

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN I



BÀI BÁO CÁO THỰC HÀNH SỐ 3

Môn học: PHÂN TÍCH MÃ ĐỘC

Giới thiệu công cụ Ghidra

Tên sinh viên: Ninh Chí Hường

Mã sinh viên: B20DCAT094

Nhóm lớp : 02

Giảng viên hướng dẫn: PGS.TS Đỗ Xuân Chợt

HÀ NỘI, THÁNG 10/2023

I. Giới thiệu:

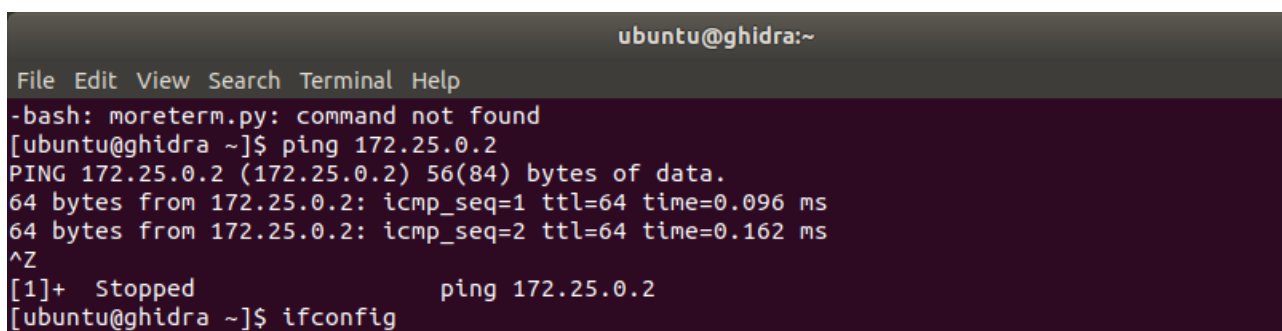
- Ghidra là một công cụ kỹ thuật đảo ngược (reverse engineering) hoạt động dựa trên Java có giao diện đồ họa người dùng (GUI)
- Ghidra giúp phân tích các loại mã độc, phần mềm chứa virus, được thiết kế để chạy trên nhiều nền tảng như Windows, macOS và Linux, có những chức năng tương tự như IDA-Pro nhưng nó là một công cụ mã nguồn mở hoàn toàn miễn phí.
- Nó hỗ trợ chuyển thành code Assembly và code C, cho ta biết tập tin đã import những dll nào
- Ngoài ra còn có những chức năng khác để hỗ trợ việc phân tích như : Function call graph tree, Strings, Memory map, Viết script python

II. Khái quát chung:

- Bài thực hành này giới thiệu công cụ Ghidra của ghidra-sre.org. Sinh viên sẽ sử dụng Ghidra để phân tích một tập tin thực thi nhằm xác định một số thuộc tính của nó.
- Yêu cầu đối với sinh viên: Sinh viên sử dụng được ngôn ngữ lập trình C, Assembly cơ bản, có kiến thức về mạng máy tính.

III. Cấu hình mạng:

- Bài thực hành bao gồm một máy client và 1 máy server. Sinh viên sử dụng máy client để truy cập vào server, trên máy client, tên là ghidra chứa bộ công cụ Ghidra và bản sao của dịch vụ đang chạy trên máy chủ. Bạn có hai cửa sổ dòng lệnh trên máy tính này. Sử dụng lệnh `moreterm.py` để mở thêm.
- Địa chỉ IP của máy chủ là 172.25.0.2. Sử dụng lệnh ping để xác nhận kết nối:
ping 172.25.0.2



```
ubuntu@ghidra:~  
File Edit View Search Terminal Help  
-bash: moreterm.py: command not found  
[ubuntu@ghidra ~]$ ping 172.25.0.2  
PING 172.25.0.2 (172.25.0.2) 56(84) bytes of data.  
64 bytes from 172.25.0.2: icmp_seq=1 ttl=64 time=0.096 ms  
64 bytes from 172.25.0.2: icmp_seq=2 ttl=64 time=0.162 ms  
^Z  
[1]+  Stopped                  ping 172.25.0.2  
[ubuntu@ghidra ~]$ ifconfig
```

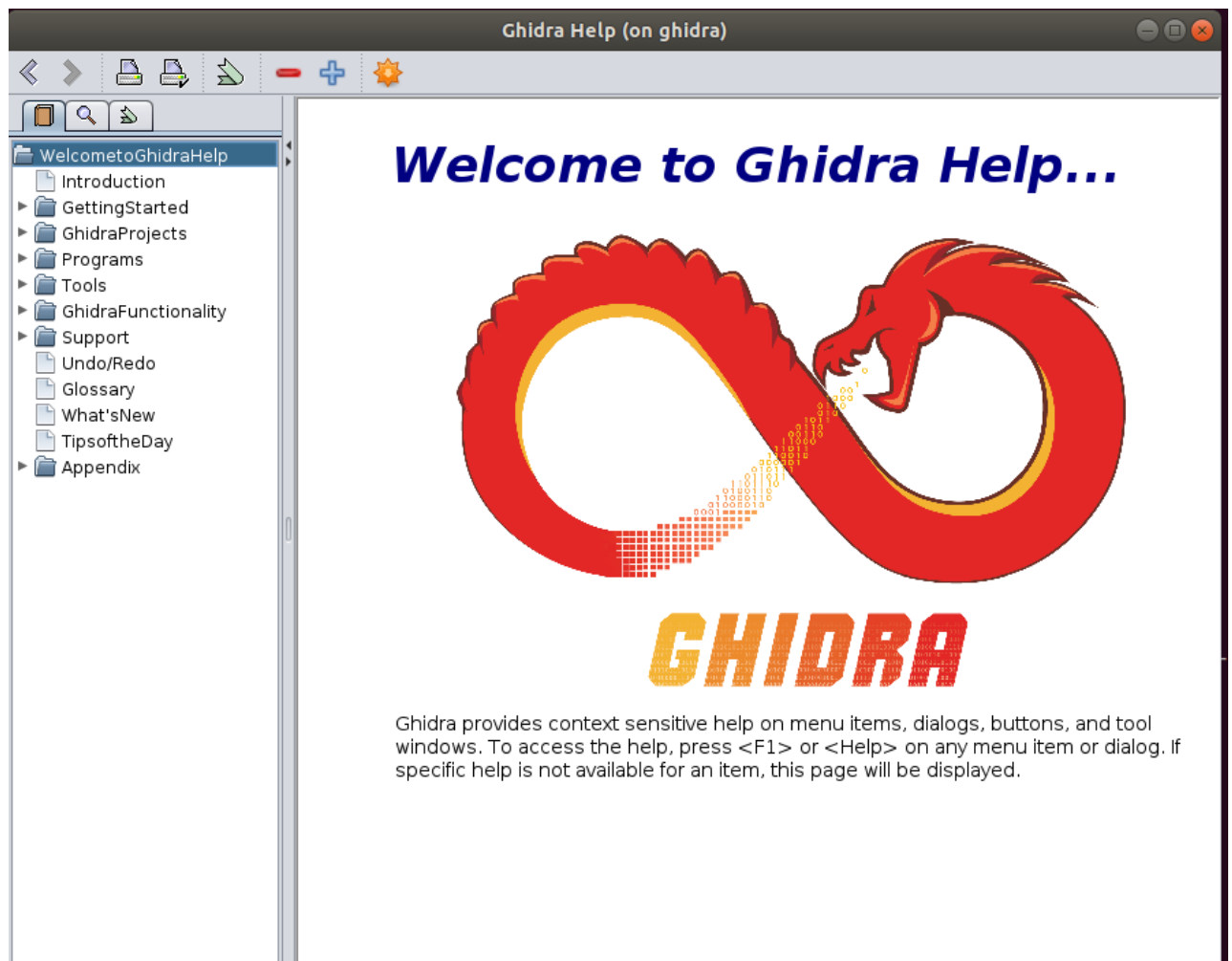
IV. Thực hiện nhiệm vụ:

1. Giới thiệu:

- Một bản sao của phần mềm dịch vụ có trong máy ghidra có tên là cadet011 trong thư mục Home. Chương trình này đang chạy trên máy chủ. Mục tiêu của bạn là kết nối đến dịch vụ; khiến nó hiển thị “easter egg”; và sau đó làm chương trình bị lỗi. Sinh viên sử dụng Ghidra để phân tích chương trình cadet01 nhằm đạt được các mục tiêu trên.

2. Sử dụng công cụ Ghidra:

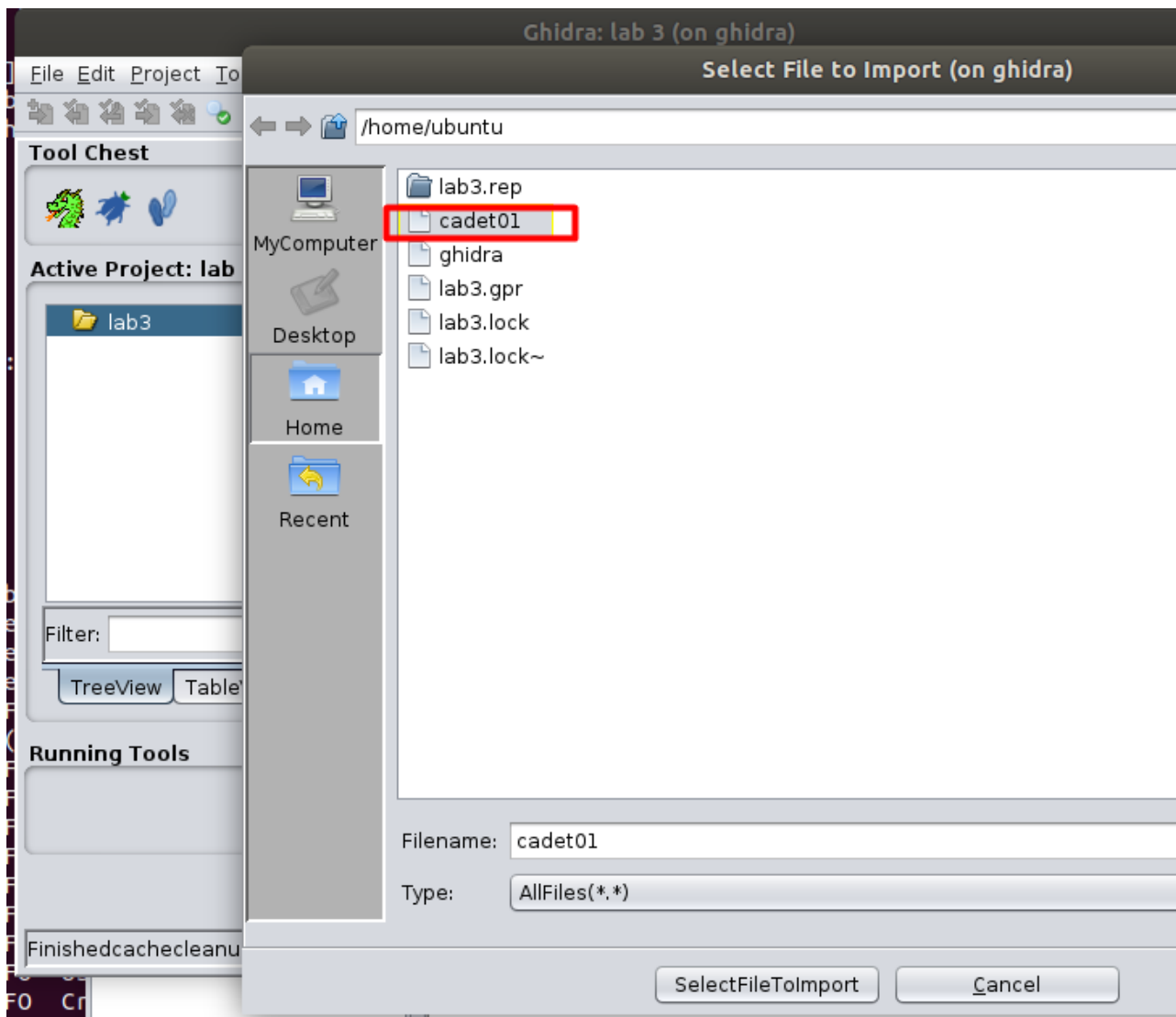
- Ghidra đã được cài sẵn trên máy. Chạy lệnh ./ghidra để khởi động chương trình. Sau khi chấp nhận điều khoản sử dụng, sinh viên sẽ thấy hai cửa sổ. Một trong các cửa sổ là trợ giúp trực tuyến. Hãy sử dụng trợ giúp này để làm quen với công cụ.



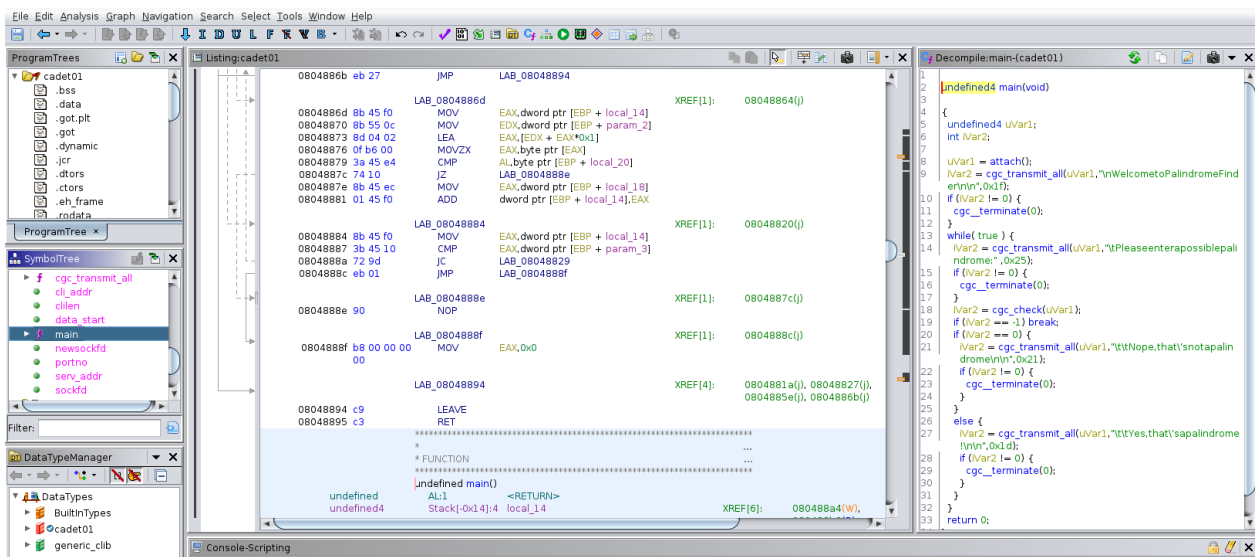
3. Tạo project và nhập chương trình cadet01:

- Sử dụng menu File/New project trên cửa sổ chính của Ghidra để tạo một dự án mới.
- Sau đó sử dụng menu File/Import file để nhập chương trình cadet01. Sau khi

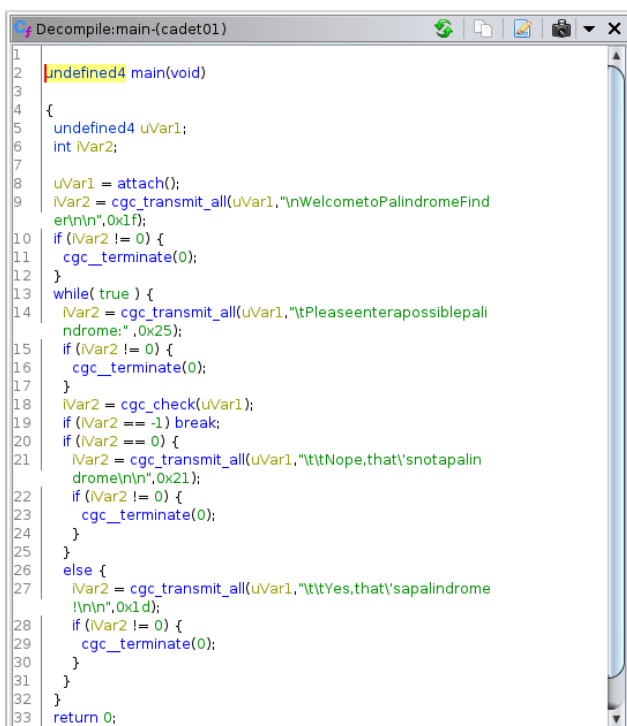
chương trình cadet01 xuất hiện trong cửa sổ Active Project, click đúp vào nó. Khi yêu cầu phân tích, hãy chọn Yes và chấp nhận các phân tích mặc định.



- Sinh viên sẽ thấy một cửa sổ mới có tiêu đề CodeBrowser... Trên thanh dọc bên trái giữa cửa sổ, có tiêu đề SymbolTree, mở thư mục có tên Functions. Trong thư mục này có các hàm mà Ghidra đã xác định trong tập tin thực thi. Tìm hàm chính và chọn nó. Hãy lưu ý rằng khung lớn ở giữa hiện chứa danh sách đã disassembled của hàm chính, và cửa sổ bên phải hiển thị danh sách pseudo-code được decompiled ra từ phân tích của Ghidra.



- Các hàm được gọi bởi hàm chính:



- attach()
- cgc_transmit_all()
- cgc_terminate()
- cgc_check()

- Các hàm được gọi bởi attach():

```
Decompile:attach-(cadet01)
1 int attach(void)
2 {
3     int iVar1;
4     sockfd = socket(2,1,0);
5     if (sockfd < 0) {
6         perror("ERROROpeningsocket" );
7         /*WARNING:Subroutinedoesnotreturn*/
8         exit(1);
9     }
10    bzero(serv_addr,0x10);
11    portno = 0x135b;
12    serv_addr_0_2_ = 2;
13    serv_addr_4_4_ = 0;
14    serv_addr_2_2_ = htons(0x135b);
15    iVar1 = bind(sockfd,(sockaddr *)serv_addr,0x10);
16    if (iVar1 < 0) {
17        perror("ERRORonbinding" );
18        /*WARNING:Subroutinedoesnotreturn*/
19        exit(1);
20    }
21    listen(sockfd,5);
22    cliilen = 0x10;
23    newsockfd = accept(sockfd,(sockaddr *)cli_addr,&cliilen);
24    if (newsockfd < 0) {
25        perror("ERRORonaccept" );
26        /*WARNING:Subroutinedoesnotreturn*/
27        exit(1);
28    }
29    return newsockfd;
30 }
31
32
33
34
```

- socket()
- bzero()
- listen()
- accept()
- sockaddr()

- Các hàm được gọi bởi cgc_transmit_all()

```
Decompile:cgc_transmit_all-(cadet01)
1 undefined4 cgc_transmit_all(undefined4 param_1,int param_2,u
2 int param_3)
3 {
4     undefined4 uVar1;
5     int local_18;
6     uint local_14;
7     int local_10;
8
9
10    local_14 = 0;
11    local_18 = 0;
12    if (param_2 == 0) {
13        uVar1 = 1;
14    }
15    else if (param_3 == 0) {
16        uVar1 = 2;
17    }
18    else {
19        for (; local_14 < param_3; local_14 = local_14 + local_18) {
20            local_10 = cgc_transmit(param_1,param_2 + local_14,param
21            _3 - local_14,&local_18);
22            if (local_18 == 0) {
23                return 3;
24            }
25            if (local_10 != 0) {
26                return 3;
27            }
28        }
29        uVar1 = 0;
30    }
31    return uVar1;
32 }
```

- cgc_transmit()

- Các hàm được gọi bởi cgc_terminate()

```
Decompile:cgc_terminate-(cadet01)
1
2 void cgc_terminate(void)
3
4 {
5     /*WARNING:Subroutine does not return*/
6     exit(1);
7 }
8
```

- Các hàm được gọi bởi cgc_check()

```
Decompile:cgc_check-(cadet01)
1
2 undefined4 cgc_check(undefined4 param_1)
3
4 {
5     int iVar1;
6     char local_5c [64];
7     int local_1c;
8     uint local_18;
9     undefined4 local_14;
10    int local_10;
11
12    local_1c = -1;
13    local_14 = 1;
14    for (local_18 = 0; local_18 < 0x40; local_18 = local_18 + 1) {
15        local_5c[local_18] = '\0';
16    }
17    iVar1 = cgc_receive_delim(param_1,local_5c,0x80,10);
18    if (iVar1 == 0) {
19        for (local_18 = 0; local_5c[local_18] != '\0'; local_18 = local_1
18 + 1) {
20            local_1c = local_1c + 1;
21        }
22        local_10 = local_1c;
23        if ((local_1c - (local_1c >> 0x1f) & 1U) + (local_1c >> 0x1f) =
23 = 1) {
24            local_10 = local_1c + -1;
25        }
26        for (local_18 = 0; (int)local_18 <= local_10 / 2; local_18 = loc
26 al_18 + 1) {
27            if (local_5c[local_18] != local_5c[(local_1c - local_18) + -1])
27            {
28                local_14 = 0;
29            }
30        }
31        if ((local_5c[0] == '[') &&
32            (iVar1 = cgc_transmit_all(param_1,"\\n\\nEASTEREGG!\\n\\n",0
32 xf), iVar1 != 0)) {
33            cgc_terminate(0);
34        }
35    }
36    else {
37        local_14 = 0xffffffff;
38    }
39    return local_14;
40 }
```

- cgc_receive_delim()

- cgc_transmit_all()

- cgc_terminate()

4. Tìm cổng mạng của dịch vụ:

- Xem qua các hàm của chương trình cadet01 để tìm số cổng mạng được sử dụng khi gắn kết với socket mạng

```
Decompile:attach-(cadet01)
1
2 int attach(void)
3
4 {
5     int iVar1;
6
7     sockfd = socket(2,1,0);
8     if (sockfd < 0) {
9         perror("ERRORonopeningsocket" );
10        /*WARNING:Subroutine does not return*/
11        exit(1);
12    }
13    bzero(serv_addr,0x10);
14    portno = 0x135b;
15    serv_addr_0_2_ = 2;
16    serv_addr_4_4_ = 0;
17    serv_addr_2_2_ = htons(0x135b);
18    iVar1 = bind(sockfd,(sockaddr *)serv_addr,0x10);
19    if (iVar1 < 0) {
20        perror("ERRORonbinding" );
21        /*WARNING:Subroutine does not return*/
22        exit(1);
23    }
24    listen(sockfd,5);
25    clien = 0x10;
26    newsockfd = accept(sockfd,(sockaddr *)cli_addr,&cli);
27    if (newsockfd < 0) {
28        perror("ERRORonaccept" );
29        /*WARNING:Subroutine does not return*/
30        exit(1);
31    }
32    return newsockfd;
33 }
34
```

```
serv_addr_2_2_ = htons(0x135b);
iVar1 = bind(sockfd,(sockaddr *)serv_addr,0x10);
if (iVar1 < 0) {
    perror("ERRORonbinding" );
    /*WARNING:Subroutine does not return*/
    exit(1);
}
listen(sockfd,5);
cli = 0x10;
newsockfd = accept(sockfd,(sockaddr *)cli_addr,&cli);
if (newsockfd < 0) {
    perror("ERRORonaccept" );
    /*WARNING:Subroutine does not return*/
    exit(1);
}
return newsockfd;
}
```

EditFunctionSignature	
OverrideSignature	
CommitParams/Return	P
CommitLocalNames	
Highlight	►
SecondaryHighlight	►
SetEquate...	E
Binary:0b0001001101011011	
Decimal:4955	
Octal:011533	
Char:'\x135b'	
Copy	Ctrl+C
Comments	►
Find...	Ctrl+F
References	►
Properties	

- Ở đây cổng dịch vụ mạng là 4955
- Sau khi đã tìm được số cổng, hãy sử dụng nó để kết nối đến server, sử dụng câu lệnh: **echo "Hi" | nc 172.25.0.2 4955**

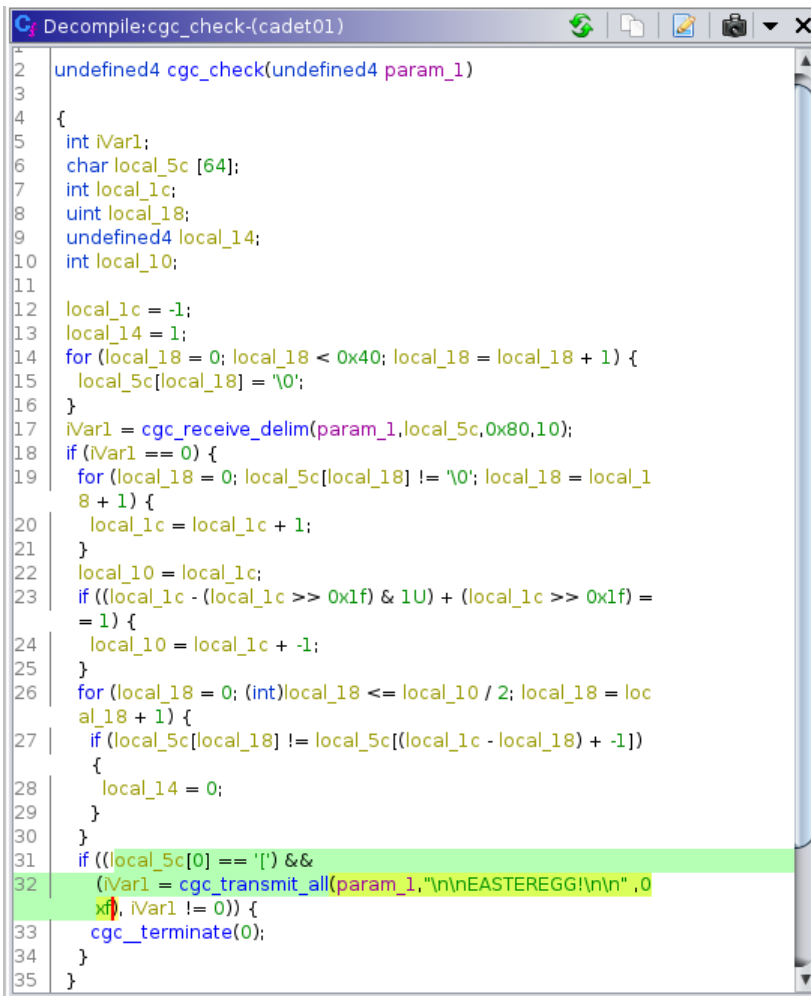
```
[ubuntu@ghidra ~]$ echo "Hi" | nc 172.25.0.2 4955
Welcome to Palindrome Finder

Please enter a possible palindrome: Nope, that's not a palindrome

Please enter a possible palindrome: [ubuntu@ghidra ~]$
```

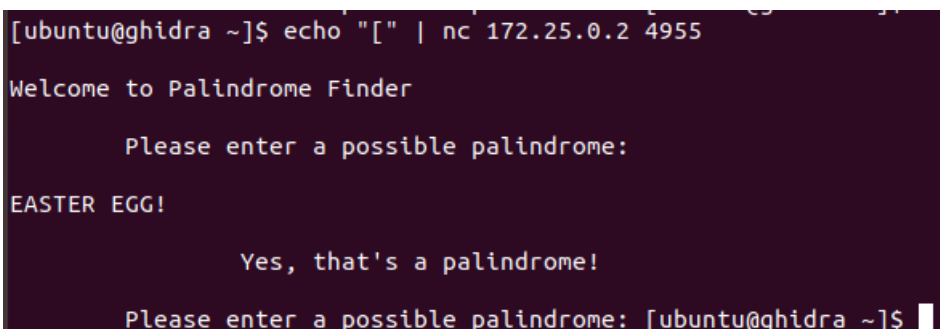

5. Tìm “Easter egg”:

- Một đầu vào cụ thể gửi đến dịch vụ sẽ khiến nó hiện “Easter egg”. Sử dụng Ghidra để xác định yêu cầu của đầu vào trong code. Sau đó dùng một chuỗi kí tự gửi đến server để hiện ra “Easter egg” này.



```
Decompile:cgccheck-(cadet01)
2  undefined4 cgccheck(undefined4 param_1)
3
4  {
5      int iVar1;
6      char local_5c [64];
7      int local_1c;
8      uint local_18;
9      undefined4 local_14;
10     int local_10;
11
12     local_1c = -1;
13     local_14 = 1;
14     for (local_18 = 0; local_18 < 0x40; local_18 = local_18 + 1) {
15         local_5c[local_18] = '\0';
16     }
17     iVar1 = cgcc_receive_delim(param_1,local_5c,0x80,10);
18     if (iVar1 == 0) {
19         for (local_18 = 0; local_5c[local_18] != '\0'; local_18 = local_1
20             8 + 1) {
21             local_1c = local_1c + 1;
22         }
23         local_10 = local_1c;
24         if ((local_1c - (local_1c >> 0x1f) & 1U) + (local_1c >> 0x1f) =
25             = 1) {
26             local_10 = local_1c + -1;
27         }
28         for (local_18 = 0; (int)local_18 <= local_10 / 2; local_18 = loc
29             al_18 + 1) {
30             if (local_5c[local_18] != local_5c[(local_1c - local_18) + -1])
31             {
32                 local_14 = 0;
33             }
34         }
35         if ((local_5c[0] == '[') &&
36             (iVar1 = cgcc_transmit_all(param_1,"\\n\\nEASTEREGG!\\n\\n",0
37                 xf), iVar1 != 0)) {
38             cgcc_terminate(0);
39         }
40     }
```

- Đoạn code này chỉ rằng nếu đầu vào là '[' thì sẽ print Easter egg



```
[ubuntu@ghidra ~]$ echo "[" | nc 172.25.0.2 4955
Welcome to Palindrome Finder

Please enter a possible palindrome:
EASTER EGG!

Yes, that's a palindrome!

Please enter a possible palindrome: [ubuntu@ghidra ~]$
```

6. Làm gián đoạn dịch vụ:

- Xem xét cách chương trình cadet01 xử lý đầu vào được đọc từ mạng. Tìm biến đệm mà dịch vụ ghi vào khi nhận dữ liệu từ mạng và đổi tên thành “buffer”,

kiểm tra trong các hàm liên quan đến biến

```
Decompile:cgc_check-(cadet01)
1 undefined4 cgc_check(undefined4 param_1)
2
3
4 {
5     int iVar1;
6     char local_5c [64];
7     int local_1c;
8     uint local_18;
9     undefined4 local_14;
10    int local_10;
11
12    local_1c = -1;
13    local_14 = 1;
14    for (local_18 = 0; local_18 < 0x40; local_18 = local_18 + 1) {
15        local_5c[local_18] = '\0';
16    }
17    iVar1 = cgc_receive_delim(param_1,local_5c,0x80,10);
```

- Ở đây biến đầu vào là local_5c và mảng này có kích thước là 64
- Thay đổi tên biến đệm thành buffer và biến giới hạn số byte sẽ đọc vào mảng đệm là 0x80(128)
- Khi ta thay đổi tên biến các thì trong các hàm có biến tên cũng tự động thay đổi

```
Decompile:cgc_check-(cadet01)
1 undefined4 cgc_check(undefined4 param_1)
2
3
4 {
5     int iVar1;
6     char buffer [64];
7     int local_1c;
8     uint local_18;
9     undefined4 local_14;
10    int local_10;
11
12    local_1c = -1;
13    local_14 = 1;
14    for (local_18 = 0; local_18 < 0x40; local_18 = local_18 + 1) {
15        buffer[local_18] = '\0';
16    }
17    iVar1 = cgc_receive_delim(param_1,buffer,0x80,10);
18    if (iVar1 == 0) {
19        for (local_18 = 0; buffer[local_18] != '\0'; local_18 = local_18
20            + 1) {
21            local_1c = local_1c + 1;
22        }
23        local_10 = local_1c;
24        if ((local_1c - (local_1c >> 0x1f) & 1U) + (local_1c >> 0x1f) =
25            = 1) {
26            local_10 = local_1c + -1;
27        }
28        for (local_18 = 0; (int)local_18 <= local_10 / 2; local_18 = loc
29            al_18 + 1) {
30            if (buffer[local_18] != buffer[(local_1c - local_18) + -1]) {
31                local_14 = 0;
32            }
33        }
34        if ((buffer[0] == '[') &&
35            (iVar1 = cgc_transmit_all(param_1,"\\n\\nEASTEREGG!\\n\\n",0
36                xf), iVar1 != 0)) {
37            cgc_terminate(0);
38        }
39    }
40}
```

- Cuối cùng, dựa vào những dữ liệu trên, nhập đầu vào để khiến dịch vụ bị lỗi.
Hoàn thành nhiệm vụ nếu netcat hiển thị thông báo “Connection reset by peer”.

- Vì giới hạn mảng đầu vào là 64 ký tự nên ta sẽ nhập đầu vào lớn hơn 64 ký tự thì dịch vụ sẽ bị lỗi

```
[ubuntu@ghidra ~]$ echo "jhsihvoiehvoihoioldvhoihasohcasphfsafjoascnoashfpaosnpfohapsofhpohwsf9ehgpvodhgvhpdagpahighaidh  
giadsghoauidgvbdajgvbadufgvbdfvhadfvagbda fvuaegvoeigvoeagfoefgabdvauwg" | nc 172.25.0.2 4955  
  
Welcome to Palindrome Finder  
  
Please enter a possible palindrome: Ncat: Connection reset by peer.  
[ubuntu@ghidra ~]$
```

Checkwork

```
student@ubuntu:~/labtainer/labtainer-student$ checkwork  
Results stored in directory: /home/student/labtainer_xfer/ghidra  
Labname ghidra  
  
Student | egg | crash |  
=====|=====|=====|  
B20DCAT094 | Y | Y |  
What is automatically assessed for this lab:  
egg: Student provided input necessary to display easter egg  
crash: Student provided input necessary to crash the server
```