1. Tấn công vào mật khẩu

1.1 Giới thiệu

Tấn công vào mật khẩu (Password attack) là dạng tấn công nhằm đánh cắp mật khẩu và thông tin tài khoản của người dùng để lạm dụng. Tên người dùng và mật khẩu không

được mã hóa có thể bị đánh cắp trên đường truyền từ máy khách đến máy chủ, hoặc các thông tin này có thể bị đánh cắp thông qua các dạng tấn công XSS, hoặc lừa đảo, bẫy người dùng cung cấp thông tin. Đây là một trong các dạng tấn công phổ biến nhất do hầu hết các ứng dụng sử cơ chế xác thực người dùng dựa trên tên người dùng, hoặc email và mật khẩu. Nếu kẻ tấn công có tên người dùng và mật khẩu thì hắn có thể đăng nhập vào tài khoản và thực hiện các thao tác như người dùng bình thường.

1.2 Mô tả

Có thể chia tấn công vào mật khẩu thành 2 dạng:

- Tấn công dựa trên từ điển (Dictionary attacks): Dạng tấn công này khai thác vấn đề người dùng có xu hướng chọn mật khẩu là các từ đơn giản cho dễ nhớ. Kẻ tấn công thử các từ có tần suất sử dụng cao làm mật khẩu trong từ điển, nhờ vậy tăng khả năng thành công.
- Tấn công vét cạn (Brute force attacks): Dạng vét cạn sử dụng tổ hợp các ký tự và thử tự động. Phương pháp này thường được sử dụng với các mật khẩu đã được mã hóa. Kẻ tấn công sinh tổ hợp ký tự, sau đó mã hóa với cùng thuật toán mà hệ thống sử dụng, tiếp theo so sánh chuỗi mã hóa từ tổ hợp ký tự với chuỗi mật khẩu mã hóa thu thập được. Nếu hai bản mã trùng nhau thì tổ hợp ký tự là mật khẩu.

1.3 Phòng chống

Để đảm bảo an toàn cho mật khẩu, cần thực hiện kết hợp các biện pháp sau:

- Chọn mật khẩu đủ mạnh: Mật khẩu mạnh cho người dùng thông thường cần có độ dài lớn hơn hoặc bằng 8 ký tự, gồm tổ hợp của 4 loại ký tự: chữ cái hoa, chữ cái thường, chữ số và ký tự đặc biệt (?#$...). Mật khẩu cho người quản trị hệ thống cần có độ dài lớn hơn hoặc bằng 10 ký tự cũng với các loại ký tự như mật khẩu cho người dùng thông thường.
- Định kỳ thay đổi mật khẩu. Thời hạn đổi mật khẩu tùy thuộc vào chính sách an ninh của cơ quan, tổ chức, có thể là 3 tháng, hoặc 6 tháng.
- Mật khẩu không nên lưu ở dạng rõ (plaintext). Nên lưu mật khẩu ở dạng đã mã hóa (thường dùng hàm băm).

- Hạn chế trao đổi tên người dùng và mật khẩu trên kênh truyền không được mã hóa.

2. Tấn công bằng mã độc

2.1 Giới thiệu

Tấn công bằng mã độc (Malicious code attacks) là dạng tấn công sử dụng các mã độc (Malicious code) làm công cụ để tấn công hệ thống nạn nhân. Tấn công bằng mã độc có thể chia thành 2 loại:

- Khai thác các lỗ hổng về lập trình, lỗ hổng cấu hình hệ thống để chèn và thực hiện mã độc trên hệ thống nạn nhân. Loại tấn công này lại gồm 2 dạng:

+ Tấn công khai thác lỗi tràn bộ đệm (Buffer Overflow)

+ Tấn công khai thác lỗi không kiểm tra đầu vào, gồm tấn công chèn mã SQL (SQL Injection) và tấn công sử dụng mã script, kiểu XSS, CSRF.

- Lừa người sử dụng tải, cài đặt và thực hiện các phần mềm độc hại, như:

+ Các phần mềm quảng cáo (Adware), gián điệp (Spyware)

+ Vi rút

+ Zombie/Bot

+ Trojan

Dạng tấn công lừa người sử dụng tải, cài đặt và thực hiện các phần mềm độc hại sẽ được đề cập ở Mục 2.4. Mục này chủ yếu đề cập về tấn công khai thác lỗi tràn bộ đệm, tấn công khai thác lỗi không kiểm tra đầu vào, trong đó tập trung vào tấn công chèn mã SQL.

2.2 Tấn công khai thác lỗi tràn bộ đệm

2.2.1 Giới thiệu và nguyên nhân

Lỗi tràn bộ đệm (Buffer overflow) là một trong các lỗi thường gặp trong các hệ điều hành và đặc biệt nhiều ở các phần mềm ứng dụng [6]. Lỗi tràn bộ đệm xảy ra khi một ứng dụng cố gắng ghi dữ liệu vượt khỏi phạm vi của bộ nhớ đệm, là giới hạn cuối hoặc cả giới hạn đầu của bộ đệm. Lỗi tràn bộ đệm có thể khiến ứng dụng ngừng hoạt động, gây mất dữ liệu hoặc thậm chí giúp kẻ tấn công chèn, thực hiện mã độc để kiểm soát hệ thống. Lỗi tràn bộ đệm chiếm một tỷ lệ lớn trong số các lỗi gây lỗ hổng bảo mật [6]. Tuy nhiên, trên thực tế không phải tất cả các lỗi tràn bộ đệm đều có thể bị khai thác bởi kẻ tấn công.

Lỗi tràn bộ đệm xuất hiện trong khâu lập trình phần mềm (coding) trong quy trình phát triển phần mềm. Nguyên nhân của lỗi tràn bộ đệm là người lập trình không kiểm tra, hoặc kiểm tra không đầy đủ các dữ liệu đầu vào nạp vào bộ nhớ đệm.

Khi dữ liệu có kích thước quá lớn hoặc có định dạng sai được ghi vào bộ nhớ đệm, nó sẽ gây tràn và có thể ghi đè lên các tham số thực hiện chương trình, có thể khiến chương trình bị lỗi và ngừng hoạt động. Một nguyên nhân bổ sung khác là việc sử dụng các ngôn ngữ với các thư viện không an toàn, như hợp ngữ, C và C++.

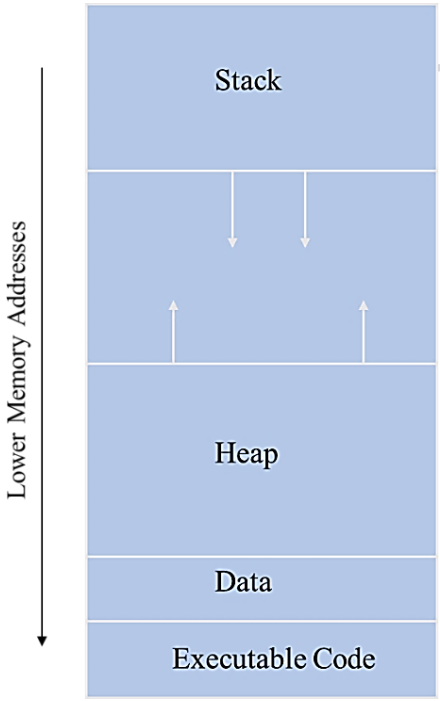
2.2.2 Cơ chế gây tràn và khai thác

* Cơ chế gây tràn

Trên hầu hết các nền tảng, khi một ứng dụng được nạp vào bộ nhớ, hệ điều hành cấp phát các vùng nhớ để tải mã và lưu dữ liệu của chương trình. Hình 2.8 minh họa các vùng bộ nhớ cấp cho chương trình, bao gồm vùng lưu mã thực hiện (Executable code), vùng lưu dữ liệu toàn cục (Data), vùng bộ nhớ cấp phát động (Heap) và vùng bộ nhớ ngăn xếp (Stack). Vùng bộ nhớ ngăn xếp là vùng nhớ lưu các tham số gọi hàm, thủ tục, phương thức (gọi chung là hàm hay chương trình con) và dữ liệu cục bộ của chúng. Vùng nhớ cấp phát động là vùng nhớ chung lưu dữ liệu cho ứng dụng, được cấp phát hay giải phóng trong quá trình hoạt động của ứng dụng.

Chúng ta sử dụng vùng bộ nhớ ngăn xếp để giải thích cơ chế gây tràn và khai thác lỗi tràn bộ đệm. Bộ nhớ ngăn xếp được cấp phát cho chương trình dùng để lưu các biến cục bộ của hàm, trong đó có các biến nhớ được gọi là bộ đệm, các tham số hình thức của hàm, các tham số quản lý ngăn xếp, và địa chỉ trở về (Return address). Địa chỉ trở về là

địa chỉ của lệnh nằm kế tiếp lời gọi hàm ở chương trình gọi được tự động lưu vào ngăn xếp khi hàm được gọi. Khi việc thực hiện hàm kết thúc, hệ thống nạp địa chỉ trở về đã lưu trong ngăn xếp vào con trỏ lệnh (còn gọi là bộ đếm chương trình) kích hoạt việc quay trở lại thực hiện lệnh kế tiếp lời gọi hàm ở chương trình gọi.



Hình 2.8. Các vùng bộ nhớ cấp cho chương trình

```
// định nghĩa một hàm
void function(int a, int b, int c){
char buffer1[8];
char buffer2[12];
}
// chương trình chính
int main(){
function(1,2,3); // gọi hàm
}
```

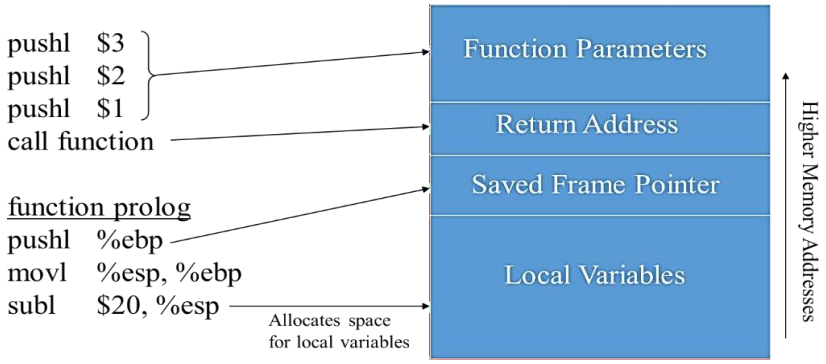
Hình 2.9. Một chương trình minh họa cấp phát bộ nhớ trong ngăn xếp

Hình 2.9 là một đoạn chương trình gồm một hàm con (*function()*) và một hàm chính (*main()*) minh họa cho việc gọi làm và cấp phát bộ nhớ trong vùng nhớ ngăn xếp. Hàm *function()* có 3 tham số hình thức kiểu nguyên và kê khai 2 biến cục bộ *buffer1* và *buffer2* kiểu xâu ký tự. Hàm chính *main()* chỉ chứa lời gọi đến hàm *function()* với 3 tham số thực.

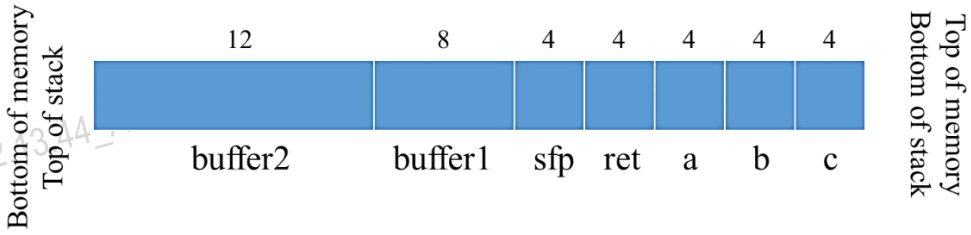
Hình 2.10 biểu diễn việc cấp pháp bộ nhớ cho các thành phần trong ngăn xếp: các tham số gọi hàm được lưu vào Function Parameters, địa chỉ trở về được lưu vào ô Return Address, giá trị con trỏ khung ngăn xếp được lưu vào ô Save Frame Pointer và các biến cục bộ trong hàm được lưu vào Local Variables. Hình 2.11

minh họa chi tiết việc cấp

phát bộ nhớ cho các biến trong ngăn xếp: ngoài ô địa chỉ trở về (ret) và con trỏ khung (sfp) được cấp cố định ở giữa, các tham số gọi hàm được cấp các ô nhớ bên phải (phía đáy ngăn xếp – bottom of stack) và các biến cục bộ được cấp các ô nhớ bên trái (phía đỉnh ngăn xếp – top of stack).



Hình 2.10. Các thành phần được lưu trong vùng bộ nhớ trong ngăn xếp



Hình 2.11. Cấp phát bộ nhớ cho các biến nhớ trong vùng bộ nhớ trong ngăn xếp

```
// định nghĩa một
hàm void
function(char *str){
char buffer[16];
strcpy(buffer,
str);
}
// chương trình
chính int main(){
char large_string[256];
int i;
for (i = 0; i < 255; i++){
large_string[i] = 'A';
```

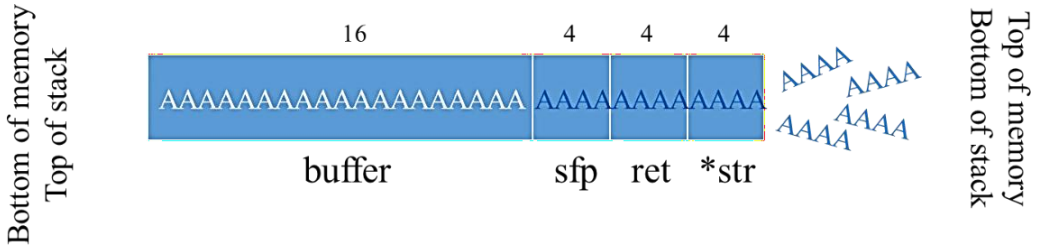
Hình 2.12. Một chương trình minh họa gây tràn bộ nhớ đệm trong ngăn xếp

Hình 2.12 là một đoạn chương trình minh họa gây tràn bộ nhớ đệm trong ngăn xếp. Đoạn chương trình này gồm hàm con *function()* và hàm chính *main()*, trong đó hàm

function() nhận một con trỏ xâu ký tự *str* làm đầu vào. Hàm này khai báo 1 biến *buffer* kiểu xâu ký tự với độ dài 16 byte. Hàm này sử dụng hàm thư viện *strcpy()* để sao chép xâu ký tự từ con trỏ *str* sang biến cục bộ *buffer*. Hàm chính *main()* kê khai một xâu ký tự *large_string* với độ dài 256 byte và sử dụng một vòng lặp để điền đầy xâu *large_string* bằng ký tự 'A'. Sau đó *main()* gọi hàm *function()* với tham số đầu vào là *large_string*.

Có thể thấy đoạn chương trình biểu diễn trên Hình 2.12 khi được thực hiện sẽ gây tràn trong biến nhớ *buffer* của hàm function() do tham số truyền vào *large_string* có kích thước 256 byte lớn hơn nhiều so với *buffer* có kích thước 16 byte và hàm *strcpy()* không hề thực hiện việc kiểm tra kích thước dữ liệu vào khi sao chép vào biến *buffer*. Như minh họa trên Hình 2.13, chỉ 16 byte đầu tiên của *large_string* được lưu vào *buffer*, phần còn lại được ghi đè lên các ô nhớ khác trong ngăn xếp, bao gồm *sfp*, *ret* và cả con trỏ xâu đầu vào *str*. Ô nhớ chưa địa chỉ trở về *ret* bị ghi đè và giá trị địa chỉ trở về mới là 'AAAA' (0x41414141).

Khi kết thúc thực hiện hàm con function(), chương trình tiếp tục thực hiện lệnh tại địa chỉ 0x41414141. Đây không phải là địa chỉ của lệnh chương trình phải thực hiện theo lôgic đã định ra từ trước.



Hình 2.13. Minh họa hiện tượng tràn bộ nhớ đệm trong ngăn xếp

Như vậy, lỗi tràn bộ đệm xảy ra khi dữ liệu nạp vào biến nhớ (gọi chung là bộ đệm) có kích thước lớn hơn so với khả năng lưu trữ của bộ đệm và chương trình thiếu các bước kiểm tra kích thước và định dạng dữ liệu nạp vào. Phần dữ liệu tràn sẽ được ghi đè lên các ô nhớ liền kề trong ngăn xếp, như các biến cục bộ khác, con trỏ khung, địa chỉ trở về, các biến tham số đầu vào,....

* Khai thác lỗi tràn bộ đệm

Khi một ứng dụng chứa lỗ hổng tràn bộ đệm, tin tặc có thể khai thác bằng cách gửi mã độc dưới dạng dữ liệu đến ứng dụng nhằm ghi đè, thay thế địa chỉ trở về với mục đích tái định hướng chương trình đến thực hiện đoạn mã độc mà tin tặc

gửi đến. Đoạn mã độc tin tặc xây dựng là mã máy có thể thực hiện được và thường được gọi là *shellcode*. Như vậy, để có thể khai thác lỗi tràn bộ đệm, tin tặc thường phải thực hiện việc gỡ rối (debug) chương trình (hoặc có thông tin từ nguồn khác) và nắm chắc cơ chế gây lỗi và phương pháp quản lý, cấp phát vùng nhớ ngăn xếp của ứng dụng.

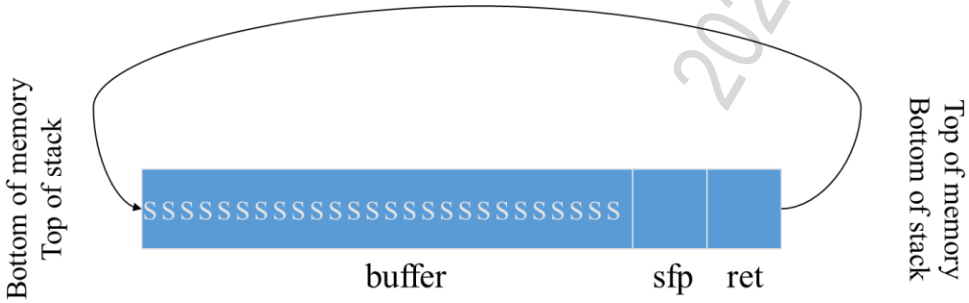
Mã *shellcode* có thể được viết bằng hợp ngữ, C, hoặc các ngôn ngữ lập trình khác, sau đó được chuyển thành mã máy, rồi chuyển định dạng thành một chuỗi dữ liệu và cuối cùng được gửi đến ứng dụng. Hình 2.14 minh họa một đoạn mã *shellcode* viết bằng hợp

ngữ và được chuyển đổi thành một chuỗi dưới dạng hexa làm dữ liệu đầu vào gây tràn bộ đệm và gọi thực hiện *shell sh* trong các hệ thống Linux hoặc Unix thông qua lệnh /bin/sh. Hình 2.15. minh họa việc chèn *shellcode*, ghi đè lên ô nhớ chứa địa chỉ trở về *ret*, tái định hướng việc trở về từ chương trình con, chuyển đến thực hiện mã *shellcode* được chèn vào. Trên thực tế, để tăng khả năng đoạn mã *shellcode* được thực hiện, người ta thường chèn một số lệnh NOP (N) vào phần đầu *shellcode* để phòng khả năng địa chỉ ret mới không trỏ chính xác đến địa chỉ bắt đầu *shellcode*, như minh họa trên Hình 2.16. Lệnh NOP (No OPeration) là lệnh không thực hiện tác vụ nào cả, chỉ tiêu tốn một số chu kỳ của bộ vi xử lý.

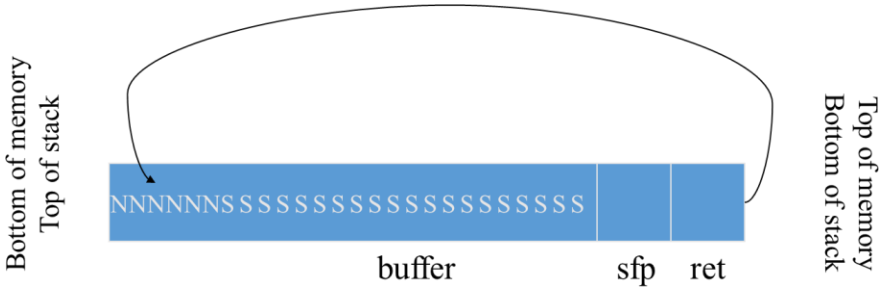
```
jmp     0x1F
popl    %esi
movl    %esi, 0x8(%esi)
xorl    %eax, %eax
movb    %eax, 0x7(%esi)
movl    %eax, 0xC(%esi)
movb    $0xB, %al
movl    %esi, %ebx
leal    0x8(%esi), %ecx
leal    0xC(%esi), %edx
int     $0x80
xorl    %ebx, %ebx
movl    %ebx, %eax
inc     %eax
int     $0x80
call    -0x24
.string "/bin/sh"
```

```
char shellcode[] =
"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89"
"\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c"
"\xcd\x80\x31\xdb\x89\xd8\x40\xcd\x80\xe8\xdc\xff"
"\xff\xff/bin/sh";
```

Hình 2.14. Một shellcode viết bằng hợp ngữ và chuyển thành chuỗi tấn công



Hình 2.15. Chèn và thực hiện shellcode khai thác lỗi tràn bộ đệm



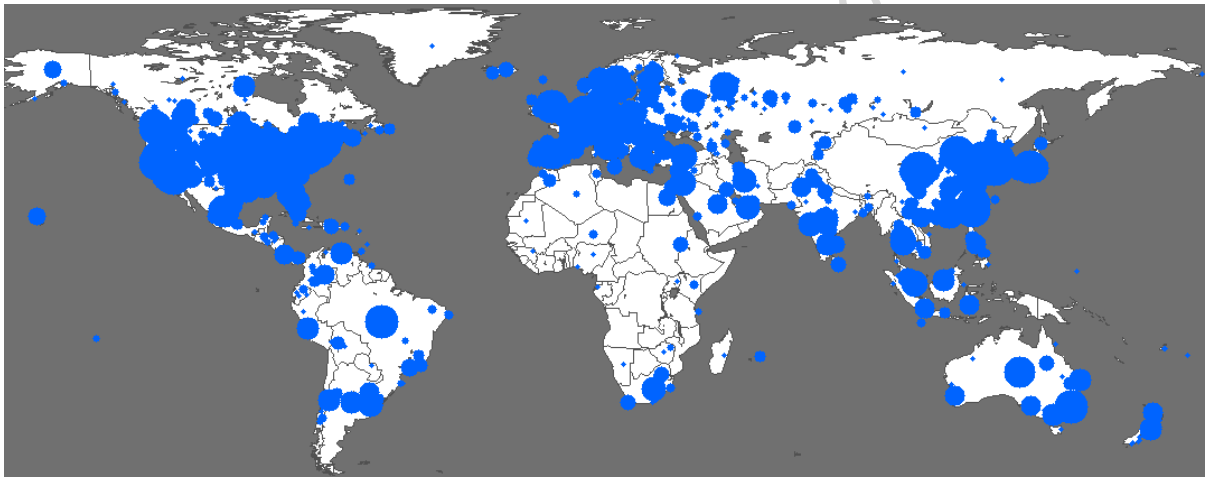
Hình 2.16. Chèn shellcode với phần đệm bằng lệnh NOP (N)

* Ví dụ về khai thác lỗi tràn bộ đệm

Sâu SQL Slammer (một số tài liệu gọi là sâu Sapphire) được phát hiện ngày 25/1/2003 lúc 5h30 (UTC) là sâu có tốc độ lây lan nhanh nhất lúc bấy giờ: nó lây nhiễm ra khoảng 75.000 máy chủ chỉ trong khoảng 30 phút, như minh họa trên Hình 2.17. Sâu Slammer khai thác lỗi tràn bộ đệm trong thành phần Microsoft SQL Server Resolution Service của hệ quản trị cơ sở dữ liệu Microsoft SQL Server 2000.

Sâu sử dụng giao thức UDP với kích thước gói tin 376 byte và vòng lặp chính của sâu chỉ gồm 22 lệnh hợp ngữ. Chu trình hoạt động của sâu SQL Slammer gồm:

- Sinh tự động địa chỉ IP;
- Quét tìm các máy có lỗi với IP tự sinh trên cổng dịch vụ 1434;
- Nếu tìm được, gửi một bản sao của sâu đến máy có lỗi;
- Mã của sâu gây tràn bộ đệm, thực thi mã của sâu và quá trình lặp lại.



Hình 2.17. Bản đồ lây nhiễm sâu Slammer (mầu xanh) theo trang www.caida.org vào ngày 25/1/2003 lúc 6h00 (giờ UTC) với 74.855 máy chủ bị nhiễm

SQL Slammer là sâu "lành tính" vì nó không can thiệp vào hệ thống file, không thực hiện việc phá hoại hay đánh cắp thông tin ở hệ thống bị lây nhiễm. Tuy nhiên, sâu tạo ra lưu lượng mạng khổng lồ trong quá trình lây nhiễm, gây tê liệt đường truyền mạng Internet trên nhiều vùng của thế giới. Do mã của SQL Slammer chỉ được lưu trong bộ nhớ nó gây tràn mà không được lưu vào hệ thống file, nên chỉ cần khởi động lại máy là có thể tạm thời xóa được sâu khỏi hệ thống. Tuy nhiên, hệ thống chứa lỗ hổng có thể bị lây nhiễm lại nếu nó ở gần một máy khác bị nhiễm sâu. Các biện pháp phòng chống triệt để khác là cập nhật bản vá cho bộ phần mềm Microsoft SQL Server 2000. Thông tin chi tiết về sâu SQL Slammer có thể tìm ở các trang: https://technet.microsoft.com/library/security/ms02-039,

hoặc https://www.caida.org/publications/papers/2003/sapphire/sapphire.html.

2.2.3 Phòng chống

Để phòng chống lỗi tràn bộ đệm một cách hiệu quả, cần kết hợp nhiều biện pháp. Các biện pháp có thể thực hiện bao gồm:

- Kiểm tra thủ công mã nguồn hay sử dụng các công cụ phân tích mã tự động để tìm và khắc phục các điểm có khả năng xảy ra lỗi tràn bộ đệm, đặc biệt lưu ý đến các hàm xử lý xâu ký tự.
- Sử dụng cơ chế không cho phép thực hiện mã trong dữ liệu DEP (Data Excution Prevention). Cơ chế DEP được hỗ trợ bởi hầu hết các hệ điều hành (từ Windows XP và các hệ điều hành họ Linux, Unix,…) không cho phép thực hiện mã chương trình chứa trong vùng nhớ dành cho dữ liệu. Như vậy, nếu kẻ tấn công khai thác lỗi tràn bộ đệm, chèn được mã

độc vào bộ đệm trong ngăn xếp, mã độc cũng không thể thực hiện.

- Ngẫu nhiên hóa sơ đồ địa chỉ cấp phát các ô nhớ trong ngăn xếp khi thực hiện chương trình, nhằm gây khó khăn cho việc gỡ rối và phát hiện vị trí các ô nhớ quan trọng như ô nhớ chứa địa chỉ trở về.
- Sử dụng các cơ chế bảo vệ ngăn xếp, theo đó thêm một số ngẫu nhiên (canary) phía trước địa chỉ trở về và kiểm tra số ngẫu nhiên này trước khi trở về chương trình gọi để xác định khả năng bị thay đổi địa chỉ trở về.
- Sử dụng các ngôn ngữ, thư viện và công cụ lập trình an toàn. Trong các trường hợp có thể, sử dụng các ngôn ngữ không gây tràn, như Java, các ngôn ngữ lập trình trên nền Microsoft .Net. Với các ngôn ngữ có thể gây tràn như C, C++, nên sử dụng các thư viện an toàn (Safe C/C++ Libraries) để thay thế các thư viện chuẩn có thể gây tràn.

2.3 Tấn công khai thác lỗi không kiểm tra đầu vào

2.3.1 Giới thiệu

Lỗi không kiểm tra đầu vào (Unvalidated input) là một trong các dạng lỗ hổng bảo mật phổ biến, trong đó ứng dụng không kiểm tra, hoặc kiểm tra không đầy đủ các dữ liệu đầu vào, nhờ đó tin tặc có thể khai thác lỗi để tấn công ứng dụng và hệ thống. Dữ liệu đầu vào (Input data) cho ứng dụng rất đa dạng, có thể đến từ nhiều nguồn với nhiều định dạng khác nhau. Các dạng dữ liệu đầu vào điển hình cho ứng dụng:

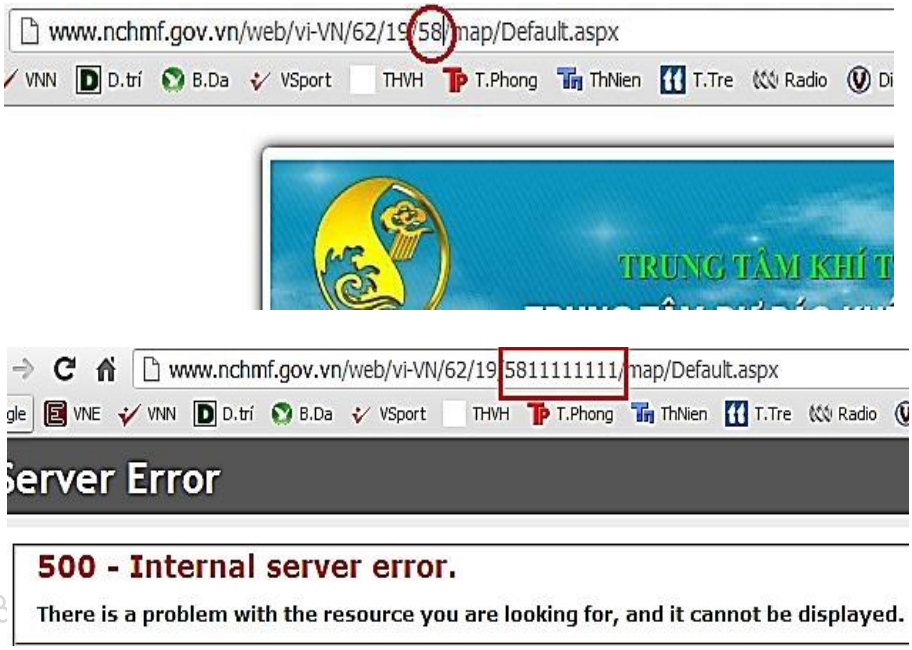
- Các trường dữ liệu văn bản (text);
- Các lệnh được truyền qua địa chỉ URL để kích hoạt chương trình;
- Các file âm thanh, hình ảnh, hoặc đồ họa do người dùng, hoặc các tiến trình khác cung cấp;
- Các đối số đầu vào trong dòng lệnh;
- Các dữ liệu từ mạng hoặc từ các nguồn không tin cậy.

Trên thực tế, tin tặc có thể sử dụng phương pháp thủ công, hoặc tự động để kiểm tra các dữ liệu đầu vào và thử tất cả các khả năng có thể để khai thác lỗi không kiểm tra đầu vào. Theo thống kê của trang web OWASP (http://www.owasp.org), một trang web chuyên về thông kê các lỗi bảo mật ứng dụng web, lỗi không kiểm tra đầu vào luôn chiếm vị trí nhóm dẫn đầu các lỗi bảo mật các trang web trong khoảng 5 năm trở lại đây.

2.3.2 Tấn công khai thác

Có hai dạng chính tấn công khai thác lỗi không kiểm tra đầu vào: (1) cung cấp

dữ liệu quá lớn hoặc sai định dạng để gây lỗi cho ứng dụng, và (2) chèn mã khai thác vào dữ liệu đầu vào để thực hiện trên hệ thống của nạn nhân, nhằm đánh cắp dữ liệu nhạy cảm hoặc thực hiện các hành vi phá hoại. Hình 2.18 minh họa tấn công khai thác lỗi không kiểm tra đầu vào dạng (1) thông qua việc nhập dữ liệu quá lớn, gây lỗi thực hiện cho trang web.



Hình 2.18. Cung cấp dữ liệu quá lớn để gây lỗi cho ứng dụng

Chúng ta minh họa tấn công khai thác lỗi không kiểm tra đầu vào dạng (2) bằng việc chèn mã tấn công SQL vào dữ liệu đầu vào, được thực hiện trên hệ quản trị cơ sở dữ liệu nhằm đánh cắp, hoặc phá hủy dữ liệu trong cơ sở dữ liệu. Giả thiết một trang web tìm kiếm sản phẩm sử dụng câu lệnh SQL sau để tìm kiếm các sản phẩm:

"SELECT * FROM tbl_products WHERE product_name like '%" + keyword + "%'" trong đó *tbl_products* là bảng lưu thông tin các sản phẩm, *product_name* là trường tên
sản phẩm và *keyword* là từ khóa cung cấp từ người dùng form tìm kiếm. Nếu người dùng
nhập từ khóa là "iPhone 7", khi đó câu lệnh SQL trở thành:

"SELECT * FROM tbl_products WHERE product_name like '%iPhone 7%'"

Nếu trong bảng có sản phẩm thỏa mãn điều kiện tìm kiếm, câu lệnh SQL sẽ trả về tập bản ghi. Nếu không có sản phẩm nào thỏa mãn điều kiện tìm kiếm, câu lệnh SQL sẽ trả về tập bản ghi rỗng. Nếu người dùng nhập từ khóa "iPhone 7';*DELETE FROM tbl_products;--*", khi đó câu lệnh SQL trở thành:

"SELECT * FROM tbl_products WHERE product_name like '%iPhone

7';*DELETE FROM tbl_products;*--%'"

Như vậy, câu lệnh SQL được thực hiện trên cơ sở dữ liệu gồm 2 câu lệnh: câu lệnh chọn SELECT ban đầu và câu lệnh xóa DELETE do tin tặc chèn thêm. Câu lệnh "*DELETE FROM tbl_products*" sẽ xóa tất cả các bản ghi trong bảng *tbl_products*. Sở dĩ tin tặc có thể thực hiện điều này là do hầu hết các hệ quản trị cơ sở dữ liệu cho phép thực

hiện nhiều câu lệnh SQL theo *mẻ* (batch), trong đó các câu lệnh được ngăn cách bởi dấu (;). Ngoài ra, dấu "--" ở cuối dữ liệu nhập để loại bỏ hiệu lực của phần lệnh còn lại do "-- " là ký hiệu bắt đầu phần chú thích của dòng lệnh. Ngoài DELETE, tin tặc có thể chèn thêm các lệnh SQL khác, như INSERT, UPDATE để thực hiện việc chèn thêm bản ghi hoặc sửa đổi dữ liệu theo ý đồ tấn công của mình.

2.3.3 Phòng chống

Biện pháp chủ yếu phòng chống tấn công khai thác lỗi không kiểm tra đầu vào là lọc dữ liệu đầu vào. Tất cả các dữ liệu đầu vào, đặc biệt dữ liệu nhập từ người dùng và từ các nguồn không tin cậy cần được kiểm tra kỹ lưỡng. Các biện pháp cụ thể bao gồm:

- Kiểm tra kích thước và định dạng dữ liệu đầu vào;
- Kiểm tra sự hợp lý của nội dung dữ liệu;
- Tạo các bộ lọc để lọc bỏ các ký tự đặc biệt và các từ khóa của các ngôn ngữ trong các trường hợp cần thiết mà kẻ tấn công có thể sử dụng:

+ Các ký tự đặc biệt: *, ', =, --

+ Các từ khóa ngôn ngữ: SELECT, INSERT, UPDATE, DELETE, DROP,.... (với dạng tấn công chèn mã SQL).

2.4 Tấn công chèn mã SQL

2.4.1 Khái quát

Tấn công chèn mã SQL (SQL Injection) là một kỹ thuật cho phép kẻ tấn công chèn mã SQL vào dữ liệu gửi đến máy chủ và cuối cùng được thực hiện trên máy chủ cơ sở dữ liệu. Tùy vào mức độ tinh vi, tấn công chèn mã SQL có thể cho phép kẻ tấn công (1) vượt qua các khâu xác thực người dùng, (2) chèn, sửa đổi, hoặc xóa dữ liệu, (3) đánh cắp các thông tin trong cơ sở dữ liệu và (4) chiếm quyền điều khiển hệ thống máy chủ cơ sở dữ liệu. Tấn công chèn mã SQL là dạng tấn công thường gặp ở các ứng dụng web, các trang web có kết nối đến cơ sở dữ liệu.

Có 2 nguyên nhân của lỗ hổng trong ứng dụng cho phép thực hiện tấn công chèn mã SQL:

- Dữ liệu đầu vào từ người dùng hoặc từ các nguồn khác không được kiểm tra hoặc kiểm tra không kỹ lưỡng;
- Sử dụng các câu lệnh SQL động trong ứng dụng, trong đó có thao tác nối dữ liệu người dùng với mã lệnh SQL gốc.

2.4.2 Vượt qua các khâu xác thực người dùng

Xem xét một form đăng nhập (Log in) và đoạn mã xử lý xác thực người dùng lưu trong bảng cơ sở dữ liệu tbl_accounts(username, password) cho như trên Hình 2.19.

Nếu người dùng nhập 'admin' vào trường *username* và 'abc123' vào trường *password* của form, mã xử lý hoạt động đúng: Nếu tồn tại người dùng với *username* và *password* kể trên, hệ thống sẽ cho phép đăng nhập với thông báo đăng nhập thành công; Nếu không tồn tại người dùng với *username* và *password* đã cung cấp, hệ thống sẽ từ chối đăng nhập và trả lại thông báo lỗi. Tuy nhiên, nếu người dùng nhập *aaaa' OR 1=1--* vào trường

username và một chuỗi bất kỳ, chẳng hạn 'aaaa' vào trường *password* của form, mã xử lý hoạt động sai và chuỗi chứa câu truy vấn SQL trở thành:

```
SELECT * FROM tbl_accounts WHERE username='aaaa' OR 1=1--' AND password='aaaa'
```

```
<!— Form đăng nhập -->
<form method="post" action="/log_in.asp">
Tên đăng nhập: <input type=text name="username"><br
\> Mật khẩu: <input type=password
name="password"><br \>
<input type=submit name="login" value="Log In">
</form>
<%
' Mã ASP xử lý đăng nhập trong file log_in.asp:
' giả thiết đã kết nối với CSDL SQL qua đối tượng conn
và bảng tbl_accounts lưu thông tin người dùng
Dim username, password, sqlString,
rsLogin ' lấy dữ liệu từ form
username                    =
Request.Form("username")
password                    =
Request.Form("password") ' tạo
và thực hiện câu truy vấn sql
sqlString = "SELECT * FROM tbl_accounts WHERE
username='" & username & "' AND password = '" &
password & "'"
set rsLogin =
conn.execute(sqlString) if (NOT
rsLogin.eof()) then
' cho phép đăng nhập, bắt đầu phiên làm việc
else
' từ chối đăng nhập, báo
lỗi end if
%>
```

Hình 2.19. Form đăng nhập (log on) và đoạn mã xử lý xác thực người dùng

Câu truy vấn sẽ trả về mọi bản ghi trong bảng do thành phần OR 1=1 làm cho điều kiện trong mệnh đề WHERE trở lên luôn đúng và phần kiểm tra mật khẩu đã bị loại bỏ bởi ký hiệu (--). Phần lệnh sau ký hiệu (--) được coi là ghi chú và không được thực hiện. Nếu trong bảng tbl_accounts có chứa ít nhất một bản ghi, kẻ tấn công sẽ luôn đăng nhập thành công vào hệ thống.

2.4.3 Chèn, sửa đổi, hoặc xóa dữ liệu

Xem xét một form tìm kiếm sản phẩm và đoạn mã xử lý tìm sản phẩm lưu trong bảng cơ sở dữ liệu tbl_products(product_id, product_name, product_desc,

product_cost) cho như trên Hình 2.20.

```
<!– Form tìm kiếm sản phẩm -->
<form method="post" action="/search.asp">
Nhập tên sản phẩm: <input type=text name="keyword">
<input type=submit name="search" value="Search">
</form>

<%
' Mã ASP xử lý tìm sản phẩm trong file search.asp:
' giả thiết đã kết nối với CSDL SQL server qua
connection ' conn và bảng tbl_products lưu thông tin
sản phẩm
Dim keyword, sqlString,
rsSearch ' lấy dữ liệu từ
form
keyword =
Request.Form("keyword") ' tạo
và thực hiện câu truy vấn SQL
sqlString = "SELECT * FROM tbl_products WHERE
product_name like '%" & keyword & "%'"
set rsSearch =
conn.execute(sqlString) if (NOT
rsSearch.eof()) then
' hiển thị danh sách các sản phẩm
else
' thông báo không tìm thấy sản
phẩm end if
%>
```

Hình 2.20. Form tìm kiếm sản phẩm và đoạn mã xử lý tìm sản phẩm

Nếu người dùng nhập chuỗi "Samsung Galaxy S8" vào trường *keyword* của form, mã xử lý hoạt động đúng: Nếu tìm thấy các sản phẩm có tên chứa từ khóa, hệ thống sẽ hiển thị danh sách các sản phẩm tìm thấy; Nếu không tìm thấy sản phẩm nào có tên chứa từ khóa, hệ thống thông báo không tìm thấy sản phẩm. Tuy nhiên, nếu người dùng nhập chuỗi "Samsung Galaxy S8';DELETE FROM tbl_products;--" vào trường *keyword* của form, mã xử lý sẽ hoạt động sai và chuỗi chứa câu truy vấn SQL trở thành:

SELECT * FROM tbl_products WHERE keyword like '%Samsung Galaxy

S8';DELETE FROM tbl_products;--%'

Chuỗi lệnh SQL mới gồm 2 lệnh SQL: câu lệnh SELECT tìm kiếm các sản phẩm có tên chứa từ khóa "Samsung Galaxy S8" trong bảng tbl_products và câu lệnh DELETE xóa tất cả các sản phẩm trong bảng tbl_products. Sở dĩ kẻ tấn công có thể làm được điều này là do hệ quản trị cơ sở dữ liệu MS-SQL server nói riêng và hầu hết các hệ quản trị cơ sở dữ liệu nói chung cho phép thực hiện nhiều lệnh SQL theo lô và dùng dấu ; để ngăn cách các lệnh. Ký hiệu -- dùng để hủy tác dụng của phần lệnh còn lại nếu có.

Bằng thủ thuật tương tự, kẻ tấn công có thể thay lệnh DELETE bằng lệnh UPDATE hoặc INSERT để chỉnh sửa, hoặc chèn thêm dữ liệu. Chẳng hạn, kẻ tấn công chèn thêm lệnh UPDATE để cập nhật mật khẩu của người quản trị bằng cách nhập chuỗi sau làm từ khóa tìm kiếm (giả thiết bảng tbl_administrators chứa thông tin người quản trị):

Galaxy S8';UPDATE tbl_administrators SET password=abc123 WHERE username = 'admin';--

Hoặc kẻ tấn công có thể chèn thêm bản ghi vào bảng tbl_administrators bằng cách nhập chuỗi sau làm từ khóa tìm kiếm:

Galaxy S8';INSERT INTO tbl_administrators (username, password) VALUES ('attacker', 'abc12345');--

2.4.4 Đánh cắp các thông tin trong cơ sở dữ liệu

Lỗ hổng chèn mã SQL có thể giúp kẻ tấn công đánh cắp dữ liệu trong cơ sở dữ liệu thông qua một số bước như sau:

- Tìm lỗ hổng chèn mã SQL và thăm dò các thông tin về hệ quản trị cơ sở dữ liệu:

+ Nhập một số dữ liệu mẫu để kiểm tra một trang web có chứa lỗ hổng chèn mã SQL, như các dấu nháy đơn, dấu --,…

+ Tìm phiên bản máy chủ cơ sở dữ liệu: nhập các câu lệnh lỗi và kiểm tra thông báo lỗi, hoặc sử dụng @@version (với MS-SQL Server), hoặc version() (với MySQL) trong câu lệnh ghép với UNION SELECT.

- Tìm thông tin về số lượng và kiểu dữ liệu các trường của câu truy vấn hiện tại của trang web.

+ Sử dụng mệnh đề ORDER BY <số thứ tự của trường>

+ Sử dụng UNION SELECT 1, 2, 3, …

- Trích xuất thông tin về các bảng, các trường của cơ sở dữ liệu thông qua các bảng hệ thống (metadata).
- Sử dụng lệnh UNION SELECT để ghép các thông tin định trích xuất

vào câu truy vấn hiện tại của ứng dụng.

2.4.5 Chiếm quyền điều khiển hệ thống máy chủ cơ sở dữ liệu

Khả năng máy chủ cơ sở dữ liệu bị chiếm quyền điều khiển xảy ra khi trang web tồn tại đồng thời 2 lỗ hổng: (1) lỗ hổng cho phép tấn công chèn mã SQL và (2) lỗ hổng thiết lập quyền truy nhập cơ sở dữ liệu – sử dụng người dùng có quyền quản trị để truy nhập và thao tác dữ liệu của website. Khai thác 2 lỗ hổng này, kẻ tấn công có thể gọi thực hiện các lệnh hệ thống của máy chủ cơ sở dữ liệu cho phép can thiệp sâu vào cơ sở dữ liệu, hệ quản trị cơ sở dữ liệu và cả hệ điều hành trên máy chủ. Chẳng hạn, hệ quản trị cơ sở dữ liệu MS-SQL Server cung cấp thủ tục *sp_send_dbmail* cho phép gửi email từ máy chủ cơ sở dữ liệu và thủ tục *xp_cmdshell* cho phép chạy các lệnh và chương trình cài đặt trên hệ điều hành Microsoft Windows. Sau đây là một số ví dụ chạy các lệnh Microsoft Windows thông qua thủ tục *xp_cmdshell*:

EXEC xp_cmdshell 'dir *.exe' : liệt kê nội dung thư mục hiện thời

EXEC xp_cmdshell 'shutdown /s /t 00' : tắt máy chủ nền chạy hệ quản trị CSDL EXEC xp_cmdshell 'net stop W3SVC' : dừng hoạt động máy chủ web

EXEC xp_cmdshell 'net stop MSSQLSERVER' : dừng hoạt động máy chủ CSDL Ngoài ra, kẻ tấn công có thể thực hiện các thao tác nguy hiểm đến cơ sở dữ liệu nếu có quyền của người quản trị cơ sở dữ liệu hoặc quản trị hệ thống, như:

Xóa cả bảng (gồm cả cấu trúc): DROP TABLE <tên bảng> Xóa cả cơ sở dữ liệu: DROP DATABASE <tên CSDL>

Tạo 1 tài khoản mới truy nhập CSDL: sp_addlogin <username> <password> Đổi mật khẩu tài khoản truy nhập CSDL: sp_password <password>

2.4.6 Phòng chống

Do tính chất nguy hiểm của tấn công chèn mã SQL, nhiều giải pháp đã được đề xuất nhằm hạn chế tác hại và ngăn chặn triệt để dạng tấn công này. Nhìn chung, cần áp dụng kết hợp các biện pháp phòng chống tấn công chèn mã SQL để đảm bảo an toàn cho hệ thống. Các biện pháp, kỹ thuật cụ thể có thể áp dụng gồm:

- Các biện pháp phòng chống dựa trên kiểm tra và lọc dữ liệu đầu vào:

+ Kiểm tra tất cả các dữ liệu đầu vào, đặc biệt dữ liệu nhập từ người dùng và từ các nguồn không tin cậy;

+ Kiểm tra kích thước và định dạng dữ liệu đầu vào;

+ Tạo các bộ lọc để lọc bỏ các ký tự đặc biệt (như *, ‘, =, --) và các từ khóa của ngôn ngữ SQL (SELECT, INSERT, UPDATE, DELETE, DROP,.) mà kẻ tấn công có thể sử dụng:

- Sử dụng thủ tục cơ sở dữ liệu (stored procedures) và cơ chế tham số hóa dữ liệu:

+ Đưa tất cả các câu truy vấn (SELECT) và cập nhật, sửa, xóa dữ liệu (INSERT, UPDATE, DELETE) vào các thủ tục. Dữ liệu truyền vào thủ tục thông qua các tham số, giúp tách dữ liệu khỏi mã lệnh SQL, nhờ đó hạn ngăn chặn hiệu quả tấn công chèn mã SQL;

+ Hạn chế thực hiện các câu lệnh SQL động trong thủ tục;

+ Sử dụng cơ chế tham số hóa dữ liệu hỗ trợ bởi nhiều ngôn ngữ lập trình web như ASP.NET, PHP và JSP.

- Các biện pháp phòng chống dựa trên thiết lập quyền truy nhập người dùng cơ sở dữ liệu:

+ Không sử dụng người dùng có quyền quản trị hệ thống hoặc quản trị cơ sở dữ liệu làm người dùng truy nhập dữ liệu. Ví dụ: không dùng người dùng *sa* (Microsoft SQL) hoặc *root* (MySQL) làm người dùng truy nhập dữ liệu. Chỉ dùng các người dùng này cho mục đích quản trị.

+ Chia nhóm người dùng, chỉ cấp quyền vừa đủ để truy nhập các bảng biểu, thực hiện câu truy vấn và chạy các thủ tục.

+ Tốt nhất, không cấp quyền thực hiện các câu truy vấn, cập nhật, sửa, xóa trực tiếp trên các bảng dữ liệu. Thủ tục hóa tất cả các câu lệnh và chỉ cấp quyền thực hiện thủ tục.

+ Cấm hoặc vô hiệu hóa (disable) việc thực hiện các thủ tục hệ thống (các thủ tục cơ sở dữ liệu có sẵn) cho phép can thiệp vào hệ quản trị cơ sở dữ liệu và hệ điều hành nền.

Sử dụng các công cụ rà quét lỗ hổng chèn mã SQL, như SQLMap, hoặc Acunetix Vulnerability Scanner để chủ động rà quét, tìm các lỗ hổng chèn mã SQL và có biện pháp khắc phục phù hợp

3. Tấn công kiểu người đứng giữa

3.1 Giới thiệu

Tấn công kiểu người đứng giữa (Man in the middle) là dạng tấn công dụng quá trình chuyển gói tin đi qua nhiều trạm (hop) thuộc các mạng khác nhau, trong đó kẻ tấn công chặn bắt các thông điệp giữa 2 bên tham gia truyền thông và chuyển

thông điệp lại cho bên kia. Mục đích chính của dạng tấn công này là đánh cắp thông tin. Hình 2.27 minh họa mô hình tấn công kiểu người đứng giữa trong một phiên truyền file ở dạng rõ (plaintext) sử dụng giao thức FTP giữa máy khách (Client) và máy chủ (Server).

Hình 2.27. Mô hình tấn công kiểu người đứng giữa

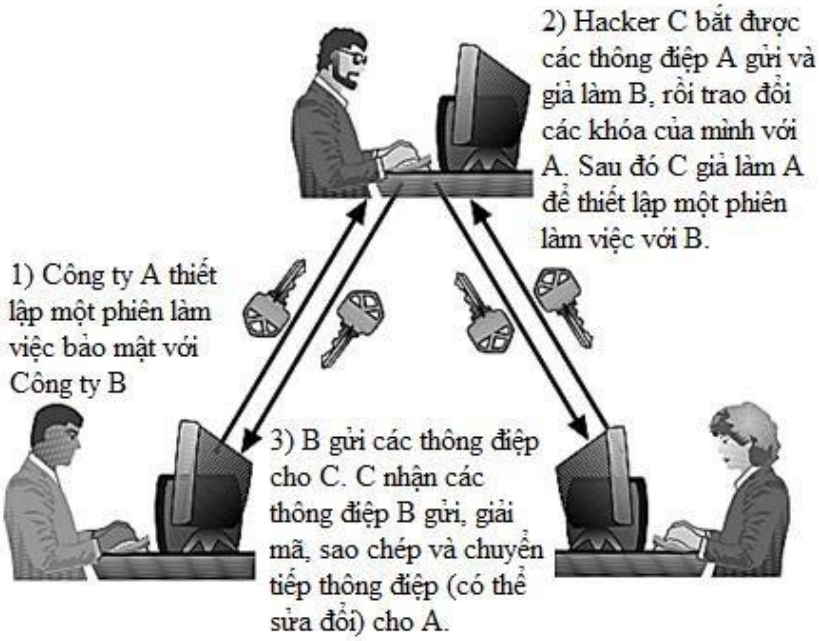
3.2 Kịch bản

Hình 2.28 minh họa một kịch bản tấn công kiểu người đứng giữa, trong đó hai bên A và B (Công ty A và Công ty B) trao đổi các thông điệp bí mật và kẻ tấn công C (Hacker) bắt và có thể sửa đổi, lạm dụng các thông điệp truyền giữa A và B. Các bước tấn công cụ thể như sau:

- A gửi các thông điệp để thiết lập một phiên làm việc bảo mật với B;
- C bắt được các thông điệp A gửi. C giả làm B và trao đổi các khóa của mình với A. Sau đó C giả làm A để thiết lập một phiên làm việc với B (có trao đổi khóa với B);
- B gửi các thông điệp cho C mà vẫn tưởng như đang liên lạc với A. C nhận các thông điệp B gửi, giải mã bằng khóa của mình (và có thể sửa đổi), sau đó chuyển tiếp thông điệp cho A. A nhận các thông điệp mà không biết là chúng đã bị C lạm dụng.

3.3 Phòng chống

Một trong các biện pháp hiệu quả để phòng chống tấn công kiểu người đứng giữa là hai bên tham gia truyền thông phải có cơ chế xác thực thông tin nhận dạng của nhau và xác thực tính toàn vẹn của các thông điệp trao đổi. Chẳng hạn, các bên có thể sử dụng chứng chỉ số khóa công khai (Public key certificate) để xác thực thông tin nhận dạng của nhau và sử dụng chữ ký số để đảm bảo tính toàn vẹn của các thông điệp trao đổi.



Hình 2.28. Một kịch bản tấn công kiểu người đứng giữa

4. Tấn công bằng bom thư và thư rác

Tấn công bằng bom thư (Mail bombing) là một dạng tấn công DoS khi kẻ tấn công gửi một lượng rất lớn email đến hộp thư của nạn nhân. Khi đó hộp thư và cả máy chủ nạn nhân có thể bị tê liệt và không thể hoạt động bình thường. Tấn công bằng bom thư có thể được thực hiện bằng một số thủ thuật:

- Gửi bom thư bằng cách sử dụng kỹ thuật xã hội, đánh lừa người dùng phát tán email;
- Khai thác lỗi trong hệ thống gửi nhận email SMTP;
- Lợi dụng các máy chủ email không được cấu hình tốt để gửi email (relay) cho chúng.

Tấn công bằng thư rác (Spamming email) là dạng tấn công gửi các thư không mong muốn, như thư quảng cáo, thư chứa các phần mềm độc hại. Theo một số thống kê,

khoảng 70-80% lượng email gửi trên mạng Internet là thư rác. Kẻ tấn công thường sử dụng các máy tính bị điều khiển (bot/zombie) để gửi email cho chúng. Spam email gây lãng phí tài nguyên tính toán và thời gian của người dùng.

5. Tấn công sử dụng các kỹ thuật xã hội

5.1 Giới thiệu

Tấn công sử dụng các kỹ thuật xã hội (Social Engineering) là dạng tấn công phi

kỹ thuật nhằm vào người dùng. Dạng tấn công này khai thác các điểm yếu cố hữu của người dùng, như tính cả tin, ngây thơ, tò mò và lòng tham. Dạng thường gặp của kiểu tấn công này là thuyết phục người dùng tiết lộ thông tin truy nhập hoặc các thông tin có giá trị cho kẻ tấn công. Một số kỹ thuật mà kẻ tấn công thường áp dụng gồm:

- Kẻ tấn công có thể giả danh làm người có vị trí cao hơn so với nạn nhân để có được sự tin tưởng, từ đó thuyết phục hoặc đánh lừa nạn nhân cung cấp thông tin;
- Kẻ tấn công có thể mạo nhận là người được ủy quyền của người có thẩm quyền để yêu cầu các nhân viên tiết lộ thông tin về cá nhân/tổ chức;
- Kẻ tấn công có thể lập trang web giả để đánh lừa người dùng cung cấp các thông tin cá nhân, thông tin tài khoản, thẻ tín dụng,…

5.2 Trò lừa đảo Nigeria 4-1-9

Trò lừa đảo Nigeria 4-1-9 là một trong các dạng tấn công sử dụng các kỹ thuật xã hội nổi tiếng nhất, trong đó đã có hàng chục nghìn người ở Mỹ, Canada và Châu Âu đã sập bẫy của kẻ lừa đảo. Kẻ lừa đảo lợi dụng sự ngây thơ và lòng tham của một số người với kịch bản tóm tắt như sau:

- Kẻ lừa đảo gửi thư tay, hoặc email đến nhiều người nhận, mô tả về việc có 1 khoản tiền lớn (từ thừa kế, hoặc lợi tức,..) cần chuyển ra nước ngoài, nhờ người nhận giúp đỡ để hoàn thành giao dịch. Khoản tiền có thể lên đến hàng chục, hoặc trăm triệu USD. Kẻ lừa đảo hứa sẽ trả cho người tham gia một phần số tiền (lên đến 20- 30%);
- Nếu người nhận có phản hồi và đồng ý tham gia, kẻ lừa đảo sẽ gửi tiếp thư, hoặc email khác, yêu cầu chuyển cho hắn 1 khoản phí giao dịch (từ vài ngàn đến hàng chục ngàn USD);
- Nếu người nhận gửi tiền phí giao dịch theo yêu cầu thì người đó sẽ mất tiền, do giao dịch mà kẻ lừa đảo hứa hẹn là giả mạo.

Nhiều biến thể của trò lừa đảo Nigeria 4-1-9 đã xuất hiện trong những năm gần đây trên thế giới cũng như ở Việt Nam, chẳng hạn như thông báo lừa trúng thưởng các tài sản có giá trị lớn để chiếm đoạt khoản "phí trả thưởng", lừa đầu tư vào tài khoản ảo với hứa hẹn lãi suất cao,…

5.3 Phishing

Phishing là một dạng đặc biệt phát triển rất mạnh của tấn công sử dụng các kỹ thuật xã hội, trong đó kẻ tấn công bẫy người dùng để lấy thông tin cá nhân, thông tin tài khoản, thẻ tín dụng,… Kẻ tấn công có thể giả mạo trang web của các tổ chức tài chính, ngân hàng, sau đó chúng gửi email cho người dùng (địa chỉ email

thu thập trên mạng), yêu cầu

xác thực thông tin. Hình 2.29 và Hình 2.30 minh họa 2 phishing email gửi cho khách hàng của mạng đấu giá trực tuyến eBay và ngân hàng Royal Bank yêu cầu người dùng cập nhật thông tin thanh toán đã hết hạn, hoặc xác nhận thông tin tài khoản không sử dụng. Nếu người dùng làm theo hướng dẫn thì sẽ vô tình cung cấp các thông tin cá nhân, thông tin tài khoản, thẻ tín dụng cho kẻ tấn công.

Hình 2.29. Một phishing email gửi cho khách hàng của mạng đấu giá eBay

Hình 2.30. Một phishing email gửi cho khách hàng của ngân hàng Royal Bank

5.4 Phòng chống

Do tấn công sử dụng các kỹ thuật xã hội nhắm đến người dùng nên biện pháp phòng chống hiệu quả là giáo dục, đào tạo nâng cao ý thức cảnh giác cho người dùng. Một số khuyến nghị giúp người dùng phòng tránh dạng tấn công này:

- Cảnh giác với các lời mời, hoặc thông báo trúng thưởng bằng email, tin nhắn điện thoại, hoặc quảng cáo trên các trang web, diễn đàn mà không có lý do, nguồn gốc trúng thưởng rõ ràng;
- Cảnh giác với các yêu cầu cung cấp thông tin, xác nhận tài khoản, thông tin thanh toán, thông tin thẻ tín dụng,..;
- Kiểm tra kỹ địa chỉ (URL) các trang web, đảm bảo truy nhập đúng trang web của cơ quan, tổ chức.

6. Tấn công pharming

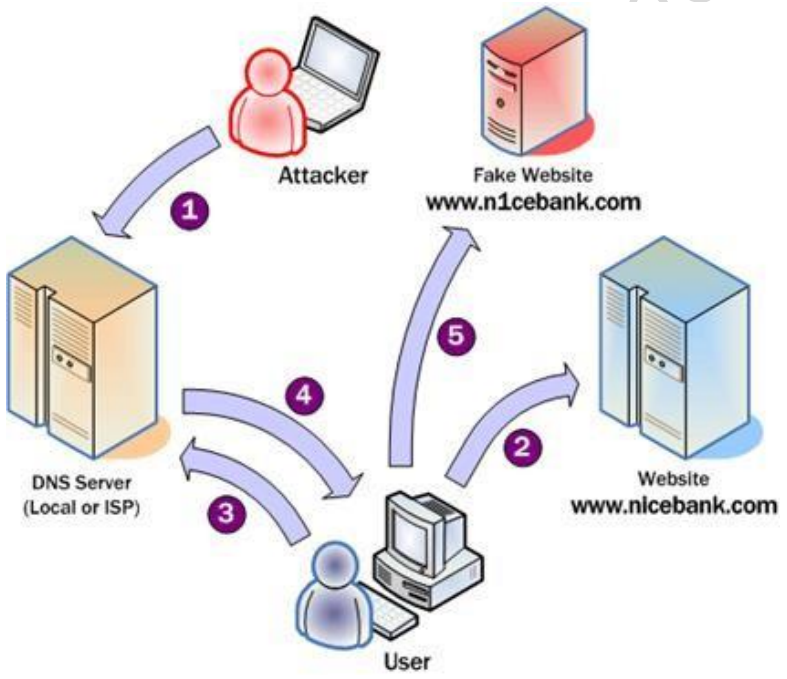
Pharming là kiểu tấn công vào trình duyệt của người dùng, trong đó người dùng gõ địa chỉ 1 website, trình duyệt lại yêu cầu và tải 1 website khác, thường là website độc hại. Có 2 dạng tấn công pharming: (1) kẻ tấn công thường sử dụng sâu, vi rút hoặc các phần mềm độc hại cài vào hệ thống để điều khiển trình duyệt của người dùng và (2) kẻ tấn công có thể tấn công vào hệ thống tên miền (DNS) để thay đổi kết quả truy vấn: thay địa chỉ IP của website hợp pháp thành IP của website độc hại.

Hình 2.31. Tấn công pharming "cướp" trình duyệt

Hình 2.31 minh họa cửa sổ trình duyệt của người dùng bị tấn công pharming ở dạng (1), hay còn gọi là *tấn công cướp trình duyệt* (Browser hijacking), trong đó người dùng nhập địa chỉ trang google.com thì trình duyệt lại nạp trang adventureinsecurity.com. Trong trường hợp này, trình duyệt của nạn nhân đã bị cài đặt trình cắm (plug-in, hoặc add-on) độc hại có khả năng điều khiển trình duyệt.

Hình 2.32 minh họa các bước của tấn công pharming dạng (2), trong đó kẻ tấn công xâm nhập vào máy chủ DNS chỉnh sửa địa chỉ IP của website hợp pháp thành địa chỉ IP

của máy chủ của chúng. Kết quả là trình duyệt người dùng bị chuyển hướng yêu cầu nạp website của kẻ tấn công.



Hình 2.32. Tấn công pharming thông qua tấn công vào máy chủ DNS