác nhau nên không thể so sánh tốc độ hai phần mềm khác nhau về kích thước tập mẫu. Ngoài ra, cách thức phân tích dữ liệu, kỹ thuật so khớp mẫu, nhu cầu tài nguyên, xử lý luồng... của mỗi AV cũng ảnh hưởng tốc độ quét của phần mềm [7].

Để đánh giá kết quả nghiên cứu, chúng tôi khảo sát phần mềm Avira Antivirus với giả định kích thước tập mẫu ảnh hưởng nhiều đến tốc độ so với các yếu tố khác. Avira (Hình 6) là AV miễn phí của Avira GmbH & Co. KG (Germany). Thử nghiệm Avira phiên bản 10.0.0.4052 (mã động cơ 8.02.12.112; CSDL 150,000 mẫu) trong cùng điều kiện với D2 (hệ điều hành, CPU, RAM, tập dữ liệu kiểm tra...), chúng tôi thu được số liệu thực nghiệm qua 3 lần chạy Avira (Bảng 4).

Hình 6: Khảo sát hệ Avira Antivirus

Bảng 4: Khảo sát hệ Avira Antivirus

Lần kiểm tra	Thời gian kiểm tra (giây)	Tốc độ kiểm tra (MB/giây)	Thời gian kiểm tra trung bình (miligiây)	
			1 mẫu	1 MB
1	1027	102.5609	118.1003	9.75030
2	1050	100.3143	120.7452	9.96867
3	1038	101.4740	119.3652	9.854742
Trung bình	1038.333	101.4497	119.4036	9.857907

Kết quả khảo sát cho thấy D2 có số mẫu nhiều hơn Avira 4 lần nhưng thời gian quét của D2 ít hơn Avira 200 lần (số liệu so sánh tương đối, chỉ mang tính tham khảo, không dùng để đánh giá phần mềm Avira Antivirus).

5 KẾT LUẬN VÀ HƯỚNG NGHIÊN CỨU TƯƠNG LAI

Trong bối cảnh gia tăng các cuộc tấn công mạng, mã độc xuất hiện ngày càng nhiều, quét nhanh chữ ký mã độc luôn là bài toán đặt ra cho các hệ AV hiện nay. Để cải tiến tốc độ duyệt quét của các hệ AV dành cho máy tính cá nhân, chúng tôi nghiên cứu cài đặt kỹ thuật nhận dạng mã độc sử dụng cơ chế băm theo chỉ mục trên không gian phân hoạch cho hệ phần mềm D2 Anti-virus* 2013. Không chỉ giải quyết bài toán tăng tốc truy xuất tập mẫu của các AV, kỹ thuật này có thể ứng dụng để lập chỉ mục cho các hệ quản trị CSDL quan hệ kích thước dữ liệu lớn.

Trong thời gian tới, chúng tôi sẽ tiếp tục nghiên cứu phương pháp sử dụng tập luật trong các hệ quản trị CSTT làm tập chỉ mục tìm kiếm khai thác dữ liệu. Kết quả thử nghiệm chứng tỏ kỹ thuật này giúp cải thiện tốc độ cho các hệ duyệt quét mã độc, mở ra hướng giải quyết các bài toán khai thác dữ liệu, tối ưu mô hình quan hệ CSDL và CSTT cho các hệ học ngày nay.

TÀI LIỆU THAM KHẢO

1. Dantong Yu, Aidong Zhang, 2003. ClusterTree: Integration of Cluster Representation and Nearest – Neighbor Search for Large Data Sets with High Dimension. IEEE Transaction on Knowledge and Data Engineering. Vol. 15, No. 3. 1-23.
2. Hoàng Kiếm, Trương Minh Nhật Quang, 2008. Cơ chế máy học chẩn đoán virus máy tính. Tạp chí Tin học và Điều khiển học. Số 1 (2008), Tập 24, Việt Nam. 32-41.
3. Jelena Mirkovic, Peter Reiher, 2004. A taxonomy of DDoS attack and DDoS defense mechanisms. ACM SIGCOMM Computer Communication. Vol. 32 Issue 2. 39-53.
4. Joseph Rabaiotti, 2007. Counter Intrusion Software. PhD. Thesis, Computer Science, Cardiff University. 38-43.
5. Konstantin Rozinov, 2005. An Abstract Efficient Static Analysis of Executables for Detecting Malicious Behaviors. Master of Science Thesis. Brooklyn Polytechnic University. USA.
6. Maurício R. Mediano, Marco A. Casanova, Marcelo Dreux , 1994. V-Trees, A Storage Method for Long Vector Data. Proceedings of the 20th VLDB Conferenc. Santiago - Chile.
7. Ozgun Erdogan, Pei Cao, 2007. Hash-AV: Fast Virus Signature Scanning by Cache-Resident Filters. International Journal of Security and Networks. Volume 2 Issue 1/2. 50-59
8. Peter Szor, 2005. The Art of Computer Virus Research and Defense. Addison Wesley Professional Press (ISBN 0-321-30454-3)
9. Po-Ching Lin, Ying-Dar Lin, Yuan-Cheng Lai, 2011. Hybrid Algorithm of Backward Hashing And Automaton Tracking. IEEE Transactions On Computers. Vol. 60. No. 4. 594-601.
10. Truong Minh Nhat Quang, Hoang Trong Nghia, 2008. A Multi-agent Mechanism in Machine Learning Approach to Anti-virus System. The Proceedings of the 2nd Symposium on Agents and Multi-Agent Systems, KES-AMSTA, Korea. Springer

Lecture Notes in Artificial Intelligence, Vol. 4953, 743-752.

11. Trương Minh Nhật Quang, Hoàng Kiếm, Nguyễn Thanh Thủy, 2008. Ứng dụng Máy học và Hệ chuyên gia trong phân loại và nhận dạng virus máy tính. Tạp chí Công nghệ Thông tin và Truyền thông (ISSN 0866-7039). Số 19,2-2008, Việt Nam. 93-101.
12. US Patent 6763466, July 2004. Fast virus scanning. Inventor – Glover. Assignee Networks Associates Technology.
13. US Patent 6898712, May 2005. Test driver ordering. Inventor - Vignoles et al. Assignee Networks Associates Technology.
14. US Patent 6928555, Aug 2005. Method and apparatus for minimizing file scanning by anti-virus programs. Inventor - Drew. Assignee Networks Associates Technology.
15. US Patent 6952776, Oct 2005. Method and apparatus for increasing virus detection speed using a database. Inventor – Chess. Assignee International Business Machines Corporation.
16. US Patent 7036147, April 2006. System, method and computer program product for eliminating disk read time during virus scanning. Inventor – Hursey. Assignee McAfee.
17. US Patent 7043634, May 2006. Detecting malicious alteration of stored computer files. Inventor - Wolff et al. Assignee McAfee.