CÁC KỸ THUẬT NGỤY TRANG CỦA VIRUS ĐA HÌNH

Phạm Thanh Bình
Khoa Công nghệ Thông tin - Đại học Thuỷ lợi. Email: binhpt@wru.edu.vn

1. GIỚI THIỆU CHUNG

Để không bị phần mềm diệt virus phát hiện, các virus phải có khả năng nguỵ trang, dấu mình dưới nhiều hình thức khác nhau. Trong khuôn khổ bài viết này, trình bày về các kỹ thuật nguỵ trang của virus đa hình - một loại virus đang ngày càng phát triển mạnh mẽ vì nó có khả năng thay đổi chính bản thân nó, và rất khó bị phát hiện - để từ đó đưa ra được các giải pháp tìm và diệt tương ứng.

2. PHƯƠNG PHÁP NGHIÊN CỨU

Việc nghiên cứu các kỹ thuật nguỵ trang của virus đa hình được dựa trên các tài liệu về mã hoá - bảo mật truyền thống, và dựa trên việc phân tích các mẫu virus thu được trên thực tế.

3. KẾT QUẢ NGHIÊN CỨU

Trong phần này chúng tôi xin trình bày về các khái niệm, các kỹ thuật liên quan tới việc viết và diệt virus đa hình (tập trung vào các kỹ thuật nguỵ trang) mà chúng tôi đã đúc kết lại trong quá trình nghiên cứu.

3.1. Virus đa hình là gì?

Virus đa hình (polymorphic virus) khác với các virus máy tính thông thường ở chỗ nó có khả năng tự biến đổi bản thân thành nhiều dạng khác nhau. Do virus đa hình có thể liên tục biến đổi mã lệnh, nên thật khó để tìm được đoạn mã đặc trưng của nó, và do đó rất khó phát hiện ra nó bằng phương pháp so sánh mã truyền thống.

3.2. Các kỹ thuật nguỵ trang của virus đa hình

Để có thể tự biến đổi mã lệnh, các virus đa hình áp dụng rất nhiều kỹ thuật khác nhau. Dưới đây là một số kỹ thuật thường gặp:

a) Mã hoá

Một virus đa hình có thể mã hoá chính mã lệnh của nó bằng cách sử dụng một giải thuật mã hoá nào đó. Giải thuật mã hoá thường khá đơn giản vì mục đích của quá trình này không phải để giữ bí mật mà là để biến đổi mã lệnh (giải thuật càng đơn giản thì tốc độ mã hoá càng nhanh). Để tạo ra nhiều bản sao khác nhau, virus phải mã hoá chính nó bằng các khoá khác nhau khi lây vào các file khác nhau. Một kỹ thuật thường được các virus sử dụng là sử dụng phép toán XOR, phép toán này có đặc điểm là (A XOR K) XOR K = A, tức là sau hai lần XOR với cùng một khoá K thì A sẽ trở về giá trị ban đầu. Trước khi tạo ra một bản sao mới, virus sẽ tạo ra một khoá K ngẫu nhiên (có thể dựa vào thời gian của hệ thống khi đó). Sau đó nó sẽ XOR mã nguồn của nó với khoá này, lần lượt từng byte một để tạo ra một virus được mã hoá nằm trong file bị nhiễm. Khoá K cũng sẽ được cất trong file. Vì khoá là ngẫu nhiên nên các bản sao sinh ra tại các thời điểm khác nhau sẽ khác nhau. Khi muốn chạy, virus sẽ phải giải mã chính nó, nên cần có hàm giải mã cất trong file (hàm giải mã có thể thực hiện XOR mã virus với khoá K lần thứ hai để thu được mã nguồn ban đầu).

Nhược điểm của giải pháp này là không được phép mã hoá hàm giải mã. Vì nếu mã hoá toàn bộ mã nguồn virus, bao gồm cả hàm giải mã, thì virus sẽ không thể chạy được do không khôi phục được mã nguồn ban đầu. Nhưng nếu không mã hoá hàm giải mã thì chương trình diệt virus có thể nhận dạng hàm giải mã, và phát hiện ra virus!

b) Hoán vị

Một kỹ thuật khác để tạo ra các bản sao virus mới là hoán vị ngẫu nhiên các modul trong mã lệnh virus. Giả sử một virus có N modul khác nhau thì số lượng phiên bản virus có thể sinh ra là N! (ví dụ 10! = 3628800, một con số rất lớn!). Để có thể đổi chỗ các modul mà không làm thay đổi tác dụng của chúng, mã lệnh virus phải có khả năng định vị lại chương trình trong bộ nhớ, hoạt động của các modul phải độc lập với vị trí, và sử dụng các địa chỉ tương đối thay thế cho địa chỉ tuyệt đối.

Tuy nhiên, nếu chỉ áp dụng đơn thuần phương pháp hoán vị thì việc phát hiện ra virus không phải là điều quá khó khăn, cho dù số lượng phiên bản virus có thể rất lớn. Chương trình diệt chỉ cần áp dụng phương pháp so sánh mã tuần tự, tức là lần lượt so sánh đoạn mã đặc trưng của virus với từng vị trí trong file thi hành, mỗi vị trí cách nhau 1 byte. Vì virus chỉ thay đổi trật tự các modul, chứ không làm thay đổi mã lệnh trong modul, nên không sớm thì muộn chương trình quét sẽ tìm ra đoạn mã đặc trưng trong file (nếu file đã bị nhiễm virus).

c) Thay đổi mã lệnh

Để khắc phục nhược điểm của kỹ thuật hoán vị, những người viết virus đã đưa ra một giải pháp khác,

đó là tìm cách thay đổi mã lệnh virus. Nguyên tắc cơ bản là phải biến đoạn mã ban đầu thành một đoạn mã khác, mà công dụng của nó vẫn giữ nguyên. Có nhiều cách để làm được điều đó như chèn lệnh, thay đổi thanh ghi, thêm biến, thay thế lệnh…

• *Chèn lệnh*

Để minh hoạ cho kỹ thuật này, chúng ta sẽ lấy đoạn mã sau đây làm ví dụ:

```
MOV AX, A
ADD AX, B
ADD AX, C
SUB AX, D
MOV E, AX
```

Đoạn mã này có tác dụng thực hiện phép toán E = A + B + C – D. Ta có thể biến đổi đoạn mã này mà không làm thay đổi tác dụng của nó bằng cách chèn thêm vào giữa các lệnh nói trên một số lệnh "vô nghĩa", ví dụ:

```
MOV AX, A
NOP
ADD AX, B
NOP
ADD AX, C
NOP
SUB AX, D
NOP
MOV E, AX
```

NOP là một lệnh không làm gì cả, và việc chèn nó vào giữa các lệnh sẽ không ảnh hưởng gì tới kết quả hoạt động của đoạn mã. Đoạn mã nhị phân mới sau khi chèn trông sẽ rất khác so với đoạn mã cũ. Tất nhiên đây chỉ là một ví dụ đơn giản, vì việc xuất hiện các lệnh NOP liên tiếp, xen kẽ như vậy sẽ khiến chương trình diệt virus nghi ngờ, và có thể trở thành đầu mối để phát hiện virus. Trên thực tế người ta có thể sử dụng nhiều kỹ thuật chèn khác nhau như chèn một lệnh, chèn hai lệnh, hoặc nhiều hơn… Các lệnh cũng rất đa dạng về chủng loại, có thể là lệnh logic, lệnh số học, lệnh nhảy… miễn sao thoả mãn điều khiện là không làm thay đổi tác dụng của đoạn lệnh ban đầu.

• *Thay đổi thanh ghi*

Các phiên bản virus khác nhau có thể sử dụng các thanh ghi khác nhau cho cùng một mục đích. Ví dụ đoạn mã ở trên sử dụng thanh ghi AX làm trung gian cho việc thực hiện phép toán E = A + B + C – D, có thể thay thế thanh ghi AX bằng các thanh ghi khác (như BX, CX…) mà không ảnh hưởng gì tới kết quả:

```
PUSH BX
MOV BX, A
ADD BX, B
ADD BX, C
SUB BX, D
MOV E, BX
POP BX
```

Lệnh PUSH BX ở đầu đoạn lệnh và lệnh POP BX ở cuối đoạn lệnh nhằm bảo toàn giá trị cho BX khi sử dụng BX thay thế cho AX ở đoạn mã trên. Có thể áp dụng tương tự cho các thanh ghi khác.

• *Thêm biến*

Người viết virus có thể chuẩn bị sẵn một loạt các biến dùng để thêm vào mã lệnh và thay đổi luân phiên nhau, khiến cho quá trình phân tích mã lệnh bị rối, bị đánh lạc hướng. Ví dụ ta có thể thêm vào đoạn lệnh trên các biến X, Y, Z như sau:

```
MOV AX, A
ADD X, AX
SUB Y, BX
ADD AX, B
ADD AX, C
SUB AX, D
MOV E, AX
XOR Z, CX
```

Thực chất tất cả các lệnh liên quan tới các biến X, Y, Z thêm vào đều vô giá trị. Do việc khai báo biến tương đối thoải mái, số lượng biến có thể rất lớn, việc thêm biến và sử dụng chúng luân phiên nhau sẽ tạo ra nhiều phiên bản mã lệnh khác nhau.

• *Thay thế lệnh*

Đôi khi có thể thay thế một lệnh trong đoạn mã bằng một hoặc vài lệnh khác tương đương. Ví dụ:

```
;MOV AX, A
XOR AX, AX
ADD AX, A
ADD AX, B
ADD AX, C
SUB AX, D
MOV E, AX
```

Hai lệnh đầu tiên có tác dụng tương đương với lệnh MOVAX, A. Lệnh thứ nhất để xoá AX về 0 và lệnh thứ hai sẽ cộng A vào AX. Như vậy thanh ghi AX sẽ nhận giá trị bằng A mà không cần dùng tới lệnh MOV. Tương tự như vậy, có thể áp dụng giải pháp này cho hầu hết các lệnh MOV và các lệnh khác nữa.

Có rất nhiều kỹ thuật khác nhau để thay đổi mã lệnh, tuỳ thuộc vào trình độ của người viết virus và độ phức tạp của chương trình. Cũng có thể phối hợp nhiều kỹ thuật với nhau nhằm tạo ra được nhiều phiên bản virus hơn. Mục đích chung của những người viết virus đa hình là càng sinh ra được nhiều phiên bản mã lệnh mới thì càng tốt, càng có vẻ ngẫu nhiên thì càng tốt.

d) Kết hợp nhiều kỹ thuật nguỵ trang

Mỗi kỹ thuật nguỵ trang nói trên đều có ưu và nhược điểm của riêng nó, nên việc chỉ áp dụng đơn thuần một kỹ thuật sẽ không đem lại hiệu quả cao. Ví dụ kỹ thuật mã hoá với khoá ngẫu nhiên sẽ cho phép sinh ra các mã nhị phân gần như ngẫu nhiên, nhưng lại để lộ hàm giải mã, còn kỹ thuật thay đổi mã lệnh có thể che dấu được toàn bộ mã lệnh virus, nhưng độ ngẫu nhiên lại chưa cao, và đôi khi vẫn có thể tìm ra quy luật. Việc kết hợp cả hai kỹ thuật này sẽ là một ý tưởng tuyệt với: Mã hoá hầu hết mã virus bằng khoá ngẫu nhiên (trừ hàm giải mã), và áp dụng kỹ thuật thay đổi mã lệnh cho hàm giải mã. Như vậy

sẽ tận dụng được ưu điểm của cả hai kỹ thuật: toàn bộ mã virus sẽ được che dấu, và các phiên bản virus sinh ra sẽ gần như ngẫu nhiên.

Thêm nữa, nếu áp dụng kỹ thuật hoán vị để thay đổi trật tự của các modul virus trước khi mã hoá thì sẽ tạo ra rất nhiều khó khăn cho những người phân tích mã virus. Thời gian phân tích mã sẽ phải kéo dài hơn và virus sẽ có thêm nhiều thời gian để lây lan rộng hơn.

3.3. Đối phó với virus đa hình

Đặc trưng của virus đa hình là khả năng nguỵ trang khéo léo nhằm tránh bị phát hiện, nên việc đối phó với virus đa hình thực chất là cuộc chiến của những người viết chương trình diệt virus với những người viết virus đa hình. Người viết virus luôn tìm cách cải tạo mã lệnh sao cho virus ngày càng hoàn chỉnh, khả năng biến đổi ngày càng đa dạng. Còn người viết chương trình diệt virus thì luôn tìm cách để phân tích mã virus một cách hiệu quả và nhanh chóng nhất. Dưới đây là những công việc mà người diệt virus thường làm để đối phó với virus đa hình:

a) Phân tích mã

Công việc đầu tiên là phải tách được mã virus khỏi các thành phần khác của file thi hành. Điều đó có thể được thực hiện bằng cách dịch ngược file thi hành ra mã assembly, rồi đọc và phân tích các mã lệnh xem đoạn mã virus nằm ở đâu, sau đó tách riêng mã lệnh virus ra để phân tích. Một phương pháp khác đơn giản hơn là kích hoạt virus, để virus lây vào một file thi hành đã chuẩn bị trước. File này có đặc điểm là gần như trống rỗng, khi bị virus lây vào ta sẽ thu được toàn bộ mã lệnh virus mà không phải mất công bóc tách. Ví dụ chương trình dưới đây khi được dịch ra file exe sẽ tạo ra một file thi hành không làm gì cả, nội dung của nó cũng chẳng có gì ngoại trừ phần header của file:

```
void main()
{ }
```

Sau khi tách riêng được mã lệnh virus, người viết chương trình diệt virus phải tiến hành phân tích mã lệnh dưới dạng assembly một cách cẩn thận, nhằm biết được cách thức hoạt động của virus và các kỹ thuật lây lan, nguỵ trang, phá hoại của virus… Từ đó người diệt virus sẽ lựa chọn đoạn mã đặc trưng của virus, rồi cất đoạn mã đó vào cơ sở dữ liệu của chương trình diệt, làm cơ sở cho sự so sánh mã để phát hiện virus.

Tuy nhiên, đối với virus đa hình thì quá trình nêu trên phải khác đi một chút. Do không phải lúc nào cũng tìm được đoạn mã đặc trưng của virus, nên người diệt virus sẽ phải tìm kiếm những thứ khác để thay thế. Ví dụ, khi phân tích kỹ thuật thay đổi mã lệnh của virus, người diệt virus cần cố gắng tìm kiếm các quy luật chèn lệnh được lặp đi lặp lại (nếu có), như trường hợp chèn lệnh NOP đã trình bày ở phần trước, để từ đó đưa ra giải pháp phù hợp. Cũng có khi may mắn tìm được những đoạn mã lệnh, mà vì một lý do nào đó, khiến kỹ thuật thay đổi mã lệnh không có được tính ngẫu nhiên cao, và người diệt virus sẽ dễ dàng tìm ra quy luật.

b) Xây dựng cơ sở dữ liệu

Sau khi phân tích thành công mã lệnh virus, công việc tiếp theo là phải tiến hành xây dựng cơ sở dữ liệu cho chương trình diệt virus. Trước đây cơ sở dữ liệu này thường là một cấu trúc (struct), mỗi bản ghi của cấu trúc bao gồm tên virus, đoạn mã đặc trưng, vị trí xuất hiện của đoạn mã trong file thi hành…

Tuy nhiên, để đối phó với virus đa hình người ta cần sử dụng class chứ không phải struct. Mỗi thành phần của class, ngoài những trường nêu trên, cần có thêm các phương thức cụ thể để đối phó với từng virus cụ thể (vì có thể không tìm được đoạn mã đặc trưng của virus).

c) Viết chương trình diệt virus

Những khó khăn chính đã được giải quyết ở hai bước trên, bây giờ người viết chương trình chỉ cần hoàn thiện mã lệnh của mình để chương trình có được tốc độ quét nhanh nhất, các phương thức tìm và diệt virus phải hoạt động chính xác để không tìm nhầm hoặc gây lỗi file sau khi diệt. Cần chú ý là các phương thức này phải có khả năng phân tích mã (chứ không phải chỉ biết so sánh mã một cách thụ động) để tìm kiếm trên file thi hành những đoạn mã có quy luật giống với quy luật mà bước phân tích mã bằng tay đã tìm ra.

4. KẾT LUẬN

Do các kỹ thuật nguỵ trang của virus đa hình rất đa dạng và phong phú, nên việc phát hiện ra virus đa hình là một vấn đề khó khăn và đòi hỏi nhiều thời gian. Việc nghiên cứu về các kỹ thuật nguỵ trang này sẽ rất hữu ích đối với các sinh viên ngành Công nghệ thông tin và những người ham thích lập trình hệ thống nói chung. Các kỹ thuật đó luôn được thay đổi và cập nhật mới, nên việc nghiên cứu chúng sẽ giúp sinh viên học hỏi được rất nhiều điều, để từ đó đóng góp một phần công sức vào "mặt trận chống virus", nơi đang ngày càng trở nên nóng bỏng hơn.

TÀI LIỆU THAM KHẢO

[1]. Andrew S. Tanenbaum, 2002, Modern Operating Systems, Prentice Hall.
[2]. D.S. Stinson, 1995, Cryptography - Theory and practice, CRC Press.
[3]. Charles P.Pf Leeger, 1989, Security in Computing, Prentice Hall.
[4]. Ngô Anh Vũ, 1991, Virus tin học - Huyền thoại và thực tế, NXB TP Hồ Chí Minh.