

TÌM HIỂU VỀ AN TOÀN THÔNG TIN

NỘI DUNG:

Tìm hiểu các kiến thức cơ bản liên quan đến ngành an toàn thông tin như: SQL Injection, Buffer Overflow, Malware bao gồm:

- Giới thiệu, định nghĩa
- Tìm hiểu về cách thực hiện, các dạng tấn công...
- Cách phòng chống

I. SQL Injection

1. SQL Injection là gì?

- SQL Injection là kỹ thuật tấn công sử dụng các câu lệnh SQL chèn vào phần input của website, từ đó làm thay đổi kết quả trả về câu truy vấn. Thông qua đó, hacker có thể lấy được các thông tin quan trọng từ website như username, password của trang quản trị, các tài khoản của khách hàng,...
- Lỗi này thường xảy ra trên các ứng dụng web có dữ liệu được quản lý bằng các hệ quản trị cơ sở dữ liệu như MySQL, SQL Server, Oracle, DB2, Sysbase.

2. Các dạng tấn công bằng SQL Injection

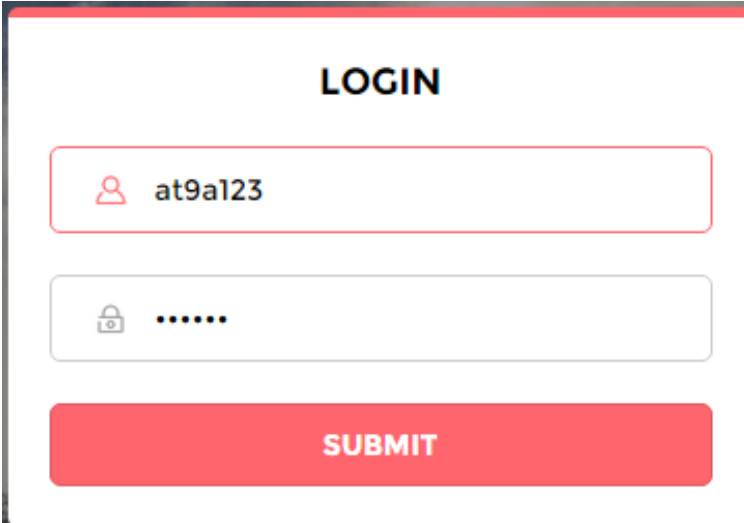
Có 4 dạng thông thường bao gồm

- Vượt qua kiểm tra lúc đăng nhập
- Sử dụng câu lệnh SELECT
- Sử dụng câu lệnh INSERT
- Sử dụng các stored-procedures

a) Dạng tấn công vượt qua lúc kiểm tra đăng nhập

Đăng nhập nhờ vào lỗi khi dùng các câu lệnh SQL thao tác trên cơ sở dữ liệu của ứng dụng web. Thông thường để cho phép người dùng truy cập vào các trang web được bảo mật, hệ thống thường xây dựng trang đăng nhập để yêu cầu người dùng nhập thông tin về tên đăng nhập và mật khẩu.

Ví dụ:



The image shows a web form titled "LOGIN". It contains two input fields: the first for a username, which has the text "at9a123" entered, and the second for a password, which has six dots "....." entered. Below these fields is a red button labeled "SUBMIT".

Ta có đoạn code xử lý:

```
<?php
$username = isset($_POST['username'])? $_POST['username']: "...";
$password = isset($_POST['password'])? $_POST['password']: "...";

$query = "SELECT * FROM tbl_users WHERE
username = '"+$username+"'
        AND password = '"+$password+"'";
        ....
?>
```

- Trường hợp 1:

Người dùng login với username = at9a123 và password=7899661.

Câu truy vấn lúc này:

```
SELECT * FROM tbl_user WHERE username = 'at9a123'
        AND password = '7989661'
```

- Trường hợp 2:

Hacker login với username = **admin'or 1=1-- -** và password = **123456**

Câu truy vấn lúc này:

```
SELECT * FROM tbl_user WHERE username = 'admin'or 1=1-- -
        AND password = '123456'
```

- Giải thích:

SQL check username='admin' là False; 1=1 là True;

False or True = True;

Dấu -- - là comment trong SQL;

➔ Hacker có thể đăng nhập vào mà không cần tài khoản.

b) Dạng tấn công sử dụng câu lệnh SELECT

Để thực hiện được kiểu tấn công này, kẻ tấn công phải có khả năng hiểu và lợi dụng các sơ hở trong các thông báo lỗi từ hệ thống để dò tìm các điểm yếu khởi đầu cho việc tấn công. Thông thường, sẽ có một trang nhận ID của tin cần hiển thị rồi sau đó truy vấn nội dung của tin có ID này.

Xét site có dạng: `http://site.com/chitiet.php?id=123`

Có chứa điểm yếu SQL Injection

- Thông thường đoạn mã này trả về thông tin chi tiết của sản phẩm có mã 123.
- Tuy nhiên, Nếu thay đổi như sau: `Id=0 or 1=1` thì lúc này sẽ đưa ra toàn bộ thông tin của bảng chứa sản phẩm vì câu sql bây giờ là:

`SELECT * FROM sanpham WHERE id =0 or 1=1`

Dạng tấn công này phức tạp hơn. Để thực hiện được kiểu tấn công này, kẻ tấn công phải có khả năng hiểu và lợi dụng các sơ hở trong các thông báo lỗi từ hệ thống để dò tìm các điểm yếu khởi đầu cho việc tấn công. Khi đó hacker sẽ lợi dụng lỗi của site sẽ khai thác và lấy thông tin như : table, columns ..., hoặc hiệu chỉnh, xóa dữ liệu bằng các câu lệnh SQL. Tấn công kiểu select này tuy phức tạp nhưng thường được hacker sử dụng, hacker thường khai thác lỗi này để lấy cấp tài khoản chủ hoặc chiếm quyền Admin của một website nào đó.

c) Dạng tấn công sử dụng câu lệnh INSERT

Chức năng không thể thiếu là sau khi đăng kí thành công, người dùng có thể xem và hiệu chỉnh thông tin của mình. SQL injection có thể được dùng khi hệ thống không kiểm tra tính hợp lệ của thông tin nhập vào.

Ví dụ: Một câu lệnh INSERT có thể có cú pháp dạng:

```
INSERT INTO TableName
```

```
VALUES('Value One', 'Value Two', 'Value Three')
```

Nếu đoạn mã xây dựng câu lệnh SQL có dạng:

```
<%  
    strSQL = "INSERT INTO TableName  
VALUES(' " & strValueOne & " ',  
' " _ & strValueTwo & " ', ' " & strValueThree & " ') “  
  
    Set objRS = Server.CreateObject("ADODB.Recordset") objRS.Open  
    strSQL, "DSN=..."  
  
    ...  
    Set objRS = Nothing %>
```

Nếu ta nhập vào trường thứ nhất có dạng:

‘+(SELECT TOP 1 FieldName FROM TableName)+’

Lúc này câu truy vấn là:

```
INSERT INTO TableName  
VALUES(‘+(SELECT TOP 1 FieldName FROM TableName)+’,’abc’,’cds’)
```

Khi đó, lúc thực hiện lệnh xem thông tin, xem như bạn đã thực hiện thêm 1 lệnh nữa là:

SELECT TOP 1 FieldName FROM TableName

d) Dạng tấn công sử dụng stored-procedures

Việc tấn công bằng stored-procedures sẽ gây tác hại rất lớn nếu ứng dụng được thực thi với quyền quản trị hệ thống.

Ví dụ, nếu ta thay đoạn mã tiêm vào dạng: ' ;

EXEC xp_cmdshell 'cmd.exe dir C: '.

Lúc này hệ thống sẽ thực hiện lệnh liệt kê thư mục trên ổ đĩa C:\ cài đặt server. Việc phá hoại kiểu nào tùy thuộc vào câu lệnh đăng sau cmd.exe.

3. Kỹ thuật tấn công bằng SQL Injection

- Các bước tiến hành:
B1: Tìm kiếm mục tiêu.
B2: Kiểm tra chỗ yếu của trang web.
B3: Khai thác các thông tin.
B4: Xử lý kết quả tìm được.

a) Tìm kiếm mục tiêu

- Bạn có thể dùng các bất kỳ một search-engine nào trên mạng như các trang login, search, feedback...
- Bạn có thể “custome Search Engine” lại cho phù hợp với yêu cầu của bạn.

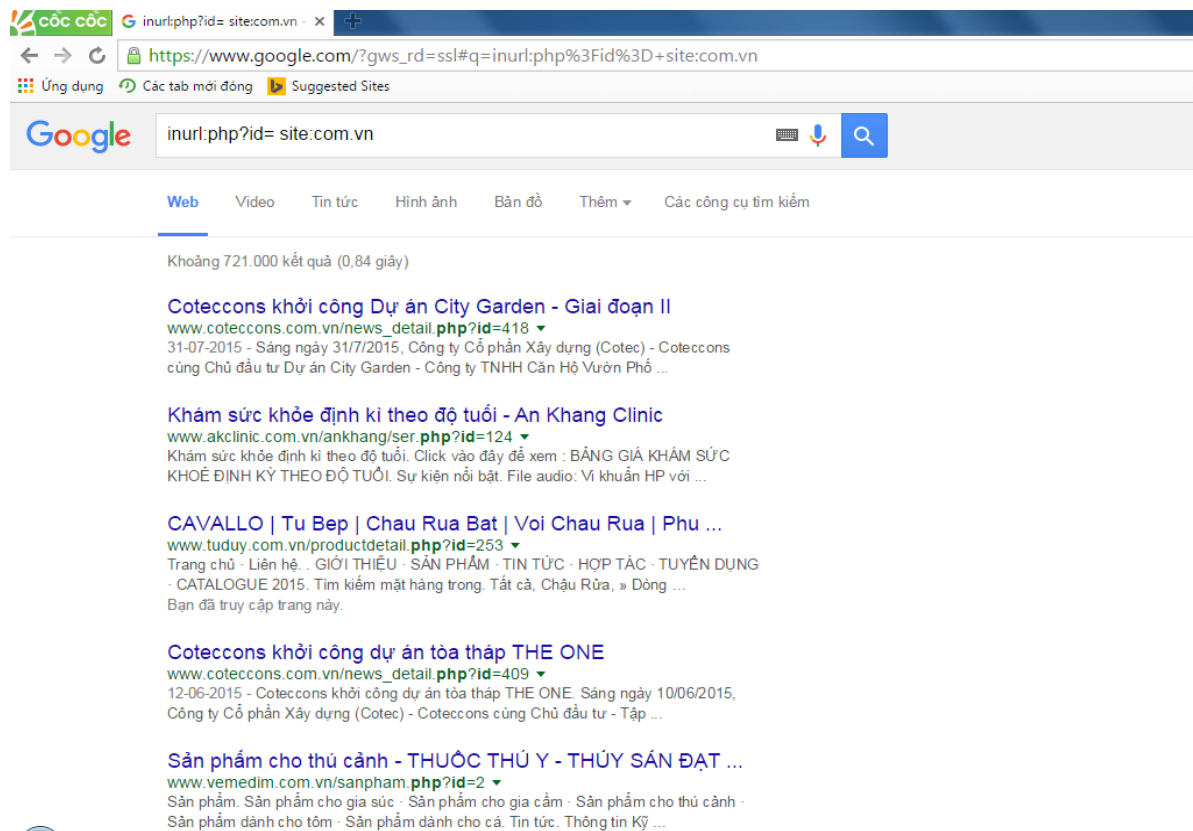
Thực hiện search:

Inurl:php?id= site:com.vn

Inurl: sanpham.php?id=

Inurl: products_detail.php?id=

...



Kết quả thu được 1 số site nằm trong nhóm tính nghi:

[http://www.tuduy.com.vn/productdetail.php?id=253'](http://www.tuduy.com.vn/productdetail.php?id=253)
[http://www.akclinic.com.vn/ankhang/ser.php?id=124'](http://www.akclinic.com.vn/ankhang/ser.php?id=124)
[http://www.vemedim.com.vn/chitiettt.php?id=72'](http://www.vemedim.com.vn/chitiettt.php?id=72)
[http://www.shugahaircare.com/product_details.php?id=9'](http://www.shugahaircare.com/product_details.php?id=9)
[http://www.sieuthibexinh.com/chitiet.php?id=179'](http://www.sieuthibexinh.com/chitiet.php?id=179)
[http://thanhloisteel.com/chitietsanpham.php?id=7'](http://thanhloisteel.com/chitietsanpham.php?id=7)
[http://aladinphoto.com/sanpham.php?id=5'](http://aladinphoto.com/sanpham.php?id=5)
[http://www.daotaonlyt.edu.vn/index.php?id=320'](http://www.daotaonlyt.edu.vn/index.php?id=320)
[http://www.beemabuild.co.uk/view_product.php?id=217'](http://www.beemabuild.co.uk/view_product.php?id=217)

b) Kiểm tra chỗ yếu trang web

Bạn có thể điền thêm một số lệnh trên url, hoặc trên các form login, search, hoặc search để phát hiện lỗi. Một số thông báo lỗi của MySQL:

```
Mysql_fetch_array();  
Database query failed...  
You have an error in your SQL systax...  
....
```

c) Khai thác các thông tin

Đây là bước quan trọng nhất và đòi hỏi nhiều kỹ thuật lẫn sự am hiểu về cơ sở dữ liệu.

Trong Mysql có 2 thành phần quan trọng:

INFORMATION_SCHEMA.TABLES

INFORMATION_SCHEMA.COLUMNS

Dựa vào lệnh UNION SELECT, có thể khai thác được:

- Phiên bản của CSDL
- Tên Database

- Xác định được tên các Tables quan trọng
- Xác định được các Columns của Tables

d) Xử lý kết quả tìm được

Khi bạn đã có tên của tất cả các column trong table, bạn có thể UPDATE hoặc INSERT một record mới vào table này. Hoặc bạn login trực tiếp vào và thực hiện dưới quyền user đó.

4. Phòng tránh

- Loại bỏ các ký tự và từ khóa nguy hiểm như: --, select, where, drop, shutdown...
- Để phòng tránh các nguy cơ có thể xảy ra, hãy bảo vệ các câu lệnh SQL là bằng cách kiểm soát chặt chẽ tất cả các dữ liệu nhập nhận được từ đối tượng Request
- Cần có cơ chế kiểm soát chặt chẽ và giới hạn quyền xử lý dữ liệu đến tài khoản người dùng mà ứng dụng web đang sử dụng. Các ứng dụng thông thường nên tránh dùng đến các quyền như dbo hay sa. Quyền càng bị hạn chế, thiệt hại càng ít
- Nên chú ý loại bỏ bất kỳ thông tin kỹ thuật nào chứa trong thông điệp chuyển xuống cho người dùng khi ứng dụng có lỗi. Các thông báo lỗi thông thường tiết lộ các chi tiết kỹ thuật có thể cho phép kẻ tấn công biết được điểm yếu của hệ thống.

Đối với website (dành cho lập trình viên):

- Cần kiểm tra tính đúng đắn của tất cả dữ liệu đầu vào. Dữ liệu đầu vào không chỉ là các tham số, mà bao gồm cả cookie, user agent, referer ...
- Việc kiểm tra tính đúng đắn của dữ liệu có thể dựa trên các phương pháp sau:

Kiểm tra dựa vào kiểu dữ liệu (số, ngày tháng ...)

Kiểm tra, giới hạn độ dài đầu vào

Loại bỏ các ký tự đặc biệt như: ‘ % ’ ? # @ & ...

Loại bỏ các từ đặc biệt: select, delete, information_schemal, insert,...

Đối với web server (dành cho quản trị mạng):

Hầu hết các máy chủ web (web server) hiện nay đều có các module hỗ trợ việc phòng chống SQL Injection. Ví dụ: Apache có modsecurity, IIS có URLScan,... Bạn chỉ cần bật tính năng này và cấu hình cho phù hợp. Nếu website của bạn là dạng trang tin tức thì rất phù hợp để triển khai. Trong một số trường hợp khác, các module này có thể chặn nhầm, dẫn tới website hoạt động không chính xác.

Đối với database server (dành cho quản trị mạng):

Bạn cần thực hiện việc cấu hình phân quyền chặt chẽ đối với các tài khoản. Khi đó, dù tồn tại lỗi SQL Injection, thiệt hại cũng sẽ được hạn chế. Ngoài ra, bạn cần loại bỏ các bảng, thành phần và tài khoản không cần thiết trong hệ thống.

Phòng chống từ bên ngoài:

Giải pháp này sẽ dùng tường lửa đặc biệt để bảo vệ bạn khỏi những ứng dụng dùng việc truy cập database với mục đích xấu. Chúng ta cần lưu ý rằng attacker tương tác với ứng dụng web thông qua một trình duyệt với kết nối từ xa. Sau đó, ứng dụng gửi yêu cầu đến database. Như vậy chúng ta có thể ngăn chặn các tấn công giữa attacker với ứng dụng, giữa ứng dụng với database và ngay cả trên chính bản thân database đó.

Một số công cụ quét và kiểm tra lỗi SQL Injection hiệu quả:

Acunetix Web Vulnerability Scanner

N-Stealth

SQLmap

....

II. Buffer Overflow

1. Tổng quan về Buffer Overflow

- Định nghĩa: Lỗi tràn bộ đệm là một điều kiện bất thường khi một tiến trình lưu dữ liệu vượt ra ngoài biên của bộ nhớ đệm có chiều dài cố định. Kết quả là dữ liệu có thể đè lên các bộ nhớ liền kề. Dữ liệu bị ghi đè có thể bao gồm các bộ nhớ đệm khác, các biến và dữ liệu điều khiển luồng chảy của chương trình (program flow control).
- Các lỗi tràn bộ đệm có thể làm cho tiến trình bị đổ vỡ hoặc cho ra kết quả sai. Các lỗi này có thể được kích hoạt bởi các dữ liệu vào được thiết kế đặc biệt

để thực thi các đoạn mã phá hoại hoặc để làm cho chương trình hoạt động không như mong đợi. Bằng cách đó các lỗi tràn bộ đệm gây ra nhiều lỗ hổng bảo mật đối với phần mềm và tạo cơ sở cho nhiều thủ thuật khai thác.

- Trong các ngôn ngữ lập trình thì ngôn ngữ C dễ sinh ra các lỗi tràn nhớ mà attacker có thể khai thác .
- Trong ngôn ngữ C, các chuỗi (string) hay các buffer được thể hiện như sau: Con trỏ (pointer) sẽ chỉ trỏ vào byte đầu tiên của chuỗi hay buffer đó, và chúng ta xác định được kết điểm kết thúc khi con trỏ trỏ đến 1 byte null => không xác định độ dài của đối tượng nhập vào => có thể copy 1 buffer có độ dài lớn vào 1 buffer có độ dài nhỏ hơn => gây tràn nhớ(buffer overflow)
- Thông thường có 2 cách khai thác lỗi buffer overflow mà attacker có thể sử dụng:
 - Khai thác dựa vào các lỗ hổng phần mềm thông qua ngôn ngữ lập trình(phần mềm viết bằng ngôn ngữ C)
 - Khai thác các trang web có tương tác người dùng nhưng không ràng buộc dữ liệu nhập vào như các trường username ,password..
- Nguyên nhân gây ra các lỗi buffer overflow của các chương trình/ứng dụng:
 - Phương thức kiểm tra biên (boundary) không được thực hiện đầy đủ, hoặc là được bỏ qua.
 - Các ngôn ngữ lập trình, như là ngôn ngữ C, bản thân nó đã tiềm ẩn các lỗi mà hacker có thể khai thác.
 - Các phương thức strcat(), strcpy(), printf(), bcopy(), gets() ,và scanf() trong ngôn ngữ C có thể được khai thác vì các hàm này không kiểm tra xem những buffer được cấp phát trên stack có kích thước lớn hơn dữ liệu được copy vào buffer hay không?
- Một lỗi buffer overflow xuất hiện khi buffer cố gắng cấp phát 1 không gian lưu trữ có dữ liệu lớn hơn khả năng lưu trữ của nó

VD:

```
#include<stdio.h>
int main ( int argc , char **argv)
{
    char target[5]="TTTT";
    char attacker[11]="AAAAAAAAAA";
    strcpy( attacker," DDDDDDDDDDDDDDD");
    printf("%s \n",target);
    return 0;
}
```

Có hai kiểu buffer overflow chính:

- Stack overflow
- Heap over flow

Stack và heap là địa điểm lưu trữ cung cấp cho các biến khi chạy. một chương trình. Biến được lưu trữ trong stack hoặc heap cho đến khi chương trình có nhu cầu sử dụng.

Stack: Là nơi lưu trữ tĩnh của không gian bộ nhớ.

Heap: Là nơi lưu trữ động của không gian bộ nhớ, được sinh ra khi chạy một chương trình.

Thông thường có 2 cách khai thác lỗi buffer overflow mà attacker có thể sử dụng:

- Khai thác dựa vào các lỗ hổng phần mềm thông qua ngôn ngữ lập trình (phần mềm viết bằng ngôn ngữ C)
- Khai thác các trang web có tương tác người dùng nhưng không ràng buộc dữ liệu nhập vào như các trường username, password...

2. Stack Overflow

a) Giới thiệu về stack:

- Stack là một kiểu cấu trúc dữ liệu hoạt động theo cơ chế LIFO (last in first out) được máy tính sử dụng để chuyển các đối số vào các hàm để tham chiếu đến các biến cục bộ.
- Stack sẽ lưu giữ tất cả các thông tin mà hàm cần.
- Stack được khởi tạo khi bắt đầu của một hàm và được “phóng thích” khi hàm kết thúc.

Stack overflow xuất hiện khi buffer tràn trong stack space. Đoạn code độc hại sẽ được push vào stack. Lỗi tràn nhớ có thể viết lại giá trị của return pointer, vì thế hacker có thể điều khiển con trỏ tới các đoạn code độc hại mà họ muốn thực thi.

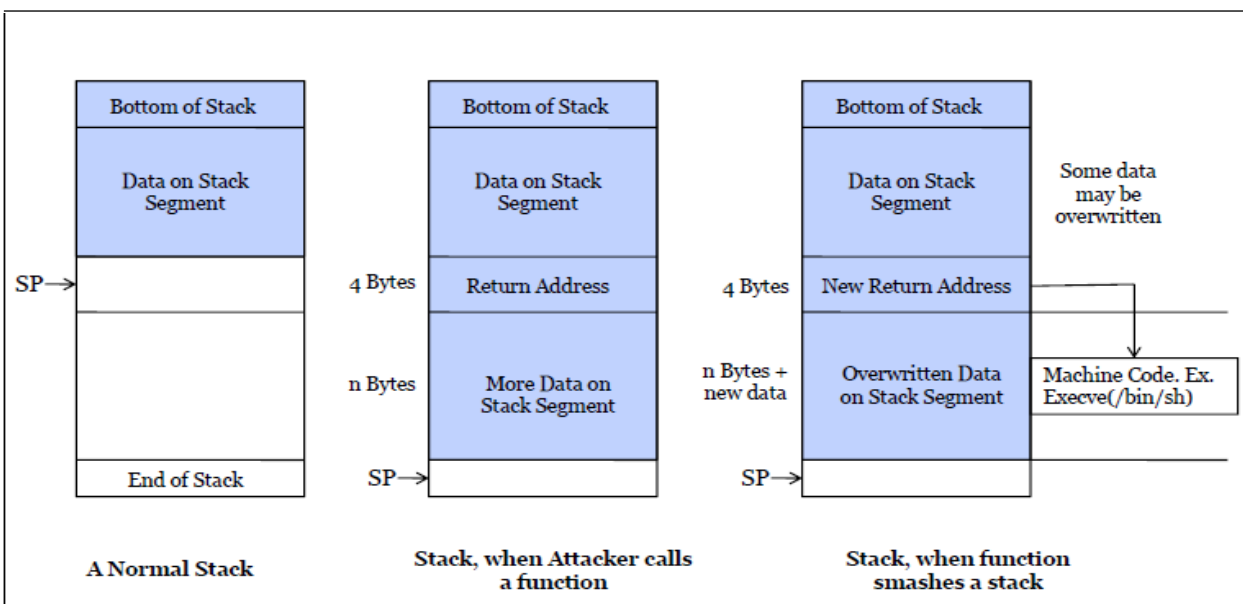
b) Mục đích chính của stack buffer overflow:

- Ghi đè một biến địa phương nằm gần bộ nhớ đệm trong stack để thay đổi hành vi của chương trình nhằm phục vụ cho ý đồ của hacker.

- Ghi đè địa chỉ trả về trong khung stack (stack frame). Khi hàm trả về thực thi sẽ được tiếp tục tại địa chỉ mà hacker đã chỉ rõ, thường là tại một bộ đệm chứa dữ liệu vào của người dùng.

c) Ví dụ cơ bản về tràn nhớ trong stack

```
/* This is a program to show a simple uncontrolled overflow of the stack. It will
   overflow EIP with 0x41414141, which is AAAA in ASCII. */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int bof()
{
    char buffer[8]; /* an 8 byte character buffer */
    strcpy(buffer, "AAAAAAAAAAAAAAAAAAAA"); /*copy 20 bytes of A into the buffer*/
    return 1; /*return, this will cause an access violation due to stack corruption.*/
}
int main(int argc, char **argv)
{
    bof(); /*call our function*/
    /*print a short message, execution will never reach this point because of the
       overflow*/
    printf("Lets Go\n");
    return 1; /*leaves the main function*/
}
```



3. Heap Overflow

- Các biến được cấp phát tự động trong hàm, như là malloc() được tạo ra trong heap.
- Trong cách tấn công heap overflow, attacker làm tràn bộ nhớ sẽ có thể overwrite các dynamic variables, vì thế có thể dẫn đến các hiệu ứng không mong muốn.
- Trong hầu hết các môi trường ứng dụng, việc này có thể giúp attacker có thể điều khiển các thực thi của chương trình.

Ví dụ cơ bản về Heap Overflow:

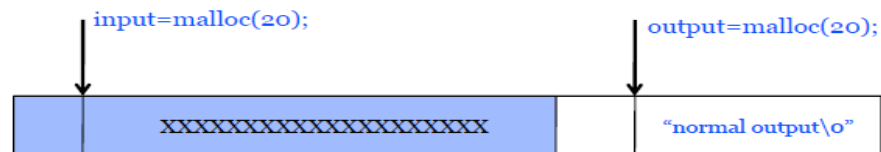
```
/*heap1.c - the simplest of heap overflows*/
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    char *input = malloc (20);
    char *output = malloc (20);

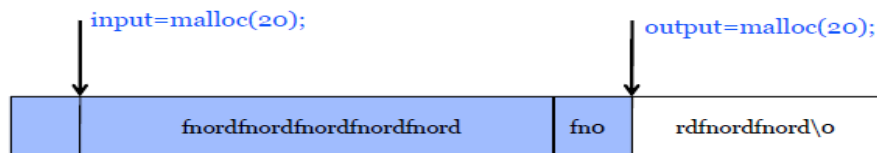
    strcpy (output, "normal output");
    strcpy (input, argv[1]);

    printf ("input at %p: %s\n", input, input);
    printf ("output at %p: %s\n", output, output);

    printf("\n\n%s\n", output);
}
```



Heap: Before Overflow



Heap: After Overflow

Các dạng tấn công dựa vào heap:

- Overwriting pointers (viết lại con trỏ): attacker có thể sử dụng phương pháp này để viết lại filename, password, uid..

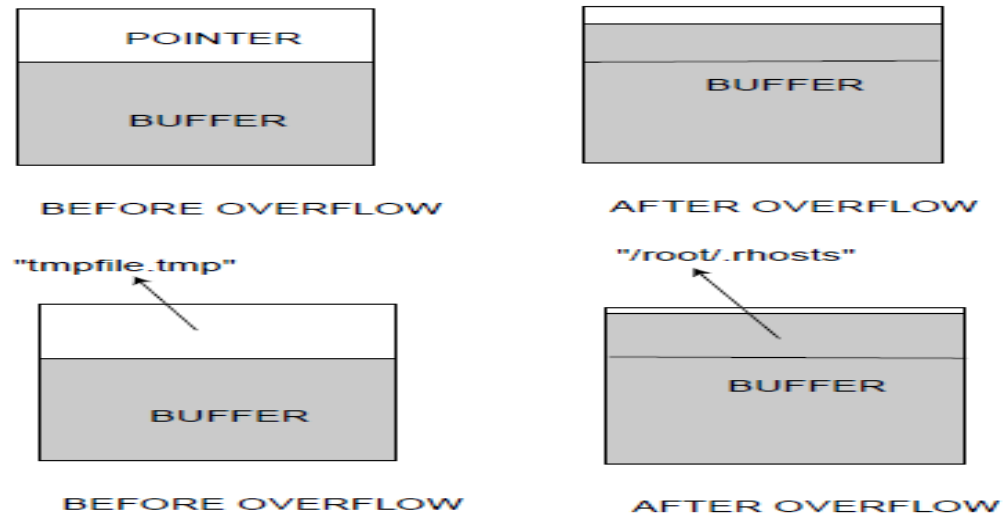
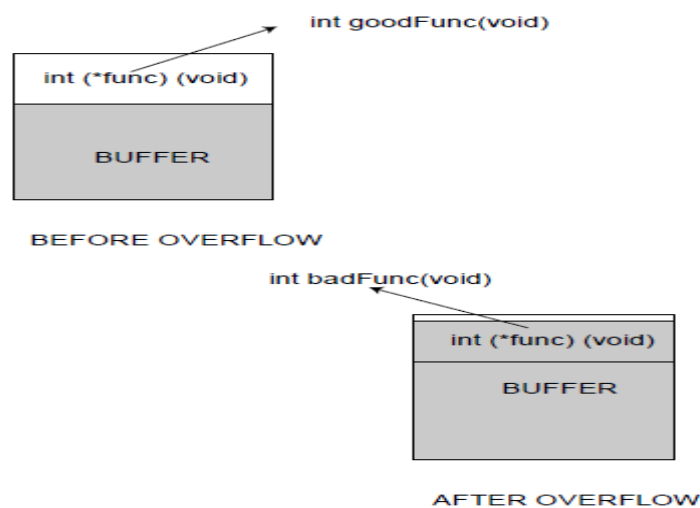
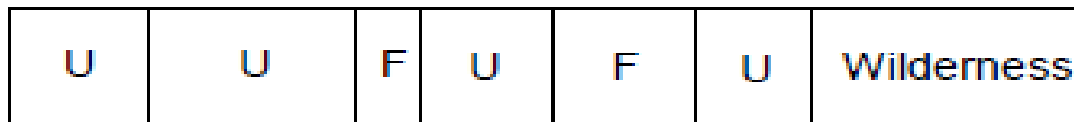


Figure 3.1: Overwriting a pointer in the heap

- Cuộc tấn công kiểu này cần một số điều kiện tiên quyết trong mã nguồn của vulnerable binary: một định nghĩa buffer và pointer.
- Khó khăn của phương pháp này là tìm ra hai điều kiện tiên quyết như trên. Một khó khăn khác là xác định địa chỉ của vulnerable program.
- Overwriting function pointers: Attacker muốn ghi đè lên một con trỏ và làm cho nó trỏ đến những gì họ muốn. nó có thể trỏ đến 1 chương trình nào đó.



- Exploiting the malloc library (Khai thác thư viện malloc): Dmalloc được gọi là thư viện Doug Lea malloc, từ tên của tác giả của nó, và cũng là malloc thư viện được sử dụng bởi LIBC gnu (DLMALLOC: cấu trúc)



U : Used chunk

F : Free chunk

Wilderness : top most free chunk

Figure 3.6: Memory layout

- Phần bên phải là một phần của heap mà có thể được tăng lên trong quá trình thực hiện (với sbrk hệ thống gọi dưới Unix, Linux).
- Mỗi đoạn bộ nhớ luôn lớn hơn kích thước yêu cầu của người dùng, vì nó cũng giữ quản lý thông tin. Về cơ bản nó chứa kích thước của khối và trỏ đến các khối tiếp theo và trước đó. Định nghĩa cấu trúc của 1 đoạn (trunk) là:

```
struct malloc_chunk {
size_t prev_size; // only used when previous chunk is free
size_t size;      // size of chunk in bytes + 2 status-bits
struct malloc_chunk *fd; // only used for free chunks: pointer to next chunk
struct malloc_chunk *bk; // only used for free chunks:
pointer to previous chunk
};
```

Mục đích của việc làm hỏng cấu trúc của DLMALLOC:

Attacker làm tràn bộ nhớ, sau đó ghi đè dữ liệu lên các mục tiêu của họ. Yêu cầu đặc biệt cho một dmalloc khai thác là có hai khối bộ nhớ thu được bằng malloc.

Ví dụ:

```
int main(void) {  
    char * buf ;  
    char * buffer1 = (char *)malloc(666) ;  
    char * buffer2 = (char *)malloc(2);  
    printf(Enter something: \n);  
    gets(buf);  
    strcpy (buffer1, buf);  
    free(buffer1);  
    free(buffer2);  
    return (1);  
}
```

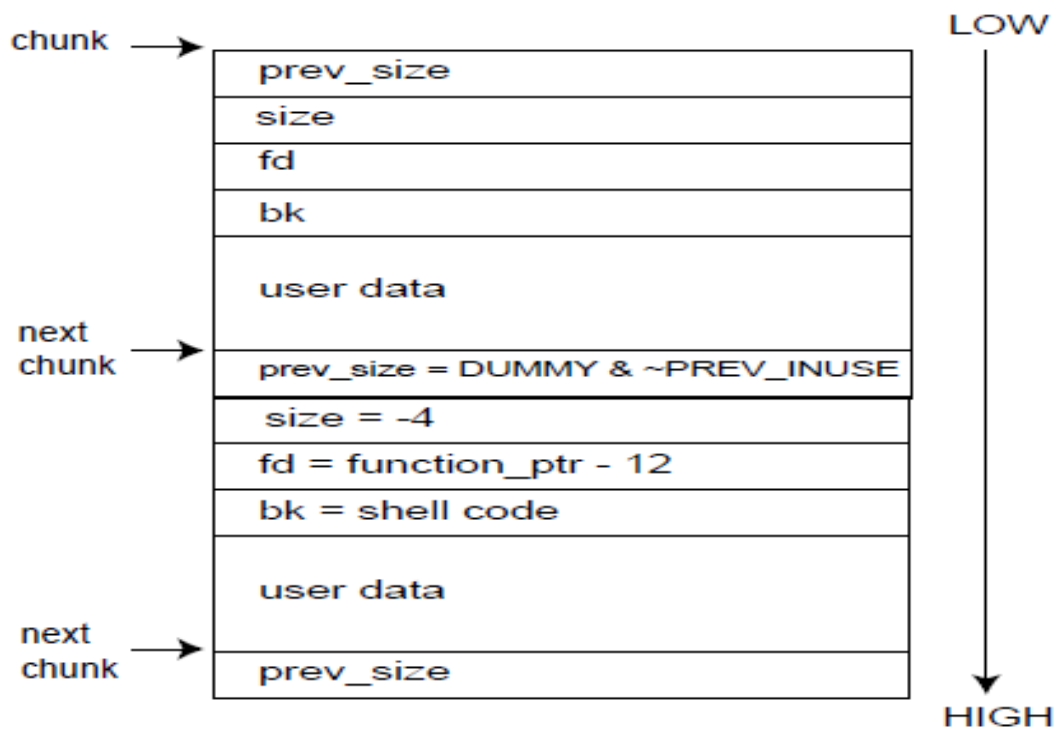


Figure 3.8: Our fake chunk

4. Các cách phát hiện Buffer overflow

Có 2 cách cơ bản để phát hiện các lỗi buffer overflow mà hacker có thể sử dụng:

- Nhìn vào source code: Hacker tìm kiếm các string là các biến cục bộ trong các hàm của chương trình, xem xét chúng có được kiểm tra biên (boundary check) chưa? Kiểm tra tiêu chuẩn của các hàm có liên quan đến phần các chuỗi trong phần input, output
- Cách thứ 2 mà hacker có thể dùng là nhập vào các ứng dụng một dữ liệu rất lớn và xem xét ứng dụng có bất cứ biểu hiện không bình thường nào không?

5. Cách phòng tránh Buffer overflow

Cản trở đối với các kỹ thuật khai thác lỗi buffer overflow: Việc xử lý bộ đệm trước khi đọc hay thực thi nó có thể làm thất bại các cố gắng khai thác lỗi tràn bộ đệm nhưng vẫn không ngăn chặn được một cách tuyệt đối. Việc xử lý bao gồm:

- Chuyển từ chữ hoa thành chữ thường
- Loại bỏ các ký tự đặc biệt và lọc các xâu không chứa ký tự là chữ số hoặc chữ cái.

Tuy nhiên vẫn có các kỹ thuật để tránh việc lọc và xử lý này:

- alphanumeric code: mã gồm toàn chữ và số
- polymorphic code : mã đa hình
- Self-modifying code : mã tự sửa đổi
- Tấn công kiểu return-to-libc

Vì thế để tránh các nguy cơ bị khai thác lỗi buffer overflow chúng ta cần sử dụng các biện pháp phòng tránh hiệu quả hơn.

a) Lựa chọn ngôn ngữ lập trình

Ngôn ngữ lập trình có một ảnh hưởng lớn đối với sự xuất hiện lỗi tràn bộ đệm:

- Ngôn ngữ lập trình C và C++ là hai ngôn ngữ lập trình thông dụng, nhưng hạn chế của nó là không kiểm tra việc truy cập hoặc ghi đè dữ liệu thông qua các con trỏ. Cụ thể nó không kiểm tra dữ liệu copy vào một mảng có phù hợp kích thước của mảng hay không?
- Cyclone, một biến thể của C, giúp ngăn chặn các lỗi tràn bộ đệm bằng việc gắn thông tin về kích thước mảng với các mảng.

- Ngôn ngữ lập trình D sử dụng nhiều kỹ thuật đa dạng để tránh gần hết việc sử dụng con trỏ và kiểm tra biên do người dùng xác định.
- Nhiều ngôn ngữ lập trình khác cung cấp việc kiểm tra tại thời gian chạy. . Việc kiểm tra này cung cấp một ngoại lệ hay 1 cảnh báo khi C hay C++ ghi đè dữ liệu ví dụ như: Python, Ada, Lisp, Smalltalk, Ocaml.
- Ngoài ra các môi trường của Java hay .NET cũng đòi hỏi kiểm tra biên đối với tất cả các mảng.

b) Sử dụng các thư viện an toàn

- Sử dụng các thư viện được viết tốt và đã được kiểm thử dành cho các kiểu dữ liệu trừu tượng mà các thư viện này thực hiện tự động việc quản lý bộ nhớ, trong đó có kiểm tra biên có thể làm giảm sự xuất hiện và ảnh hưởng của các hiện tượng tràn bộ đệm.
- Các thư viện an toàn gồm có The Better String Library, Arri Buffer API và Vstr.

c) Chống tràn bộ đệm trên stack

- Stack-smashing protection là kỹ thuật dùng để phát hiện các hiện tượng tràn bộ đệm phổ biến nhất. Kỹ thuật này kiểm tra xem stack đã bị sửa đổi hay chưa khi một hàm trả về. Nếu stack đã bị sửa đổi, chương trình kết thúc bằng một lỗi segmentation fault.
- Chế độ Data Execution Prevention (cấm thực thi dữ liệu) của Microsoft bảo vệ các con trỏ và không cho chúng bị ghi đè.
- Có thể bảo vệ stack bằng cách phân tán stack thành 2 phần, một phần dành cho dữ liệu và một phần dành cho các bước trả về của hàm. Sự phân chia này được dùng trong ngôn ngữ Forth.

d) Bảo vệ không gian thực thi

- Kỹ thuật này ngăn chặn việc thực thi mã tại stack hay heap. Hacker có thể sử dụng tràn bộ đệm để chen một đoạn mã tùy ý vào bộ nhớ của chương trình, với việc bảo vệ không gian thực thi mọi cố gắng chạy đoạn mã đó sẽ gây ra một ngoại lệ.
- Một số CPU hỗ trợ một tính năng có tên bit NX (No eXecute) hoặc bit XD (eXecute Disable). Khi kết hợp với phần mềm các tính năng này có thể được

dùng để đánh dấu các trang dữ liệu (chẳng hạn như các trang chứa stack và heap) là đọc được chứ không thực thi được.

- Các biến thể mới của Microsoft Windows cũng hỗ trợ bảo vệ không gian thực thi với tên gọi Data Execution Prevention và các phần mềm gắn kèm bao gồm: SecureStack, OverflowGuard, BufferShield.

e) Ngẫu nhiên hóa sơ đồ không gian địa chỉ

Ngẫu nhiên hóa sơ đồ không gian địa chỉ (Address space layout randomization ASLR) là một tính năng an ninh máy tính có liên quan tới việc sắp xếp các vùng dữ liệu quan trọng (thường bao gồm nơi chứa mã thực thi và vị trí các thư viện, heap và stack) một cách ngẫu nhiên trong không gian địa chỉ của một tiến trình.

f) Kiểm tra sâu đối với gói tin

- Biện pháp kiểm tra sâu đối với gói tin (deep packet inspection-DPI) có thể phát hiện việc cố gắng khai thác lỗi tràn bộ đệm từ xa ngay từ biên giới mạng. Các kỹ thuật này có khả năng ngăn chặn các gói tin có chứa chữ ký của một vụ tấn công đã biết hoặc chứa các chuỗi dài các lệnh No-Operation (NOP-lệnh rỗng không làm gì).
- Việc rà gói tin không phải là một phương pháp hiệu quả vì nó chỉ có thể ngăn chặn các cuộc tấn công đã biết và có nhiều cách để mã hóa một lệnh NOP.

III. Malware

1. Malware là gì?

Malware xuất phát từ cụm từ Malicious Software có nghĩa là phần mềm độc hại. Malware là cụm từ dùng để chỉ chung tất cả những phần mềm, công cụ, đoạn mã,... Có khả năng làm hại máy tính hay gây ảnh hưởng trái phép tới người dùng. Chính vì thế định nghĩa malware đã bao gồm cả: Virus, Worm, Trojan, Spyware, Adware... **Vậy Virus, Worm, Trojan, Spyware,... là gì?**

Có khá nhiều định nghĩa về những thứ này. Nhưng ta nên hiểu đơn giản:

- Virus: là một số dạng mã độc cần đến sự tác động của con người để lây lan. Ví dụ click tập tin mã độc, click tập tin có đính kèm mã độc, truy cập website có mã độc,...
- Worm: là một số dạng mã độc có thể tự lây lan, và chính vì thế nó thường lây lan rất kinh khủng. Ví dụ: Tự động lây lan qua những máy tính có OS chứa lỗ hổng trong cùng lớp mạng (ví dụ con conficker,...) ,...
- Trojan: Trước đây thì Trojan chỉ đơn giản là những dạng mã độc không thể tự lây lan và chức năng chủ yếu của nó đúng với tên gọi Trojan đó là chứa hoặc "Tuồn" những loại mã độc khác vào hệ thống. Tuy nhiên ngay nay trojan đã có nhiều biến tướng.
- Spyware: Phần mềm gián điệp, thu thập những thông tin trái phép trong máy tính
- Adware: Phần mềm quảng cáo, đây thường là nguyên nhân làm nên những hiện tượng như: Popup lạ, thông báo lạ thường xuyên nhảy ra trên máy tính, trang chủ trình duyệt web bị thay đổi, trang chủ tìm kiếm bị thay đổi,... Đôi khi Adware cũng có thể bí mật ghi lại một số thông tin như lịch sử truy cập web, số điện thoại (đối với adware trên smartphone), địa chỉ Mail,... Và gửi về cho chủ nhân, điều này làm nó gần chạm tới người anh em Spyware của nó.

Ngoài ra còn rất nhiều khái niệm như backdoor, Rootkit, w32,..... Tất cả được phân loại dựa trên khả năng và đặc tính của nó.

2. Các kỹ thuật phân tích mã độc?

Có 2 kỹ thuật phân tích mã độc chính:

- Dynamic analysis (Phân tích động): Hiểu nôm na quá trình này có nghĩa là quan sát hành vi của mã độc khi nó đang chạy. Cụ thể là những công việc như: Khởi chạy mã độc trên một môi trường ảo, quan sát xem khi mà mã độc chạy

nó sẽ làm những gì,... Debug mã độc sử dụng một công cụ Debugger nào đó Ví dụ: winDBG, OllyDBG để quan sát từng bước hành vi của mã độc khi nó đang được thực thi bởi bộ nhớ và load trong RAM.

- Static analysis (Phân tích tĩnh): Là quá trình dịch ngược mã độc (RCE - Đọc thêm về kỹ thuật reverse engineering). Dịch ngược mã độc từ mã máy ra ngôn ngữ mà con người có thể đọc hiểu (asm, IL,...). Trong quá trình này các nhà phân tích sẽ sử dụng những công cụ dịch ngược như IDA, khi thực hiện dịch ngược IDA không load mã độc vào RAM như ollyDBG.

Người phân tích có thể thực hiện chỉ 1 trong 2 quá trình trên hoặc là cả 2 tùy theo từng trường hợp. Và mục tiêu cuối cùng sau các quá trình là họ phải đáp ứng được 2 yêu cầu:

- Yêu cầu hàng đầu đó là xác định được dấu hiệu nhận diện của mã độc, khi xác định những dấu hiệu nhận diện mã độc thì có thể bàn đến chuyện loại bỏ mã độc. Những dấu hiệu nhận diện này sẽ được bổ sung vào Cơ sở dữ liệu của các anti virus.
- Yêu cầu thứ 2 là hiểu mã độc sẽ làm những gì khi lây nhiễm vào hệ thống. Có một số trường hợp qua quá trình phân tích họ đã rút ra ngay dấu hiệu nhận diện mã độc và trang bị anti của mình khả năng nhận diện và tiêu diệt mã độc đó mà không cần thiết phải hiểu ngọn ngành từng dòng code, từng chức năng của mã độc.

3. Môi trường phân tích mã độc?

Để phân tích mã độc người phân tích cần tạo ra những môi trường ảo để phân tích, đảm bảo mã độc không thể lây nhiễm ra ngoài hay làm hại đến máy tính của chính mình. Giả sử bạn đang ngồi phân tích một mã độc có khả năng lây lan mạnh nhất trong lịch sử. Và bạn bắt đầu phân tích nó bằng cách chạy nó ngay trên máy tính có kết nối ra internet của mình, Việc đó sẽ khiến mã độc ngay lập tức có thể lây lan ra ngoài và gây ra hậu quả không lường trước được. Hơn nữa có một quy tắc khi phân tích mã độc đó là tuyệt đối không nên để cho chủ mưu nhận ra có người đang cố gắng phân tích mã độc của hắn. Chẳng hạn 1 attack điều hành nhiều mạng lưới Botnet lớn, ta bắt được 1 mẫu trong đồng Zombies của hắn. Trong quá trình phân

tích và cố gắng tìm cách tóm gọn toàn bộ mạng lưới botnet đó, nếu vô tình attack nhận ra có người đang cố gắng phân tích mã độc của hắn, hắn sẽ nhanh chóng xóa hết dấu vết và ta sẽ chẳng bao giờ tóm được nó.

Môi trường ảo ở đây sẽ được xây dựng tùy trường hợp, tùy từng đơn vị phân tích. Đối với một người phân tích mã độc tự do không chuyên thì môi trường ảo đó chỉ đơn giản là:

- Một Vmware hay Virtualbox có cài sẵn windows xp,7,... cùng một số ứng dụng thông dụng như java, adobe flash, firefox,... và máy ảo này sẽ không có kết nối mạng ra ngoài và ra máy thật (host only) hoặc có kết nối ra máy thật, máy thật sử dụng Linux và Không có kết nối mạng (offline) (Trong trường hợp này thường người ta sẽ đăng tcpdump ngoài máy thật và wireshark ở trong hoặc cũng có thể thực hiện sinkhole)
- Chuẩn bị đầy đủ các công cụ phục vụ cho quá trình phân tích tĩnh và phân tích động.
- Chuẩn bị mẫu mã độc.

Đối với những chuyên gia phân tích chuyên nghiệp trong các phòng thí nghiệm mã độc họ có thể xây dựng những mô hình phức tạp hơn, Chẳng hạn xây dựng nhiều máy ảo thông với nhau để thử nghiệm lây lan trong Lan, hay Xây dựng nhiều máy ảo với nhiều nền tảng khác nhau, nhưng máy chạy win 7, máy chạy xp, máy chạy win 8.1, máy chạy Linux,... máy có .Net framework và c++ redistributable,... máy không hỗ trợ....

4. Các công cụ phục vụ cho quá trình phân tích tĩnh và động?

Công cụ môi trường:

- Compiler, IDE (Dev C++, visual C++, python, MASM, winasm, avtive perl,...)
- Thư viện hỗ trợ (.Net Framework, C++ redistributable, wincap,...)

Công cụ phục vụ phân tích động

- Công cụ theo dõi hành vi mã độc: cực kì nhiều, nhưng có thể kể đến 1 số như: Bộ SysinternalsSuite, Regshot, networkminer, wireshark, PC Hunter Một số công cụ check MD5, SHA-1...
- Công cụ Debugger OllyDBG, winDBG, Immunity Debugger

Công cụ phục vụ phân tích tĩnh:

- IDA
- Ngoài ra có nhiều trình Dịch ngược khác tùy từng loại mã độc, ví dụ mã độc code = autoit thì dùng Exe2Aut, code = Delphi thì dùng DeDe,....

Các công cụ dịch ngược / Debug trên các nền tảng khác nhau: x86, x64, Dịch ngược các tập tin thực thi khác nhau như python, ELF, Exe,.... dịch ngược các thư viện (Dll, so,...),...

Các công cụ khác

- PE file format: ExePEinfo, PEiD,...
- Các công cụ unpack auto: UPX,...
- CFF Explorer, Hex workshop / FileInsight, OfficeMalScanner, Offvis, Pestudio, Notepad ++ , Peepdf,...
- Các công cụ Dump và phân tích memory: volatility, Redline, DumpIt,...

5. Phòng tránh mã độc

- Không click tùy tiện
- Cập nhật các bản vá lỗi ứng dụng / Hệ điều hành quan trọng (Không nên cập nhật tất cả)
- Sử dụng 1 Antivirus cập nhật liên tục và 1 Firewall, sử dụng USB Secure để quét máy tính định kì.
- Theo dõi Hệ thống thường xuyên và liên tục nếu có kinh nghiệm