Tạp chí Tin học và Điều khiển học, T.14, S.2, (1998), (42-52)

CÁC CƠ CHẾ CHUẨN ĐOÁN VIRUS TIN HỌC THÔNG MINH DỰA TRÊN TRI THỨC.

NGUYỄN THANH THUỶ[1] TRƯƠNG MINH NHẬT QUANG [2]

Abstract: In this paper we shall investigate a knowledge-based diagnostic mechanism in an intelligent antivirus system.

An overall system structure will be presented. The first difficult task in the diagnostic processing is the feature extraction. We proposed some basic behavious of B-viruses and F-viruses. Then, based on these caracteristics, a careful statistics for 100 typical F-viruses and 60 B-viruses is carried out. The obtained results helped us to create a konwledge base in the form of the production rules. Two different inferent mechanisms over knowledge base are discussed. Another contribution in this paper is a proposition of binary representation of an executable program E. Some experimentation of the Interlligent Antivirus system are studied based on a virtual machine.

Tóm tắt: Trong bài này chúng tôi sẽ trình bày những cơ chế chuẩn đoán virus tin học thông minh dựa trên tri thức. Cũng giống như quá trình khám và chuẩn đoán bệnh, quá trình được đề cập ở đây bao gồm các giai đoạn: trích chọn đặc trưng, chẩn đoán và lựa chọn các giải pháp xử lý.

Để nhận biết các tác nhân lạ, điều quan trọng là đưa ra các đặc trưng cơ bản của chúng. Các nghiên cứu thực nghiệm chỉ ra rằng du được che đậy dưới bất kỳ hình thức nào, cách lây nhiễm nào, mọi virus đều phải thực hiện các hành vi cơ bản. Chúng tôi đã tiến hành trích chọn được 7 đặc trưng cơ bản đối với F-virus và 8 đặc trưng cơ bản đối với B-virus. Các đặc trưng cơ bản này có thể được chia nhỏ tiếp.

Dựa trên các đặc trưng nhận dạng, một cơ sở tri thức bao gồm khoảng 200 luật đã được tạo lập. Các luật được hình thành dựa trên các thông kê chi tiết và tỷ mỷ, thực hiện trên hàng trăm mẫu virus thông dụng. Để phục vụ cho việc chẩn đoán, mỗi chương trình thực thi sẽ phải được biểu diễn dưới dạng cây nhị phân. Các cơ chế chẩn đoán có thể là suy diễn tiến hoặc suy diễn lùi. Để giảm bớt không gian tìm kiếm, chúng tôi cũng đề xuất một số heuristics. Cuối cùng chúng tôi đã tiến hành thử nghiệm cài đặt một máy ảo, chạy cho kết quả tốt. Mô-tơ suy diễn trên cơ sở tri thức đạt tỷ lệ thành công 96% đối với B-virus và 89% đối với F-virus.

1. TÌNH HÌNH THỰC TẾ VÀ YÊU CẦU ĐẶT RA CHO PHẦN MỀM:

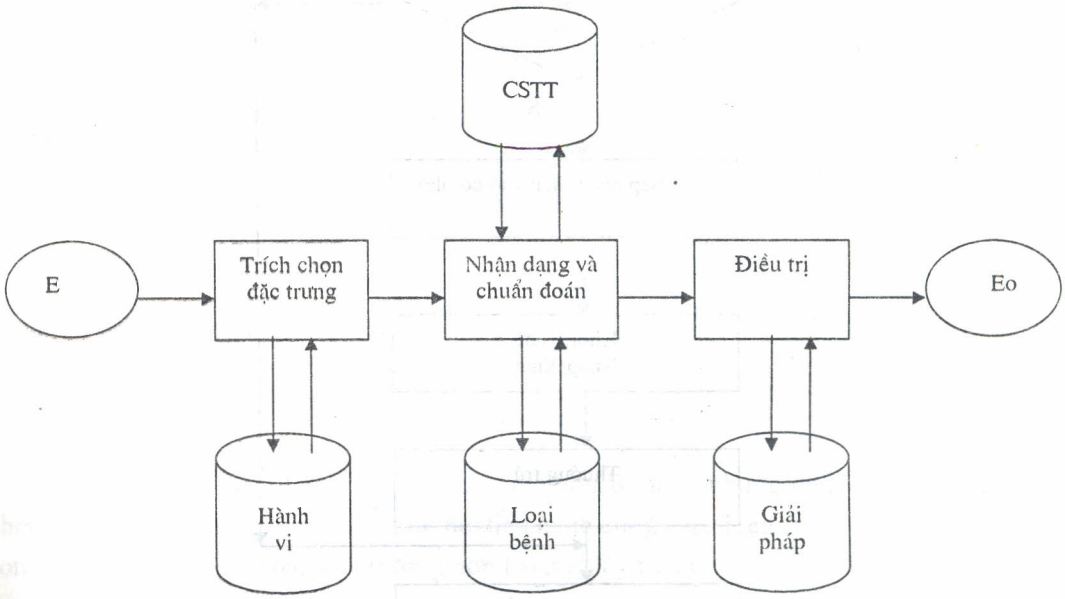
Phần lớn các phần mềm chống virus tin học (Anti Virus) hiện nay đều gặp trở ngại trong quá trình nhận dạng các virus mới. Các chương trình này chỉ có thể phát huy tác dụng của nó trên một tập các mẫu virus đã được dữ liệu hoá và cập nhật vào "ngân hàng virus" của phần mềm. Đối với các virus mới, tác dụng của Anti Virus bị vô hiệu hoàn toàn. Trong một cố gắng chừng mực nào đó, Anti Virus chỉ có thể nhận dạng sự tồn tại của các chương trình thường trú lạ, chứ không thể đưa ra một nhận xét tích cực nào về tình trạng trong sạch của các tập tin trên hệ thống được thẩm định.

Quá trình phòng chống virus tin học cho máy tính cũng giống như việc bảo vệ sức khoẻ cho con người. Trong mọi trường hợp, phương châm "Phòng bệnh hơn chữa bệnh" vẫn là tư tưởng chủ đạo. Vì vậy chỉ cần phát hiện sự có mặt của virus lạ, tiêu diệt chúng "từ trong trứng nước" là đã chặn đứng được các hiểm hoạ có nguy cơ bùng nổ trên hệ thống.

Để giải quyết yêu cầu thực tế, chương trình chống virus thông minh (Anti Virus*) sẽ sử dụng các phương pháp nhận dạng nào, các chiến lược tìm kiếm, cấu trúc chương trình... ra sao. Chúng ta sẽ xem xét các vấn đề trong phần trình bày dưới đây.

2. CƠ CHẾ CHẨN ĐOÁN:

Để nhận dạng một virus V lạ trên đối tượng E (là các Mẫu tin khởi động MTKĐ, tập tin thi hành COM, EXE, DLL...), Anti Virus* sẽ tiến hành phân tích mã lệnh thực thi của E. Do V cũng sử dụng các mã lệnh nhị phân của bộ xử lý giống như mã lệnh của E nên không thể xem xét từng mã lệnh rời rạc để có thể khẳng định về sự vô nhiễm của E. Như thế, Anti Virus* phải có khả năng "nhìn" và đánh giá một tập hợp các lệnh máy để phán đoán xem đó là hành vi của V hay chỉ là chỉ thị của E. Điều này có thể giải quyết bằng cách xây dựng một tập luật các hành vi của virus. Bộ luận này được tạo thành bởi các sự kiện nguyên tố là mã lệnh của bộ chỉ thị, được xây dựng thành các phát biểu phản ánh được những hành vi đặc trưng của virus trên đối tượng E bất kỳ. Kế tiếp cần xây dựng cơ chế suy diễn dựa vào tập luận đã có, phương pháp vận hành động cơ suy diên, và cuối cùng là đưa ra kết quả của quá trình nhận dạng, những đề nghị thích hợp, các giải pháp phục hồi dữ liệu (điều trị) khả dĩ. Một cách tổng quát, sơ đồ cấu trúc chương trình có thể được trình bày như sau:



Để tăng tính linh hoạt và thích nghi của hệ, các bộ luật (Hành vi, Loại bệnh, Giải pháp) sẽ được chứa trong các cơ sở dữ liệu có thể dự cập nhật theo "tình hình bệnh lý của bệnh nhân" được yêu cầu chẩn đoán.

3. TRÍCH CHỌN ĐẶC TRƯNG

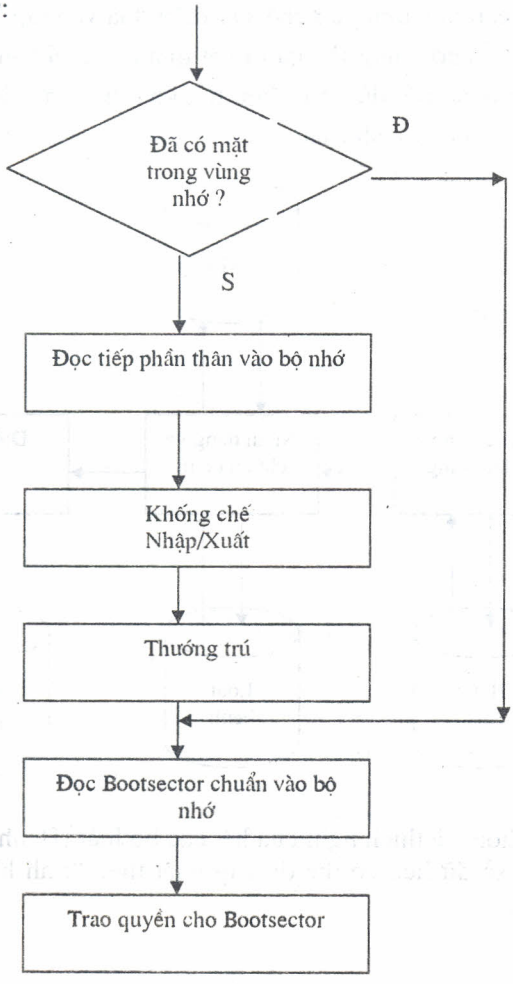
Thật khó mà đoán trước bản chất của một người nào đó nếu ta chưa bao giờ tiếp xúc với họ. Trong những trường hợp như vậy, nhờ vận dụng kho tàng kinh nghiệm dân gian, con người có thể "xem tướng" và dự đoán được phần nào tích cách của đối tượng mà mình quan tâm (ví dụ người có hành vi bất chính thường có ánh mắt không ngay thẳng, len lét, v. v...). Trở lại trường hợp của chúng ta, virus máy tính chính là tập hợp các hành vi (lây nhiễm, phá hoại...) tiềm ẩn của một người có cá tính (tạo ra virus). Anti Virus* phải dự đoán được hành vi, ý đồ của họ bằng cách dò xét các biểu hiện cơ bản của virus. Những kiến thức này thường được tích luỹ thông qua các kinh nghiệm của các chuyên gia virus trong thế giới thực. Vì vậy nếu trích chọn được các đặc trưng này, tổ chức chúng thành bộ luật hành vi với cơ chế suy diễn thích hợp thì Anti Virus* có thể "xem mặt mà bắt hình dong" các virus mới.

a. Các đặc trưng cơ bản:

Dù được che đậy dưới bất cứ hình thức nào, cách lây nhiễm nào, virus đều phải thực hiện các hành vi cơ bản. Các hành vi này giúp chúng khởi tạo trạng thái ban đầu, kiểm tra môi trường, kích hoạt các mô đun kế tiếp, thường trú, lây lan... Chúng ta sẽ dựa và nguyên tắc lấy nhiễm của chúng để phân loại và nhận dạng các hành vi cơ sở này.

i. B-virus:

Có thể chia chương trình B-virus làm hai phần: phần khởi tạo (Install) và phần thân. Do tất cả những đặc trưng cơ bản của B-virus đều nằm trọn trong phần Install, nên chúng ta chỉ phân tích sơ đồ khối của phần này:



Để thực hiện các tác vụ trên, chúng phải sử dụng một số thao tác bắt buộc. Nhiệm vụ của Anti Virus* là tổng quát hoá các tác vụ này dưới dạng các hành vi cơ bản mà chương trình có thể nhận biết. Ví dụ để thực hiện hành vi (2) - thường trú B-virus phải thực hiện các tác vụ sau:

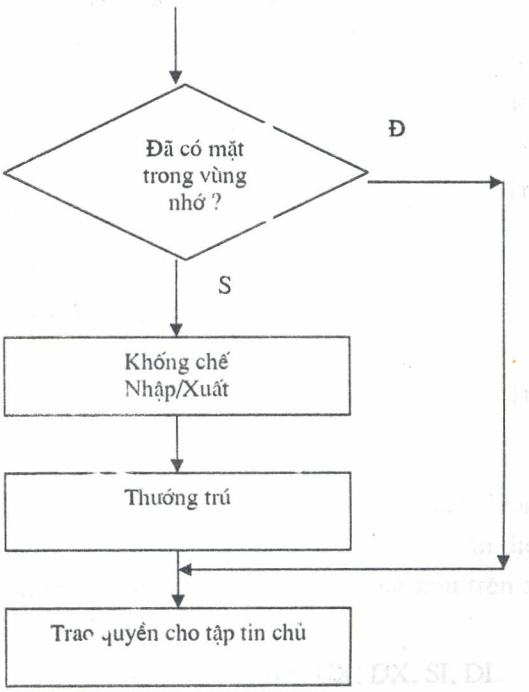
+ Lấy kích thước bộ nhớ
+ Giảm kích thước bộ nhớ môt lượng bằng kích thước progvi.
+ Đặt kích thước bộ nhớ
+ Tách chuyển một vùng nhớ khỏi quyền kiểm soát của hệ thống
+ Chuyển progvi vào vùng nhớ này

Với các phân tích như trên, chúng ta sẽ trích chọn được các đặc trưng cơ bản khác. Thật vậy, dựa vào mô hình lây nhiễm, kết hợp với kinh nghiệm thực tiễn, chúng tôi đã tổng hợp được 8 hành vi cơ bản của B-virus cho phép nhận dạng đầy đủ sự có mặt của chúng trên các MTKĐ.

ii. F-virus:

Tương tự như đối với B-virus, chúng ta cũng chia chương trình một F-virus làm hai phần: phần khởi tạo và phần thân. Sau đó tập trung phân tích phần khởi tạo để trích chọn các đặc trưng của chúng.

Sơ đồ phần khởi tạo của F-virus được mô tả như sau:



Cấu trúc phần Install của F-virus đơn giản hơn B-virus, nhưng hành vi tương ứng lại tinh vi hơn do chúng lợi dụng các dịch vụ do hệ điều hành cung cấp. Trên cơ sở đó, chúng tôi đã trích chọn được 7 đặc trưng cơ bản của trường hợp F-virus lây trên tập tin thi hành E bất kỳ.

Một điều hiển nhiên là các hành vi đặc trưng cơ bản nhận được còn mang tính tổng quát. Có thể phân rã thành các hành vi chi tiết hơn để làm mịn cơ sở tri thức, hỗ trợ quá trình nhận dạng và suy diễn sau này. Ví dụ phân rã hành vi Lấy kích thước bộ nhớ của B-virus, chúng ta nhận thấy một trong các hành vi nhỏ hơn như sau:

Lấy kích thước bộ nhớ:

+ Truy nhập vùng nhớ thấp

+ Gọi int 12h

+ ...

Trong các hành vi nhỏ này, chúng ta lại phân rã chúng thành nhỏ hơn nếu có thể, ví dụ hành vi Truy nhập vùng nhớ thấp

Truy nhập vùng nhớ thấp:

+ Nhóm lệnh chuyển giá trị (MOV, MOVSW, ...)

+ Nhóm lệnh nạp giá trị (LODSB, LDS,..)

MOV:

+[Segment]: MOV [IndexRegister], [ValueRegister]

+[Segment]: MOV [IndexRegister], [Value]

+ [Segment]: MOV [Address], [ValueRegister]

[Segment]: MOV [IndexRegister], [ValueRegister]

[Segment]

+ DS = 0

+ ES = 0

+ SS = 0

[IndexRegister]

+ BX

+ BP

+ SI

+ DI

[ValueRegister]

+ AX

+ BC

+ CX

+ DX

+ SI

+ DI

+ BP

+ SP

v. v...

b. Thống kê hành vi:

Trong thực tế, bất cứ một ứng dụng E nào cũng có thể sử dụng các thủ thuật có mã lệnh tương tự như các hành vi nói trên. Vì vậy để tránh phát hiện lầm, ta phải ước lượng tần suất sử dụng các hành vi, kết hợp các qui luật suy diễn chặt chẽ. Việc này có thể giải quết bằng phương pháp thống kê. Chúng tôi đã thực hiện các phép thống kê tren 100 mẫu F-virus và 60 B-virus,

trích chọn được các hình vi có xác suất cao, sau đó ước lượng độ ưu tiên của hành vi tương ứng. Nhờ vậy, mô tơ suy diễn của chúng ta sẽ có khả năng nhận dạng các virus phổ biến nhất, đồng thời giảm thời gian phân tích trên tập E nhiễm.

4. TẠO LẬP CƠ SỞ TRI THỨC:

[2]Để có thể khai thác hiệu quả bộ từ điển hành vi, hệ sẽ sử dụng phương pháp biểu diễn tri thức chuyên gia dưới dạng luật sản xuất:

$$r: p1 \wedge ... \wedge p_n \Rightarrow q$$

Với ngữ nghĩa:

Nếu <Hành vi p1>
Và <Hành vi p2>
...
Và <Hành vi p_n>
Thì <Hành vi/ kết luận q>

Trong đó p_i là hành vi cơ sở thứ i, q là hành vi ở mức tổng quát. Trong một số trường hợp, q là có thể là hành vi cơ sở cho các hành vi tổng quát Q nào đó. ở mức suy diễn cuối cùng, q chính là Kết luận về tình trạng của E. Một cách tổng quát, các hành vi này là sự kiện mang lại giá trị của môt biến luận lý, hoặc là kết quả của một biểu thức tính toán...

Để minh hoạ, chúng ta xem xét hành vi Lấy kích thước bộ nhớ được phát biểu như sau:

Nếu Code = 8Bh ; MOV
Và NextCode = 07; MOV AX, [BX]
Và BX = 413h; IndexRegister
Và DS = 0; Segment
Thì *Lấy kích thước bộ nhớ*

Trên đây chỉ là một minh hoạ nhỏ mô tả quá trình xây dựng luật. Trong thực tế cài đặt, bộ luật cần được phân rã chi tiết nhưng phải đảm bảo được tính vĩ mô, toàn diện. Đối chiếu cơ chế trích chọn đặc trưng với ví dụ minh hoạ quá trình xây dựng luật như trên sẽ cho thấy sự đồ sộ của bộ luật. Ví dụ, chúng ta chỉ xét sự kiện:

ValueRegister cho AX, mà chưa xét các BX, CX, DX, SI, DI...

IndexRegister cho BX, chưa xét SI, DI, BP...

Segment cho DS, chưa xét ES, SS.

Trên cơ sở đó, các luật được chia thành 2 lớp chính:

a. Lớp luật Rb mô tả hành vi của B-virut

Ví dụ:

Nếu E là MTKĐ
Và Tự kiểm tra

Và Thường trú
Và Trao quyền cho Boot sector
Thì *E chứa B-virus V*

b. Lớp Luật Rf mô tả hành vi của F-virus:

Ví dụ:

Nếu E là tập tin thực thi
Và Tự kiểm tra
Và Thường trú
Và Trao quyền cho File
Thì *E chứa F-virus V*

Trong mỗi lớp luật chính có thể có nhiều lớp luật con tương ứng với các hành vi cơ sở. Ví dụ trong lớp luật Rb có thể chia làm 8 lớp luật con, lớp luật Rf chia làm 7 lớp luật con. Mỗi luật con lại được chia thành các luật nhỏ hơn, v. v...

5. MÔ - TƠ SUY DIỄN:

Tất cả các chương trình thi hành trên máy PC (dùng bộ xử lý 8088, 80x86, Pentium) dù được viết bằng ngôn ngữ nào, cũng được trình biên dịch của nó dịch thành các tập tin thực thi chứa các chỉ thị máy của bộ xử lý. Lợi dụng đặc điểm này, người viết virus sử dụng các trình biên dịch Assembly để thiết kế virus và tìm cách đính nó vào các chương trình thực thi. Vì thế chúng luôn "hoà hợp" với các chương trình thực thi khác mà không phụ thuộc vào ngôn ngữ ban đầu của chủ thể. Như vậy để phát hiện một hành vi (bao gồm nhiều chỉ thị theo một cấu trúc nào đó), mô tơ phải thao tác trên tập hợp các mã lệnh của bộ xử lý. Có thể xem xét quá trình thực hiện chương trình trên máy PC chỉ chứa hai loại lệnh cơ bản:

(1) Các lệnh tuần tự.

Ví du: + Các lệnh thay đổi giá trị thanh ghi, biến, thao tác ngăn xếp...

+ Các phép toán số học, luận lý.

+ Các lệnh gọi chương trình con bình thường.

+ ...

(2) Các lệnh rẽ nhánh

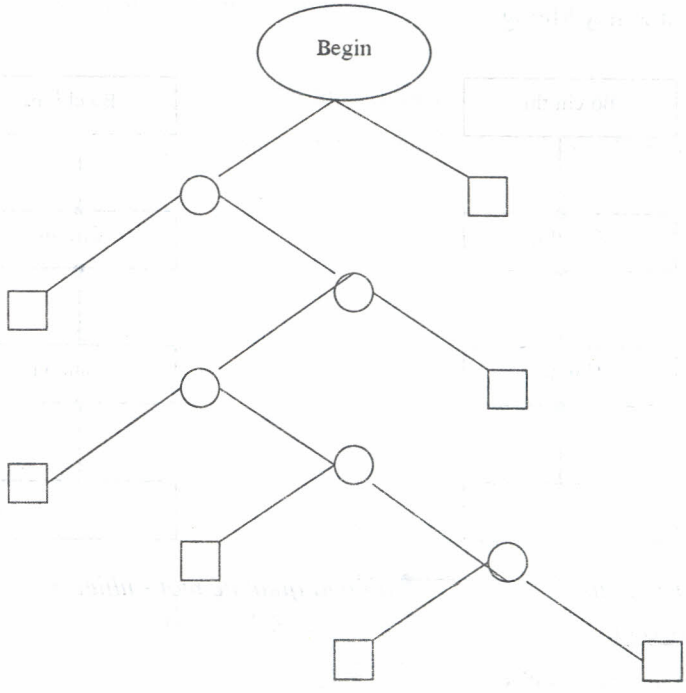
Ví dụ: + Các lệnh nhảy có điều kiện, không điều kiện, gần hoặc xa.

+ Các vòng lặp.

+ ....

Lưu ý rằng với cách phân loại trên, các lệnh nhảy là các lệnh rẽ hướng chương trình theo một trong hai hướng thích hợp tuỳ theo điều kiện. Lệnh Call được xem là lệnh tuần tự, vì nó không làm thay đổi "mạch" của chương trình. Ngược lại, các lệnh vòng lặp không được xem là

tuần tự vì bản thân chúng thường chứa các lệnh nhảy có điều kiện. Vì vậy, một cách hình thức có thể biểu diễn mô hình xử lý lệnh của bộ xử lý đối với chương trình E như một cây nhị phân.



Cây chỉ thị nhị phân C của chương trình E

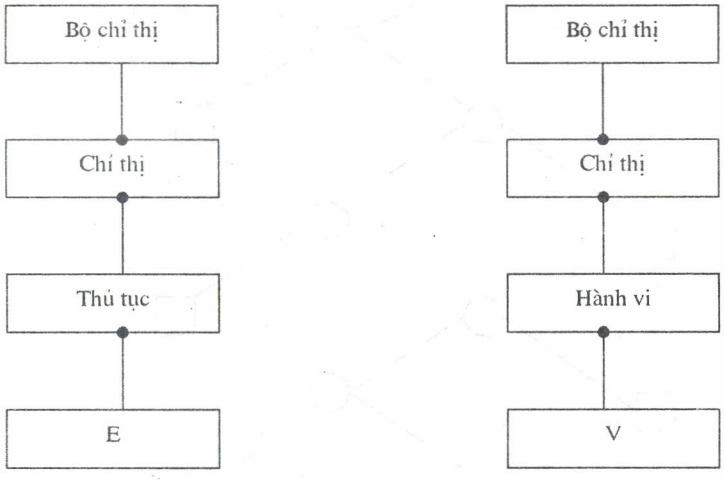
Trong đó:

* Gốc biểu diễn điểm vào lệnh đầu tiên.
* Các nhánh biển diễn các lệnh tuần tự.
* Các nút biểu diễn một lệnh nhảy.
* Nút là đánh dấu điểm kết thúc của chương trình.

Với cách biểu diễn này, môtơ suy diễn sẽ dễ dàng áp dụng các chiến lược tìm kiếm trên không gian trạng thái đã được đồ thị hoá. Để giải quyết bài toán nhận dạng virus, chúng ta cũng xây dựng một không gian trạng thái đối với cây chỉ thị nhị phân cho V, nhưng các nút là của cây V chính là điểm dừng khi môtơ đã đạt đến số nút tới hạn, hoặc đạt được kết quả nhờ quá trình tổng hợp hành vi (được ghi nhận từ quá trình tìm kiếm ở các nút cha) cho phép khẳng định sự có mặt của virus. Trong một số trường hợp, các giải thuật quay lui, heuristics sẽ được ứng dụng để rút ngắn quá trình tìm kiếm. Ví dụ nếu không gian trạng thái không lớn, có thể gia tăng giá trị số nút tới hạn để mở rộng miền tìm kiếm. Như vậy mô tơ suy diễn của chúng ta phải có khả năng nhận dạng từng chỉ thị, độ dài chỉ thị (để tính địa chỉ, nội dung của chỉ thị kế), biết cách tính địa chỉ tương đối của các lệnh nhảy (để xác định các nút).

Quá trình suy diễn của hệ bao gồm suy diễn tiến (forward chaining) và suy diễn lùi (backward chaining).

Xét mô hình thực tế (e) và (v), ta thấy chúng có cấu trúc giống nhau, trong đó thực thể Thủ tục có vai trò tương đương với thực thể Hành vi. Mặt khác V ⊆ E, nên vấn đề chỉ còn giới hạn vào việc đánh giá các hành vi trong từng thực thể Thủ tục của E xem có phù hợp với hành vi của V trong bộ luật R hay không.



(Ký hiệu ——● diễn tả quan hệ một - nhiều)

Bài toán suy diễn:

Vào: + Tập luật sản xuất R

+ Tập các sự kiện đã biết GT về đối tượng E

+ Tập mã lệnh C của đối tượng E.

Ra: + Kết luận về sự tồn tại của V trên E.

+ Biện pháp khắc phục.

Yêu cầu: Sử dụng tập luật R, các yếu tố của E để xác định V, đề xuất giải pháp cụ thể.

Quá trình suy diễn tiến được thực hiện như sau:

Ban đầu biết E (dưới dạng cây chỉ thị nhị phân C) cùng các đặc điểm GT. Sau đó dựa vào các luật thoả mãn, các sự kiện bổ sung, thực hiện giải thuật tìm kiếm cho đến khi V được phát hiện, hoặc giá trị số nút tới hạn bị vi phạm, hoặc gặp nút lá.

Giải thuật:

(1) + Khởi tạo các giá trị đầu:

. Có nút đếm Count = 0

. Hằng tới hạn Limite.

. Biến trạng thái Detect = false, EndTrace = false.

. Xác định luật R (Rf hay Rb) cùng m, n tương ứng. Trong đó:

- n là tổng các luật nguyên tố của luật cơ sở r.

- m là tổng các nguyên tố của bộ luật Rx.

(có thể định các đại lượng này một cách đệ qui).

. Ngăn xếp trạng thái Trace (v) = Null.

(Trace(v) chứa các hành vi đã phát hiện trong quá trình).

+ Mở nút đầu tiên.

+ Khởi tạo hàm sự kiện Fact(n), chọn nhánh rẽ hướng.

(2) + Đọc mã lệnh hiện tại, mã lệnh kế tiếp trên nhánh vào các biến Code, NextCode.

+ Duyệt từng mã lệnh (Code, NextCode), cho đến khi:

. Phát hiện hành vi R(pij), i = 1 ÷ m, j = 1 ÷ n, cập nhật vào Trace(v).

. Gặp nút kế tiếp.

. Gặp nút lá, bật cờ trạng thái EndTrace = true.

+ Kiểm tra ngăn xếp trạng thái Trace(v), nếu thoả (đã đạt đủ hành vi cho phép kết luận), bật cờ trạng thái Detect = true, EndTrace = true, đến bước (3).

+ Hàm Fact(n) ghi nhận các sự kiện hiện tại, định vị địa chỉ, lấy kết quả mở nút kế tiếp.

+ Tăng số nút đếm Count = Count + 1. Nếu Count <Limite, trở về bước (2).

(3). + Nếu Detect = true:

. Phân tích tập hành vi R(p) trong ngăn xếp trạng thái Trace(n).

. Dựa và R(p) và GT, lựa chọn giải pháp khắc phục.

+ Thoát

Giải thuật được xây dựng theo phương pháp tìm kiếm với tri thức bổ sung, do đó hạn chế được miền tìm kiếm. Vì vậy ta không cần phải vét cạn toàn bộ cây nhị phân chỉ thị mà vẫn bảo đảm không bỏ sót miền khả dĩ trên cây.

Có thể áp dụng cơ chế suy diễn lùi trong từng bước, từng trường hợp cụ thể để tìm cách thay việc chứng minh q (trong luật r: $p1 \wedge .... \wedge p_n \Rightarrow q$) bằng các p1, p2, pn.. Ví dụ hành vi Đặt kích thước bộ nhớ của B-virus (trong bộ luật Rb) được phát hiện, lúc đó cần kiểm chứng:

- Một trong các thanh ghi phân đoạn ES, DS, SS có giá trị 0 (hoặc 40h)
- Một trong các thanh ghi chung, thanh ghi chỉ số..., có chứa giá trị MenSize mới.
- Đã dùng lệnh chuyển (MOV, MOVSB, MOVSW), hoặc lệnh lưu giá trị (STOSB, SOTSW).

6. THỬ NGHIỆM:

Việc tổ chức không gian trạng thái cần được tiến hành trước khi tải tập chỉ thị E vào. Trường hợp E là các tập tin thực thi, Anti Virus* chỉ cần bố trí vùng nhớ cho E như một quá trình con và trao quyền cho mô tơ vận hành. Tuy nhiên, do tính phong phú của mã lệnh và kích thước của E là bất kỳ nên cần giới hạn phạm vi tìm kiếm bằng cách ước lượng số nút tới hạn. Ngược lại, do kích thước MTKĐ chỉ giới hạn trong 512 byte nên việc định giá trị số nút tới hạn cho các trường hợp này là không cần thiết, nhưng việc tổ chức không gian trạng thái lại phức tạp

vì đây chính là vùng địa chỉ thấp, nơi MTKĐ được nạp vào đầu tiên trong quá trình khởi động. Qua nghiên cứu các phiên bản của hệ điều hành MSDOS, PCDOS, WINDOW3.x, WINDOW95, cũng như các hệ thống được trang bị các trình điều khiển thiết bị (ĐKTB), các trình thường trú popup, các timer..., ta thấy vùng nhớ thấp này thường được sử dụng triệt để. Vì vậy khả năng xung đột sẽ rất cao nếu Anti Virus* sử dụng trực tiếp vùng nhớ này, lúc đó mô tơ suy diễn rất khó vận hành vì có thể bị tác động bởi các trình thường trú khác. Mặt khác Anti Virus* phải đảm bảo tính tương thích cao trên tất cả các máy PC, trên các phiên bản của hệ điều hành, cũng như với môi trường tại thời điểm chạy, sao cho việc nhận dạng V không làm ảnh hưởng hệ thống (ví dụ như phá huỷ khối MTKĐ của hệ điều hành, tê liệt các trình thường trú, timer rối loạn....)

Để giải quyết vấn đề trên, có thể định nghĩa và cài đặt một máy ảo (VM Virtual Machine). Kỹ thuật này thường được các trình biên dịch áp dụng nhằm tăng độ tương thích của ngôn ngữ lập trình cho các ngôn ngữ lập trình cho các máy có môi trường và bộ chỉ thị khác nhau. Máy ảo này sẽ có vùng nhớ, timer, bộ chỉ thị, các thanh ghi .... riêng. Sau khi cài đặt xong, MTKĐ sẽ được tải vào vào VM, riêng mô tơ suy diễn của Anti Virus* vẫn chạy trên máy thực, đóng vai trò superviser giám sát hành vi, phân tích, phán đoán và kết luận về tính trung thực của MTKĐ trên VM, không ảnh hưởng đến mọi hoạt động cơ sở đang chạy ngầm ở mức hệ điều hành. Nhờ vậy, mô tơ suy diễn nhận dạng B-virus đạt tỷ lệ thanh công khá cao (96% so với tỷ lệ 89% của mô tơ nhận dạng F-virus).

TÀI LIỆU THAM KHẢO

1. The Programmer`s Technical Reference: MS-DOS, IBM PC & Compatibles. Dave Williams, Tech Publications Pte Ltd-Sigma Press, England 1993.

2. Bên trong máy vi tính IBM - PC. Peter Norton, bản dịch của NXB Thông kê 1989.

3. Các nguyên tắc phòng chuống virus tin học. ***Trương Minh Nhật Quang****, Tin học và Đời sống, số 5 - 6/1994.*

4. Hỏi đáp về virus tin học. ***Trương Minh Nhật Quang****. Tin học và Đời sống, số 9-10/1994.*

5. The Computer Virus HandBook. ***Richard B. Levin****, Osborne/McGraw-Hill 1990.*

6. Virus tin học, huyền thoại và thực tế. Ngô Anh Vũ, NXB Thành Phố Hồ Chí Minh - 1991.

7. Thu thập trí thức trong các hệ chuyên gia ứng dụng. ***Nguyễn Thanh Thuỷ****, Hội nghị khoa học viện Công nghệ Thông tin, Đại học Bách Khoa Hà Nội, 2 - 1996.*

8. Systè me d`expert, Techniques et Application.* ***Nguyễn Thanh Thuỷ****, Institut Francophone d`Informatique, 1996.

9. Các giải pháp cho phần mềm chống virus thông minh. ***Trương Minh Nhật Quang, Nguyễn Thanh Thuỷ****, Tạp Chí Tin Học và Điều Khiển học, 1997.*