BÁO CÁO ĐỀ TÀI NGHIÊN CỨU VỀ AN NINH

A22852. Nguyễn Đức Thắng
A24126. Trần Mạnh Quỳnh
A22896. Nguyễn Trung Thành
A22758. Nguyễn Minh Trí
A23445. Đỗ Anh Tuấn

Khoa Toán-Tin, Ngành Khoa học máy tính, Đại học Thăng Long

Email: mqtent@gmail.com

Tóm tắt: *Báo cáo này đề cập đến các vấn đề an ninh, an toàn thông tin. Thảo luận về các mối đe dọa an ninh và tấn công. Đặc biêt, chúng tôi sẽ giải thích các nguyên tắc cơ bản trong mã hóa, chứng thực dữ liệu trong báo cáo này.*

Từ khóa: Breach of confidentiality, Breach of integrity, Breach of availability, Theft of service, Denial of service, Trojan Horse, Spyware, Trap Door, Logic Bomb, Stack and Buffer Overflow, Viruses, Cryptography, Symmetric Encryption, block cipher, DES, triple DES, AES, stream cipher, RC4 Asymmetric Encryption, RSA, hash function, MD5, andSHA-1, SSL.

1. Giới thiệu

Với sự bùng nổ của ngành công nghệ thông tin thì vấn đề bảo mật và an ninh đã trở nên quan trọng và đặc biệt cần thiết trên bất cứ hệ thống máy tính nào. Bài báo cáo này sẽ giúp người đọc có thể hiểu rõ hơn về những vấn đề bảo mật và an ninh hiện nay đồng thời là những phương thức bảo mật được dùng hiện nay cũng sẽ được đề cập đến.

Những vấn đề bài báo này bàn luận đến:

- Thảo luận về các mối đe dọa và tấn công;
- Giải thích các nguyên tắc cơ bản trong mã hóa, chứng thực, hàm băm.

2. Các vấn đề về an ninh – The security problem.

Trong rất nhiều ứng dụng, việc đảm bảo an toàn cho hệ thống máy tính là rất cần thiết. Bởi các hệ thống thương mại lớn, ngân hang, tài chính, những hệ thống chứa dữ liệu quan trọng,… chính là mục tiêu lôi cuốn của rất nhiều kẻ tấn công. Việc mất mát dữ liệu như vậy, cho dù là do tai nạn hoặc gian lận, có thể làm suy giảm nghiêm trọng đến hoạt động của công ty. Dưới đây là danh sách các hình thức hành vi vi phạm bảo mật:

1. Vi phạm về bảo mật - Breach of confidentiality: Là loại vi phạm liên quan đến việc truy cập trái phép nhằm đánh cắp thông tin.

2. Vi phạm về toàn vẹn dữ liệu – Breach of integrity: Là loại vi phạm làm thay đổi trái phép bất hợp pháp về dữ liệu.
3. Vi phạm về sẵn dùng – Breach of availabity: Là loại vi phạm phá hủy dữ liệu trái phép.
4. Vi phạm về ăn cắp dịch vụ - Theft of servive: Là loại vi phạm sử dụng trái phép tài nguyên.
5. Từ chối dịch vụ - Denial of service: Là loại vi phạm liên quan đến việc ngăn ngừa sử dụng hợp pháp vào hệ thống.

Những kẻ tấn công sử dụng một số phương pháp để xâm phạp an ninh. Phổ biến nhất là giả mạo, giả danh truy cập bất hợp pháp vào hệ thống nhằm mục đánh cắp thông tin một cách bất hợp pháp. Một loại tấn công *kẻ đứng giữa* "man-in-the-middle", trong đó kẻ tấn công đứng ở giữa đường truyền giả mạo là người gửi gửi cho người nhận và ngược lại hoặc móc dữ liệu (fishing) xuống nhằm mục đích đọc trộm thông tin (Hình 15.1 – Chuẩn các tấn công).

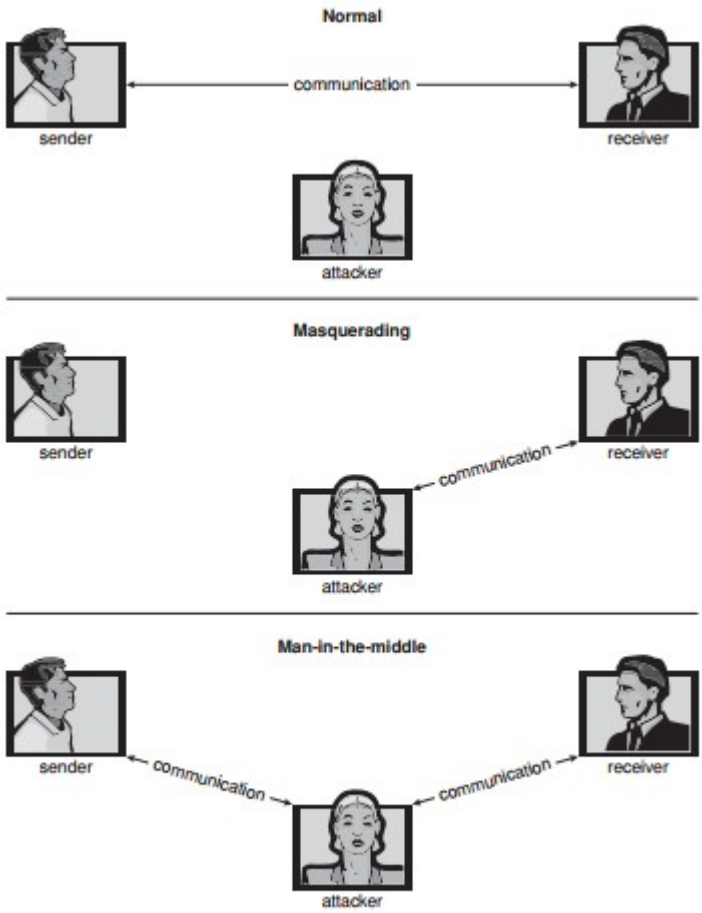


Figure 15.1 Standard security attacks.

Để bảo vệ hệ thống, chúng ta phải có các biện pháp an ninh với bốn cấp độ:

1. ***Vật lý***: Các hệ thống máy tính phải được đảm bảo tính an toàn vật lý khỏi những kẻ xâm nhập hệ thống. Cả phòng đặt hệ thống và các thiết bị đầu cuối hoặc các máy trạm truy cập vào hệ thống phải được đảm bảo.
2. ***Con người***: Việc ủy quyền phải được thực hiện một cách cẩn thận để đảm bảo rằng chỉ người có quyền mới có quyền truy cập vào hệ thống. Người dùng cũng cần cẩn thận trong việc truy cập vào một số websites lừa đảo ví dụ như email, web bất hợp pháp nhằm lấy cắp các thông tin bí mật.
3. ***Hệ điều hành:*** Hệ thống phải tự bảo vệ chính nó khỏi những hành vi vi phạm an ninh có mục đích. Một loại hình tấn công từ chối dịch vụ *(Denial-of-service)*, một số loại truy vấn nhằm tiết lộ mật khẩu, một ngắn xếp tràn có thể cho phép sự ra đời của một quá trình trái phép.
4. ***Mạng:*** Nhiều dữ liệu máy tính trong các hệ thống hiện đại đi trên đường dây bí mật, đường dây công khai như internet, kết nối không dây,.. Chặn những dữ liệu này chỉ có thể chỉ là có hại như tấn công vào hệ thống máy tính và gián đoạn thông tin liên lạc như từ chối dịch vụ,…

An ninh tại hai cấp độ đầu tiên phải được bảo trì thường xuyên là để hệ điều hành an ninh được đảm bảo.

Những kẻ xâm nhập khai thác lỗ hổng bảo mật, biện pháp đổi phó an ninh, điều này đã khiến những kẻ xâm nhập trở lên tính vi hơn. Tròn chơi mèo vơn chuột sẽ vấn tiếp tục, các công cụ bảo mật cần phải nhiều hơn để ngắn chặn những kẻ xâm nhập.

3. Các chương trình đe dọa – Program Threats

Tiến trình, cùng với *Kernel*, là các phương tiện duy nhất để hoàn thành công việc trên một máy tính. Do đó, viết một chương trình tạo ra một lỗ hổng an ninh, hoặc khiến cho một tiến trình bình thường bị thay đổi hành vi của nó và tạo ra lỗ hổng, đó là mục tiêu chung của những kẻ phá hoại. Trong thực tế, phần lớn các chương trình không có sự bảo vệ đều có thể là mục tiêu của một chương trình phá hoại. Ví dụ, một chương trình có thế giúp ta đăng nhập vào một hệ thống mà ta không được phép. Nhìn bề ngoài thì nó khá là hữu ích, tuy nhiêu ẩn sau nó là một chương trình chạy ẩn có khả năng khiến cho những kẻ phá hoại dễ dàng khai thác và truy cập thông tin ngay cả khi chương trình đã bị chặn. Trong phần này là mô tả các phương pháp phổ biến của các chương trình gây ra lỗ hổng. Lưu ý có một sự thay đổi không đáng kể các quy ước đặt tên lỗ hổng bảo mật, và chúng ta vẫn sử dụng các miêu tả hoặc thuật ngữ phổ biến nhất.

2 Trojan Horse

Rất nhiều hệ thống cho phép thực thi các chương trình được viết bởi một người dùng nhưng lại được thực thi bởi một người dùng khác. Nếu chương trình được thực hiện trong một miền cung cấp nhiều quyền truy cập, thì những người sử dụng khác có thể sử dụng sai các quyền này. Ví dụ như một chương trình soạn thảo, nó có thể chứa các mã có thể tìm kiếm các tệp tin bị sửa đổi chỉ với một từ khoá nhất định. Nếu tìm được, các tệp tin ấy có thể được sao chép vào một nơi đặc biệt rồi chuyển đến cho người viết ra chương trình ấy. Những đoạn mã như vậy được gọi là Trojan horse. Như trên các hệ thống UNIX nói chung, vấn đề Trojan hourse rất được quan tâm. Các danh sách tìm kiếm các file hoặc các chương trình phải được bảo mật, nhưng một Trojan hourse vẫn có thể xâm nhập được vào luồng của người dùng và thực thi trái phép.

Ví dụ, chúng ta sử dụng dấu "." để tìm kiếm. Dấu "." này sẽ thông báo cho shell các đường dẫn hiện tại khi ta tìm kiếm. Nếu trong đường dẫn mà người dùng tìm kiếm có dấu ".", thì có thể liên kết tới máy tính khác và chỉ cần tiếp tục nhập tên truy cập thì máy tính khác đã có thể truy cập vào máy tính của người dùng và thực thi các tác vụ trái phép hay có thể xoá các tập tin của người dùng.

Một biến thể khác của Trojan Horse là giả làm một trình đăng nhập, người dùng nhập mật khẩu và có thể phải yêu cầu nhập lại lần nữa, từ đó Trojan horse có thể lưu lại mật khẩu này và gửi về cho tin tặc sau đó nó sẽ dừng lại và bạn đã có thể đăng nhập bình thường như chưa có chuyện gì xảy ra. Các Trojan horse có thể bắt các tổ hợp phím mà người dùng hay nhập. Ví dụ như *Control–alt–delete* trong các hệ thống của Windows.

Một biến thể khác nữa đó chính là phần mềm gián điệp *(Spyware)*. Phần mềm gián điệp có thể đi kèm một phần mềm miễn phí, đôi khi có cả phần mềm trả phí. Mục tiêu của các phần mềm này là tải quảng cảo, hiển thị quảng cáo lên các trình duyệt, thu thập thông tin của người dùng và gửi tới các tin tặc. Các Spyware còn có thể được cài vào máy tính như một chương trình bình thường, nhưng nó bí mật thu thập danh bạ, email thông tin người dùng, để từ đó gửi các tin nhắn rác, spam tới các địa chỉ mà chúng thu được. Trong năm 2010, người ta ước tính rằng 90% lượng tin nhắn rác được gửi bởi phương pháp này. Việc trộm cắp dịch vụ thậm chí chưa được coi là một loại tội phạm ở hầu hết các nước.

Spyware là một ví dụ của một vấn đề vĩ mỗ: vi phạm nguyên tắc đặc quyền tối thiểu. Hầu hết các trường hợp, người dùng thường không cài đặt trực tiếp Spyware. Các chương trình như vậy thường được cài đặt bởi 2 sai lầm: đầu tiên, đó là việc người dùng cung cấp cho các chương trình quyền truy cập hệ thống nhiều hơn cần thiết (ví dụ: quyền Administrator) cho phép các chương trình có thể truy cập vào hệ thống nhiều hơn cần thiết. Đây là trường hợp do lỗi ở con người rất phổ biến. Thứ hai, đó là việc một hệ điều hành cho cung cấp quyền cho các chương trình nhiều hơn cần thiết. Đây là trường hợp do việc thiết kế hệ điều hành. Một hệ điều hành (và thực tế là một phần mềm nói chung) nên cho phép việc quản lý kiểm soát an ninh và bảo mật, nhưng nó cũng cần phải dễ quản lý và dễ hiểu. Các biện pháp an ninh bất tiện hoặc không đầy đủ ràng buộc sẽ gây ra một sự suy yếu dễ phá vỡ các hệ thống.

3 Trap Door

Những người thiết kế chương trình, hệ thống có thể cố tình để lại một lỗ hổng trong phần mềm mà chỉ họ có thể khai thác. Đây là loại vi phạm an ninh (Trap Door) được thể hiện trong bộ phim Trò chơi chiến tranh. Ví dụ, các đoạn mã có thể kiểm tra xem có một Tài khoản hoặc mật khẩu cụ thể không, và nó có thể phá vỡ các quy trình bảo mật thông thường. Các lập trình viên đã bị bắt vì tội đánh cắp tiền từ các ngân hàng bằng cách sửa các đoạn mã để có thể làm tròn số và đôi khi là chuyển tiền vào tài khoản của họ. Họ có thể lấy đi một lượng tiền lớn, hoặc theo dõi các giao dịch của một ngân hàng. Một Trap Door thông minh có thể bao gồm một trình biên dịch. Trình biên dịch này có thể tạo mã, tạo ra các đối tượng (chính là các Cánh cửa bẫy), các đối tượng này được hệ thống hiểu như một đối tượng tiêu chuẩn và không phụ thuộc vào mã nguồn của chương trình. Chúng rất cứng đầu, và gần như không để lại dấu vết gì bởi chỉ có các mã nguồn của trình biên dịch ấy sẽ chứa các thông tin gây hại.

Cửa bẫy là một vấn đề nan giải, bởi để phát hiện ra chúng thì ta phải phân tích tất cả mã nguồn của các thành phần trong hệ thống, và chúng lại có tới hàng triệu dòng mã. Các phân tích này thường không được thực hiện thường xuyên và không được thực hiện ở tất cả các thành phần trong hệ thống.

4 Logic Bomb

Đó là một chưng trình gây ra một sự cố an ninh chỉ trong một số hoàn cảnh nhất định. Nó sẽ rất khó bị phát hiện vì trong các quá trình bình thường sẽ không phát hiện ra lỗ hổng bảo mật. Tuy nhiên đến khi xảy ra một kịch bản đã định sẵn, thì các lỗ hổng sẽ được tạo ra. Các kịch bản này chính là Logic Bomb. Ví dụ, có thể lập trình để theo dõi xem một người còn làm việc với hệ thống hay không, nếu không thì một lỗ hổng có thể tạo ra để cho phép tin tặc truy cập từ xa, hoặc có thể sinh ra các mã gây hại cho hệ thống.

5 *Stack and Buffer Overflow*

Tấn công vào ngắn xếp hoặc bộ nhớ đệm là cách phổ biến nhất cho những kẻ tấn công từ bên ngoài vào hệ thống, dựa vào mạng hoặc kết nối dial-up, để có thể truy cập trái phép vào hệ thống. Về cơ bản, các cuộc tấn công khai thác lỗ hổng trong một chương trình. Lỗi có thể là do lập trình thiếu kinh nhiệm, lập trình viên bỏ quên kiểm tra dữ liệu đầu vào. Kẻ tấn công xác định lỗ hổng và viết một chương trình tấn công như sau:

- Làm tràn một trường đầu vào, đối số dòng lệnh hoặc bộ đệm cho đầu vào.
- Ghi đè địa chỉ trả lại trả lại hiện tại trên ngắn xếp với địa chỉ của mã khai thác được lạp tại bước tiếp theo.
- Viết một bộ mã đơn giản cho không gian tiếp theo trong ngăn xếp bao gồm các lệnh mà kẻ tấn công muốn thực hiện.

Kết quả của việc thực hiện chương trình tấn công sẽ là đặc quyền thực hiện lệnh trên hệ thống. Ví dụ, nếu một hình thức webpage hy vọng một tên người dùng được nhập vào một trường, kẻ tấn công có thể gắn them các ký tự tràn bộ nhớ đệm và tiếp cận ngắn xếp, cộng với một địa chỉ trả về mới để tải lên các ngăn xếp. Khi đọc bộ nhớ đệm chương trình trả về cho nới thực hiện, trả về địa chỉ là mã khai thác được và chạy mã này.

Các kĩ thuật kinh điển cho việc khai thác vào tràn bộ đệm là để ghi đè lên các địa chỉ trả lại với một con trỏ để kẻ tấn công kiểm soát dữ liệu. Hiện tượng tràn bộ nhớ đệm còn thường bị khai thác bởi tin tặc làm cho một chương trình đang chạy thực thi một đoạn mã tỳ ý được cung cấp.

```
#include <stdio.h>
#define BUFFER_SIZE 256

int main(int argc, char *argv[])
{
   char buffer[BUFFER_SIZE];

   if (argc < 2)
      return -1;
   else {
      strcpy(buffer,argv[1]);
      return 0;
   }
}
```

Figure 15.2 C program with buffer-overflow condition.

6 *Viruses*

Virus là một chương trình đe dọa tới máy tính. Chúng là những đoạn mã được nhúng trong các phần mềm hợp pháp. Virus tự động sao chép và được thiết kế để "lây nhiễm" tới các chương trình khác. Virus có thể tàn phá hệ thống bằng cách sửa đổi hoặc phá hủy các tập tin khiến hệ thống gặp sự cố, trục trặc. Hầu hết các cuộc tấn công của virus gây ra nhắm tới các kiến trúc máy tính, hệ điều hành và

các ứng dụng. Virus là một vấn đề đặc biệt của người dùng máy tính cá nhân. UNIX và một số hệ điều hành đa người dùng khác thông thường không dễ bị nhiễm virus bởi các chương trình thực thi được hệ điều hành bảo vệ khỏi sự ghi dữ liệu lên. Ngay cả khi virus lây nhiễm thì khả năng của nó thường bị giới hạn vì các khía cạnh khác của hệ thống đã được hệ điều hành bảo vệ.

Virus có thể lây truyền qua email, thư rác, chúng cũng có thể lây lan khi người dùng tải về các tập tin được chia sẻ trên Internet hoặc từ các đĩa nhiễm virus.

Một hình thức phổ biến để truyền virus chính là sử dụng các tập tin Microsoft Office, chẳng hạn như các văn bản Microsoft Word. Những văn bản này có thể chứa các macro (hoặc chương trình Visual Basic) mà các chương trình trong bộ phần mềm văn phòng *(Word, PowerPoint và Excel)* sẽ thực thi tự động. Vì các chương trình này chạy dưới tài khoản riêng của người sử dụng nên các macro có thể chạy phần lớn không bị giới hạn (ví dụ xóa các tập tin người dùng theo ý muốn). Thông thường, virus cũng sẽ tự gửi e-mail tới những người dùng khác trong danh bạ. Dưới đây là đoạn code mô tả cách đơn giản để viết một marco Visual Basic, đây là một virus khiến ổ địa bị format khi tập tin chứa marco được mở ra:

```
Sub AutoOpen()
Dim oFS
    Set oFS = CreateObject(''Scripting.FileSystemObject'')
    vs = Shell(''c: command.com /k format c:'',vbHide)
End Sub
```

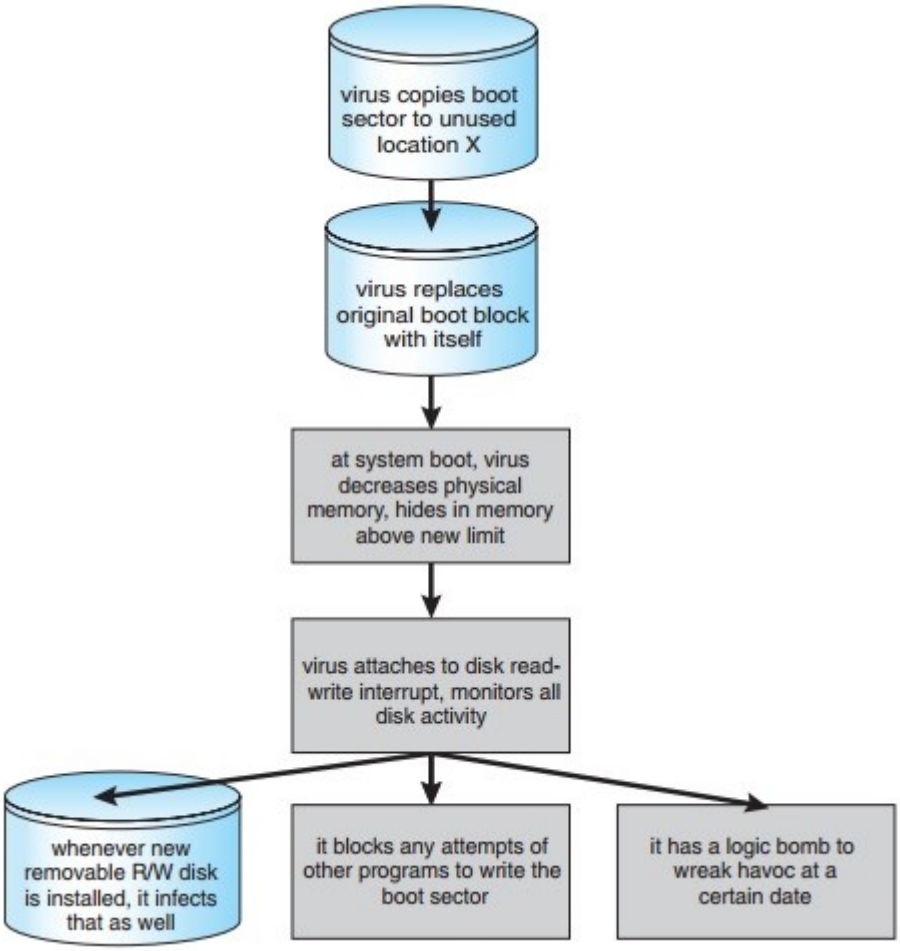
Virus được hoạt động như thế nào?

Mỗi khi virus tới máy tính mục tiêu, một chương trình được biết đến như một "ống nhỏ virus" sẽ chèn virus vào hệ thống. Ống nhỏ virus này thường là các Trojan, tuy có nhiều mục đích hoạt động nhưng cài đặt virus là một hoạt động chính của chúng. Mỗi khi được cài đặt, virus có thể gây ra một trong rất nhiều các mối nguy hại. Hiện nay có hàng ngàn loại virus nhưng chúng được chia ra thành một vài loại chính. Lưu ý rằng một virus có thể thuộc nhiều loại:

- ***Virus File*** : Virus tập tin lây nhiễm vào hệ thống bằng cách gắn chính nó vào một file. Virus này thay đổi việc bắt đầu chương trình khiến việc thực thi nhảy tới đoạn mã của chính nó. Sau khi thực hiện, virus này sẽ điều khiển cho chương trình không nhận ra chúng. Những virus loại này được gọi là virus kí

sinh bởi chúng không để lại các tập tin và chương trình chủ nơi chúng kí sinh vẫn có đầy đủ các chức năng.

- ***Virus Boot***: Virus khởi động lây nhiễm vào vùng khởi động của hệ thống, chúng hoạt động mỗi khi hệ thống được khởi động và trước khi hệ điều hành được nạp .



Vùng khởi động của máy tính nhiễm virus boot

- ***Virus Marco:*** Hầu hết các virus loại này được viết bởi ngôn ngữ bậc thấp như hợp ngữ hoặc ngôn ngữ C. Marco cũng có thể được viết bởi các ngôn ngữ bậc cao như Visual Basic. Những virus này được kích hoạt khi một chương trình có khả năng thực hiện các marco chạy. Ví dụ, một virus marco có thể chứa trong một tập tin bảng tính (ví dụ tập tin Excel).
- ***Virus Source:*** Một virus mã nguồn sẽ tìm kiếm một mã nguồn và sửa nó để chứa virus và phát tán virus
- ***Virus đa hình:*** Virus đa hình là virus có khả năng tự động biến đổi mã lệnh để tạo ra các dạng mã độc khác nhau (đa hình - nhiều hình) trong mỗi lần lây nhiễm. Chính vì vậy, virus đa hình có khả năng lẩn trốn tinh vi trước sự truy

quét của các phần mềm diệt virus. Virus đa hình thường thực hiện biến đổi mã lệnh một cách ngẫu nhiên hoặc theo một thuật toán dựa trên thời gian hay đối tượng lây nhiễm. Có nhiều loại virus đa hình khác nhau, nhưng phổ biến nhất là đa hình mã lây nhiễm và đa hình mã phá hoại.

- ***Virus mã hóa:*** Một virus mã hóa bao gồm mã giải cùng với các virus được mã hóa để tránh khỏi sự phát hiện. Ban đầu virus sẽ được giải mã và sau đó chúng được thực thi.
- ***Virus lén:*** Virus này cố gắng để tránh bị phát hiện bằng cách thay đổi các bộ phận của hệ thống có thể được sử dụng để phát hiện ra nó. Ví dụ, nó có thể sửa đổi các lời gọi hệ thống đọc để nếu các tập tin đã bị sửa đổi được hệ thống đọc thì đoạn mã ban đầu sẽ được gọi chứ không phải đoạn mã bị nhiễm.
- ***Tunneling***: Virus này sẽ cố gắng để vượt qua sự phát hiện của trình quét virus bằng cách cài đặt chính nó trong chuỗi ngắt-handler. Chúng cũng tự cài đặt mình trong các trình điều khiển thiết bị.
- ***Multipartite***: Virus loại này có thể lây nhiễm sang nhiều bộ phận của một hệ thống bao gồm cả phần khởi động, bộ nhớ và các tập tin, điều này sẽ giúp cho chúng khó bị phát hiện.

Virus vẫn đang tiếp tục phát triển từng ngày. Ví dụ trong năm 2004 một loại virus phát tán mới được tìm thấy. Chúng khai thác 3 lỗi riêng biệt để tiến hành hoạt động. Virus này bắt đầu bằng việc lây nhiễm hàng trăm máy chủ Windows (bao gồm nhiều trang web tin cậy) chạy Microsoft Internet Information Server (IIS). Bất kì một trình duyệt web Microsoft Explore truy cập vào đây đều được nhận trình duyệt tải về các virus.

Nhìn chung, phần lớn virus gây ra những cuộc tấn công an ninh bởi chúng hiệu quả, virus sẽ tiếp tục được viết ra và lan rộng.

2 Mã hóa – Cryptography

Có rất nhiều vấn đề về an ninh và các phần mềm đe dọa đang ngày càng tinh vi, vì vậy vấn đề về phòng thủ và bảo vệ tài nguyên thông tin trên các hệ thống là rất quan trọng và cấp bách. Trong báo cáo này chúng tôi sẽ giải thích những nguyên tắc cơ bản trong mã hóa.

2 Mã hóa – Encryption

Mã hóa được sử dụng thường xuyên trong rất nhiều khía cạch của các máy tính hiện đại. Nó được sử dụng để gửi văn bản bảo mật qua mạng, cũng như bảo vệ cơ sở dữ liệu, file, và đĩa cứng khỏi những truy cập trái phép của những kẻ tấn công. Một thuật toán mã hóa cho phép người gửi một *message* để chắc chắn rằng chỉ người nhận thực sự có key mới có thể đọc được *message*.

Trong báo cáo này, chúng tôi mô tả những thuật toán mã hóa thông dụng nhất. Và một thuật toán mã hóa chứa những thành phần sau:

- *K được gọi là là key;*
- *M được gọi làmessage hay plaintext hay cleartext;*
- *C được gọi là ciphertext;*
- *Mã hóa (Encrypting function) E: K -> (M -> C);*
- *Giải mã (Decrypting function) D: K -> (C -> M).*

Một thuật toán mã hoát phải cung cấp mức độ bảo mật cần thiết. Cho một M, máy tính có thể tính $C= E_k(M)$ với khóa *K*. Vì vậy chỉ có máy tính có được khóa K có thể giải mã C thành M. Bởi vậy thuật toán mã hóa phải đảm bảo tính an toàn của dữ liệu không thể bị phá vỡ khi không có khóa.

Có hai kiểu mã hóa chính đó là: Mã hóa đối xứng (Symmetric encryption) và mã hóa bất đối xứng (Asymmetric encryption). Vì vậy trong báo cáo này chúng tôi chỉ đề cập đến hai loại mã hóa này.

1 Mã hóa đối xứng – Symmetric Encryption.

Mã hóa đối xứng (Hay còn gọi là mã hóa khóa bí mật) là phương pháp mã hóa mà key mã hóa và key giải mã là như nhau (Sử dụng cùng một secret key để mã hóa và giải mã). Đây là phương pháp thông dụng nhất hiện nay dùng để mã hóa dữ liệu truyền nhận giữa hai bên. Vì chỉ cần có secret key là có thể giải mã được, nên bên gửi và bên nhận cần làm một cách nào đó để cùng thống nhất về secret key. Hình 15.7 sẽ cho chúng ta thấy ví dụ của hai người muốn trao đổi an toàn trên kênh không an toàn như mạng thông qua mã hóa đối xứng.

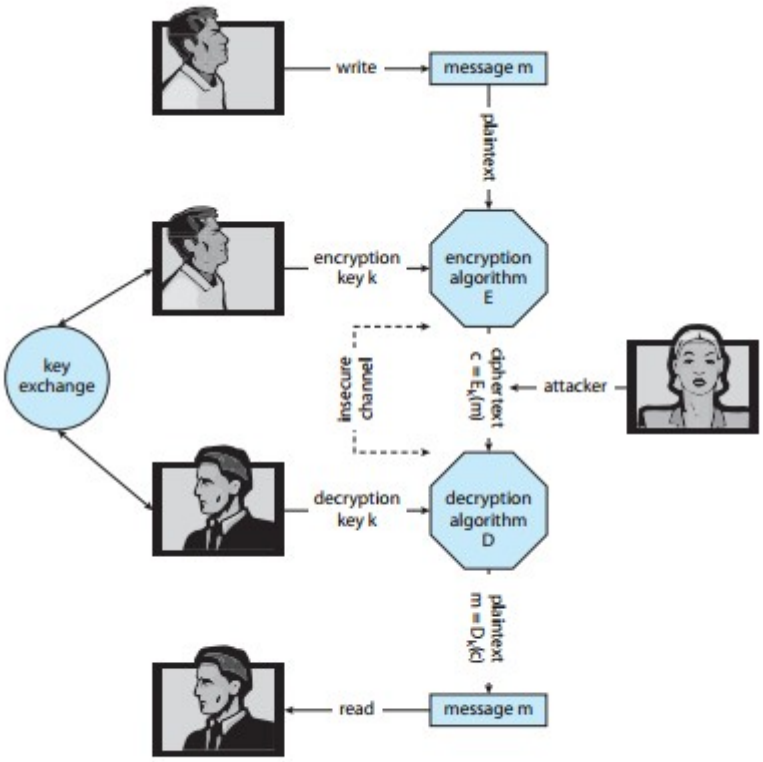


Figure 15.7 A secure communication over an insecure medium.

Vấn đề lớn nhất của phương pháp mã hóa đối xứng là làm sao để "thỏa thuận" secret key giữa bên gửi và bên nhận, vì nếu truyền secret key từ bên gửi sang bên nhận mà không dùng một phương pháp bảo vệ nào thì bên thứ ba cũng có thể dễ dàng lấy được secret key này.

Khoảng vài thập kỷ trước, loại mã hóa đối xứng được sử dụng phổ biến nhất ở nước Mỹ là data-encryption standard (DES) được thông qua bởi viện tiêu chuẩn và công nghệ quốc gia (NIST). DES làm việc với từng khối *64-bit* bởi DES làm việc trên từng khối bit tại một thời điểm, được biết đến bởi thuật ngữ là mã hóa khối *"block cipher".* Tuy nhiên ngày nay DES không còn an toàn cho việc bảo vệ tài nguyên bởi vì khóa của nó được tìm ra một chỉ với tốc độ của máy tính thông thường. NIST đã cải tiến và đưa ra một bản thay thế gọi là 3DES (triple DES), trong đó thuật toán DES được lặp lại 3 lần (2 lần mã hóa và mọt lần giải mã) trên cùng một M sử dụng 2 hoặc 3 khóa. Cho ví dụ, $C= E_{k3}(D_{k2}(E_{k1}(m)))$ – khi 3 khóa được sử dụng, độ dài của khóa là 168-bit. 3DES vẫn còn được sử dụng phổ biến cho đến ngày nay.

Vào năm 2001, NIST đã thông qua một thuật toán mã hóa khối mới là AES – Advanced encryption standard để thay thế cho DES. AES cũng là một kiểu mã hóa khối nó có thể sử dụng khóa có độ dài là *128,192,256-bits* và thường sử dụng là 128-bits/ blocks. Để mã hóa những văn bản có độ dài vượt quá độ dài của khối, người ta sử dụng thuật toán theo một chế độ mã hóa khối nào đó. Phân biệt với mã hóa khối là mã hóa dòng. Mã hóa dòng làm việc trên từng bít của dòng dữ liệu và quá trình biến đổi thay đổi theo quá trình mã hóa. Tuy nhiên, sự phân biệt giữa 2 phương pháp nhiều khi không rõ ràng vì mã hóa khối khi hoạt động theo một chế độ nào đó thì có tác dụng như một phương pháp mã hóa dòng.

RC4 có lẽ là loại mã hóa dòng phổ biến nhất. Một mã hóa dòng được thiết kế để mã hóa và giải mã một dòng bytes hoặc bits chứ không phải là một khối. Điều này là hữu ích khi chiều dài của dữ liệu sẽ khiến cho mã hóa khối quá chậm. RC4 được sử dụng trong mã hóa dòng dữ liệu, như trong WEB, giao thức LAN không dây. Tuy nhiên ngày nay RC4 được sử dụng trong WEB *(chuẩn 802.11)* đã bị tìm thấy để phá vỡ trong thời gian ngắn. Và trong thực tế chính bản thân RC4 cũng có lỗ hổng.

2 Mã hóa bất đối xứng – Asymmetric Encryption.

Thuật toán mã hóa bất đối xứng hay còn được gọi là Public-key. Năm 1976 một kiểu thuật toán khác được giới thiệu bởi Whitfield Diffie, Martin Hellman và Ralph Merkle. Trong mã hóa khóa công khai, một người dùng sở hữu một khóa an ninh như mã hóa đối xứng nhưng một khóa công khai. Mã hóa đối xứng có thể được sử dụng cho các ứng dụng như chữ ký số, …

Ví dụ cho việc làm thể nào để mã hóa bằng khóa công khai, chúng ta bảo luận một thuật toán như RSA. RSA là được sử dụng rất rộng trong thuật toán mã hóa bất đối xứng.

Trong RSA, K_e là khóa công khai và K_d là khóa bí mật. N là tích của hai số nguyên tố lớn p và q. Nó phải tính toán được để lấy được $K_{d.N}$ từ $K_{e.N}$, vậy K_e cần phải được giữ bảo mật và có được phổ biến rộng . Thuật toán mã hóa là $E_{ke.N}(m) = m^{ke} \bmod N$, ở đây K_e điều kiện $K_e K_d \bmod (p-1)(q-1)=1$;

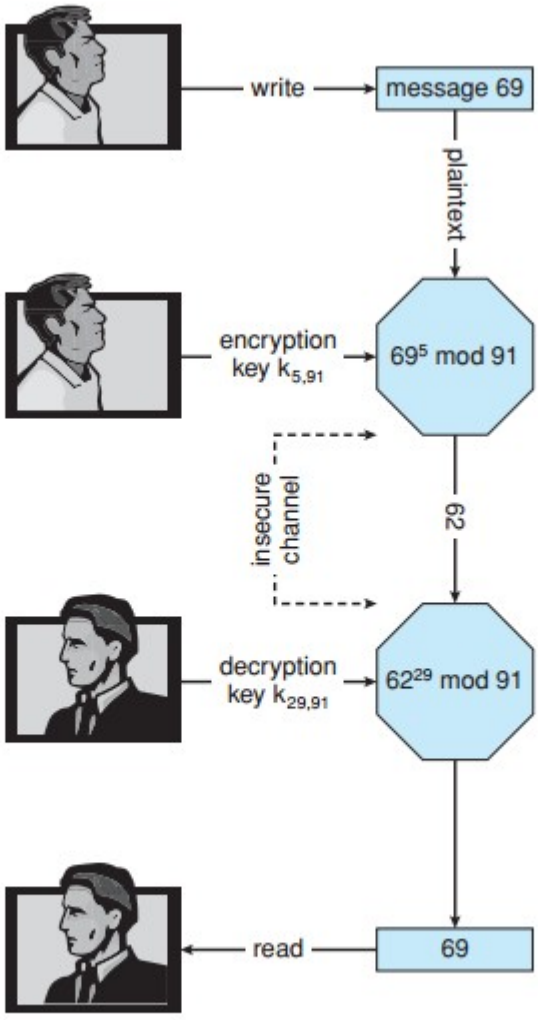
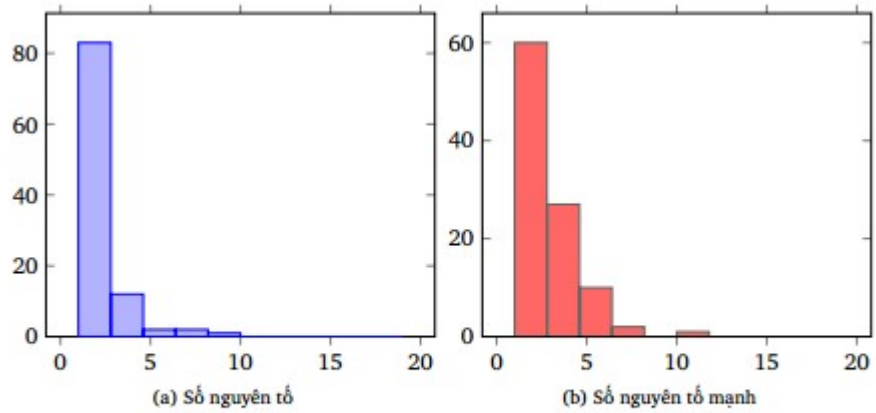


Figure 15.8 Encryption and decryption using RSA asymmetric cryptography.

Ví dụ sử dụng giá trị nhỏ được thể hiện ở hình 15.8. Trong ví dụ này, chúng tôi tạo hai số $p = 7$ và $q = 13$ là hai số nguyên tố cơ nhỏ. Chúng tôi tính được $N=p*q=7*13=91$ và $(p-1)*(q-1)=72$. Tiếp theo tiến hành chọn $K_e < 72$ và $\bmod N = 1$, ở đây chúng tôi chọn là 5. Cuối cùng chúng ta tính K_d sao cho $K_e K_d \bmod 72 =1$, ở đây chúng tôi chọn là 29. Tại đây chúng ta có khóa công khai là $K_{e,N}=5, 91$ và khóa bí mật là $K_{d,N}=29, 91$. Ta tiến hành mã hóa một văn bản có giá trị là 69 bằng khóa công khai *"public key"* kết quả được là 62, được gửi cho người tạo khóa giả mã bằng khóa bí mật "private key".

Ngày nay việc thám mã ngày một phát triển, đặc biệt trong RSA điểm quan trọng của thuật toán chính là hai số nguyên tố *p,q*. Và những kẻ tấn công thường sẽ đi thẳng vào việc phân tích thừa số nguyên tố N thành p và q điển hình là phương pháp *p - 1* của *Prollard*. Vì vậy hai số p và q phải là hai số cực lớn và mạnh, việc tìm được hai số nguyên tố lớn và mạnh các bạn có thể tham khảo hai thuật toán *Rabin-Miller-Test* và *Gordon*. Có thể tham khảo tại đây: https://github.com/Digitalsignature/*PrGlib*.

Chúng tôi đã cài đặt thử và chạy thử nhiệm thuật toán để sinh 100 sô nguyên tố có độ dài 3072-bit. Chương trình chạy trên máy laptop có bộ xử lý Intel, Core i7 Haswell (4500U, 1.8 GHz), Ram 4GB, 1600-MHz và chạy trên hệ điều hành GNU/Linux (Ubuntu 15.04). Thời gian trung bình để sinh ra mỗi số khoảng 2.09s và 2.75s với mỗi số nguyên tố mạnh.



Hình 1: Mô tả các histogram thời gian chạy cho việc sinh ngẫu nhiên 100 số nguyên tố cỡ 3072 bit. Cột của bảng mô tả số lượng số nguyên tố chạy trong thời gian (tính theo giây) chỉ ra bởi hàng.

4. Xác thực – Authentication

Xác thực là một cách chứng minh rằng một văn bản có toàn ven hay đơn giản hơn là có bị thay đổi hay không trong quá trình gửi trên mạng. Và một thuật toán xác thực sử dụng khóa đối xứng như sau:

- *K* được gọi là *key;*
- *M* được gọi là *messages;*
- *A* được gọi là *authenticators;*
- Hàm xác thực *S: K -> (M -> A);*
- Hàm xác thực *V: K -> (MxA -> {true, false}).*

5. Tài liệu tham khảo

[1] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, Charles E. Leiserson, *Introduction to Algorithms*, McGraw-Hill Higher Education 2001.

[2] Victor Shoup, *A computational introduction to number theory and algebra*, Cambridge University Press 2006,

[3] Jonathan Katz và Yehuda Lindell, *Introduction to Modern Cryptography*, Chapman & Hall/CRC, 2007.

[4] Ronald L. Rivest, Robert D. Silverman y, *Are 'Strong' Primes Needed for RSA?,* November 22, 1999.