Return-to-libc Lab

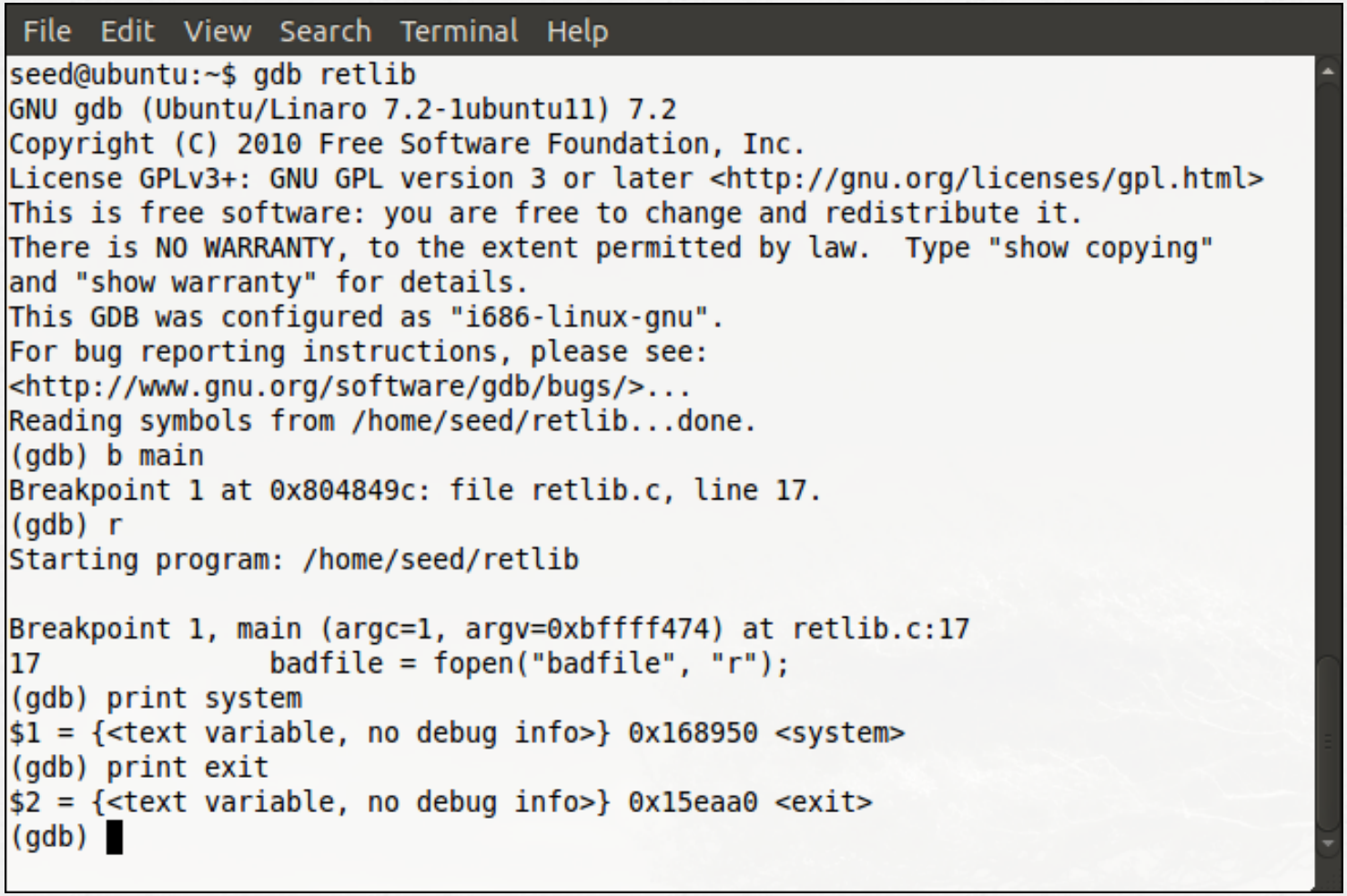
Computer Security

Basic approach

- Use exploit.c and exploit1.c to exploit a return-to-libc vulnerability in the application
- Observations from buffer-overflow-lab:
  - gcc looks like it adds two 32-bit fields just before the variable declarations begin
  - Buffer is 24 bytes away instead of 16 bytes away

gdb for libc function addresses

- Use gdb to get addresses of libc functions
  - Libc function addresses are the same for all programs


File Edit View Search Terminal Help
seed@ubuntu:~$ gdb retlib
GNU gdb (Ubuntu/Linaro 7.2-1ubuntu11) 7.2
Copyright (C) 2010 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://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/seed/retlib...done.
(gdb) b main
Breakpoint 1 at 0x804849c: file retlib.c, line 17.
(gdb) r
Starting program: /home/seed/retlib
Breakpoint 1, main (argc=1, argv=0xbffff474) at retlib.c:17
17 badfile = fopen("badfile", "r");
(gdb) print system
$1 = {<text variable, no debug info>} 0x168950 <system>
(gdb) print exit
$2 = {<text variable, no debug info>} 0x15eaa0 <exit>
(gdb)

Tasks

- Task 1
  - Store the string “/bin/sh” in memory and get it’s address
  - VERIFY that it exists at this address. You can use the following code:

Code to get the address of /bin/sh

```
#include <stdlib.h>

void main() {
 char *shell = getenv("MYSHELL");
 if (shell) {
   printf("%x\n", (unsigned int)shell);
 }
 else
   printf("Environment variable not
      set\n");
}
```

Code to verify the address of /bin/sh

```
#include <stdlib.h>

void main() {
  // Address of MYSHELL from
  // previous program
  char *shell = 0xaddress;
  printf("%s\n", shell);
}
```

Tasks

- Task 1
  - Store the string “/bin/sh” in memory and get it’s address
  - VERIFY that it exists at this address. You can use the following code:
  - “/bin/sh” may exist at an offset that’s slightly above what you get from the sample code given in the lab description
  - Recall the stack layout and where return addresses lie. Also figure out where the parameter addresses need to go.

Tasks

- Task 2
  - Find address of setuid() and use this address for the previous attack.
- Task 3
  - Disable address randomization and repeat