

[Forums / Assignments](#)[Help](#)

\xEE\xEE\xEE\xEE address at end of 65 bytes entered into buf question

✉ You are subscribed. [Unsubscribe](#)

UNRESOLVED

🔖 [project1](#) × + [Add Tag](#)Sort replies by: [Oldest first](#) [Newest first](#) [Most popular](#)[Karen West](#) · 8 hours ago 🔒

When asked to replace \xEE\xEE\xEE\xEE at the end of the buf entry:

771675175\x00AA

\xEE\xEE\xEE\xEE - I tried 2 things that did not work - caused seg faults - and I'm not sure why - I searched the disc. forums and have not yet found the answer, so I ask here. Here is what I tried. I noticed that there are 65 bytes entered at the prompt where buf is filled (56 A's plus the null byte to terminate that 56 A string, then the number 771675175 which is 9 digit characters, making a 65 char input to buf. I then tried to overwrite the 65-68th chars of buf (since buf starts at 0). I tried to enter the address of write_secret, so I entered: \x34\x85\x04\x08 for little endian entry, and that caused a seg fault. Do you need to subtract &ptrs and &write_secret and divide by 4 to get the correct value (similar to pat_on_back calculation) but if so, why do we do that at byte 65 of the entry to buf? I tried that too even though I did not understand it, and it seg faulted too. Maybe what I should have done is &(ptrs + 65) - &write_secret div. by 4? Any help appreciated, since the deadline for late submission of project 1 in Mon. at 8am, and it's currently Sat. at 11:30am. I got started only this week, since I got very delayed and almost dropped the course trying to get the VM to work for me on my machine, and then suddenly at the last second, someone did. ;-)

↑ 0 ↓ · flag

[Carsten Hansen](#) · 8 hours ago 🔒

Getting the seg fault is expected behavior!. You as the black hat crash the system you are trying to hack. Who cares. The question is: Did you recover the secret (before the seg fault)? If yes, then you are done.

↑ 0 ↓ · flag

[+ Comment](#)[Christopher Rose](#) · 8 hours ago 🔒

Karen, you sound close. Follow the logic of the preceding questions. What did you get for the previous question?

- What do you enter so that `ptrs[s]` reads (and then tries to execute) starting from the 65th byte in `buf`, i.e., the location at `buf[64]`? Enter your answer as an unsigned integer.

It should remind you of something in this question.

Why are you adding 65 to `ptrs`? Remember that you want the 65th byte of `buf`, no? And remember that the first byte of `buf` is `buf[0]` so what is `x` if `buf[x]` is the 65th byte? And you are right, `ptrs` are 4 byte pointers, so you need to adjust the byte difference to the number of 4byte pointers in that byte difference.

Good luck! You can do it.

Chris

↑ 0 ↓ · flag

[+ Comment](#)

[Christopher Rose](#) · 8 hours ago 🔒

P.S. You know, this is simple and I got a bit confused by some of your later confused logic. You don't really need to do any calculation here. Just put the address of write secret exactly where the question says. If this doesn't work are you sure that you are running in the right environment?

Remember this from the instructions:

When carrying out the lab, you must follow the instructions given above **exactly** for running the program (using `runbin.sh`) and using GDB (attaching to `wisdom-alt` *in a separate terminal*, and *not* running `gdb ./wisdom-alt`) or else the answers you get may not match the ones we are expecting. In particular, the addresses of stack variables may be different. These addresses might also be different if you have altered any environment variables in the Ubuntu terminals. To confirm that things are as they should be, recall the GDB interaction above, where we print the address `&r` with the result being `0xbffff530` -- if you are not getting that result when you reproduce that interaction then something is wrong. You should restart fresh terminals and begin from scratch, following the instructions exactly.

In my system entering your value `\x34\x85\x04\x08` in the placeholder `\xEE\xEE\xEE\xEE` of the string provided in the question and entering that string at the menu prompt just works with no additional effort. You might get a seg fault even, but do you get the secret to print? That is what matters. If it doesn't work you have probably not set up your environment correctly.

↑ 0 ↓ · flag



Karen West · 7 hours ago



So ignoring my later comment that I tried when it did not work, and only paying attention to the first thing I tried - so just doing what I did at first: When asked to replace \xEE\xEE\xEE\xEE at the end of the buf entry:

```
771675175\x00AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

\xEE\xEE\xEE\xEE - I tried 2 things that did not work - caused seg faults - and I'm not sure why - I searched the disc. forums and have not yet found the answer, so I ask here. Here is what I tried. I noticed that there are 65 bytes entered at the prompt where buf is filled (56 A's plus the null byte to terminate that 56 A string, then the number 771675175 which is 9 digit characters, making a 65 char input to buf. I then tried to overwrite the 65-68th chars of buf (since buf starts at 0). I tried to enter the address of write_secret, so I entered: \x34\x85\x04\x08 for little endian entry, and that caused a seg fault.

This should work - if I set up my environment correctly?

I did run runbin.sh in one terminal, and gdb -p `pgrep wisdom-alt` in a separate terminal.

I saw above that a seg fault is expected, but I should have seen write_secret executed and outputted to stdout.

I'll try it again but I don't think I saw that.

↑ 0 ↓ · flag

[+ Comment](#)

Karen West · 3 hours ago



I tried this one again and it does not print the secret key - this is my last problem with project 1, and I can submit it once I figure it out! I once again entered: \x34\x85\x04\x08 after the: 771675175\x00(then 56 A's) - that did NOT print the secret key - not sure what is wrong there. Any more help here appreciated!

↑ 0 ↓ · flag

Carsten Hansen · 3 hours ago

There are only 54 A's. 9 digits (from 771675175), a null byte, 54 A's give you an offset of 64.

In gdb if you do "x/8xb &buf[64]" do you see the right address?





After the C instructions fptr tmp = ptrs[s]; what is the value of tmp? It should be 0x8048534.

↑ 0 ↓ · flag

[+ Comment](#)

New post

To ensure a positive and productive discussion, please read our [forum posting policies](#) before posting.

B | *I* |  |  |  Link | `<code>` |  Pic | Math | | Edit: Rich | Preview

☐ Resolve thread

This thread is marked as unresolved. If the problem is fixed, please check the above box and make a post to let staff know that they no longer need to monitor this thread.

☐ Make this post anonymous to other students

☒ Subscribe to this thread at the same time

Add post

\xEE\xEE\xEE\xEE address at end of 65 byte...

<https://class.coursera.org/softwaresec-001/f...>