Christopher Gonzalez

Professor Jennifer Holst

CSCI 401

6 March 2025

**pwn.college Lab 2**

A screenshot of a computer

AI-generated content may be incorrect.

**Challenge 1: Loading From Memory**

A screenshot of a computer

AI-generated content may be incorrect.

To obtain this flag, I started the challenge and opened the VSCode Workspace. The instructions for this challenge introduced me to the concept of computer memory represented as a numeric address which can be accessed using commands from the last lab (mov, rax, rdi). My objective was to retrieve the secret number stored in memory address 133700. I created a new file titled “lab2.s” and entered “mov rdi, [133700]”. By putting the address in brackets, I am accessing the contents within. Afterwards, I provided the information needed to exit the program (mov rax, 60 and syscall) and checked my work by entering “/challenge/check lab2.s”. The terminal verified my work and gave me the flag highlighted in white.

**Challenge 2: More Loading Practice**

A screenshot of a computer

AI-generated content may be incorrect.

To obtain this flag, I started the challenge and opened the VSCode Workspace. This challenge was nearly identical to the one previously as the purpose was to give me extra practice with loading from computer memory. This time, I had to access the secret value stored in address 123400 instead of 133700. The only modification required was changing the value inside the bracket to 123400 (the rest did not have to change). After running “/challenge/check lab2.s” on the terminal, the flag was given to me.

**Challenge 3: Dereferencing Pointers**

A screenshot of a computer

AI-generated content may be incorrect.

To obtain this flag, I started the challenge and opened the VSCode Workspace. This challenge introduced me to the concept of dereferencing pointers. In most cases, memory addresses are stored in registers, and so users have to use the values in the registers to point to the data in memory. My objective was to dereference secret data that was stored in rax into rdi. I accomplished this by entering “mov rdi, [rax]” in the first line of my code. This ensured the rdi uses and accesses the secret value stored in rax. I checked my work and got the flag.

**Challenge 4: Dereferencing Yourself**

A screenshot of a computer

AI-generated content may be incorrect.

To obtain this flag, I started the challenge and opened the VSCode Workspace. This challenge was similar to the one before, but as the name suggests, this challenge required me to dereference myself rather than a pointer. The instructions for this challenge stated that the secret value was stored in rdi, suggesting there was no need to use rax other than to exit the program along with syscall (these two command will always be needed at the very end of the file). Since there was no numeric address stated, I had to dereference myself by entering “mov rdi, [rdi]”. I proceeded to check my work at the terminal, and the flag was revealed.

**Challenge 5: Dereferencing with Offsets**

A screenshot of a computer

AI-generated content may be incorrect.

To obtain this flag, I started the challenge and opened the VSCode Workspace. This challenge introduced me to the concept that a pointer could point to a collection of data (like an array) rather than just individual addresses. Because of this, there needs to be a way to reference specific data within that collection. This can be done like so: “mov rax, [rdi+1]”, where the number represents the specific content the user wishes to access in the sequence. For this challenge, I had to extract the secret value which was stored in the 8th offset. To do this, I entered “mov rdi, [rdi+8]” and checked my work in the terminal. After this, the flag was given to me.

**Challenge 6: Stored Addresses**

A screenshot of a computer

AI-generated content may be incorrect.

To obtain this flag, I started the challenge and opened the VSCode Workspace. The instructions for this challenged revealed that my goal was to use rdi to read numeric address 567800, dereference it, and exit the program. I began by reading the numeric address. This was accomplished by entering “mov rdi, [567800]” in the first line of the lab2.s file. Afterwards, I dereferenced myself by writing “mov rdi, [rdi]”. The third requirement was already satisfied. I checked my work by entering “/challenge/check lab2.s” in the terminal, revealing the flag.

**Challenge 7: Double Dereference**

A screenshot of a computer

AI-generated content may be incorrect.

To obtain this flag, I started the challenge and opened the VSCode Workspace. For this challenge, I was required to dereference a value in rax and then read the the same value using rdi. In other words, this is known as double dereferencing. The instructions stated to frist dereference the value stored in rax. I did this by entering “mov rax, [rax]”. Then, I read the value by entering “mov rdi, [rax]”. Once I checked my work via the terminal, the flag was revealed.

**Challenge 8: Triple Dereference**

A screenshot of a computer

AI-generated content may be incorrect.

To obtain this flag, I started the challenge and opened the VSCode Workspace. This challenge was an add on to the previous one, where this time I had dereference three times instead of two. The instructions for this challenge hinted that the rdi register was the only one I needed for triple dereferencing, so I entered “mov rdi, [rdi]” three times to extract the contents hidden. After checking my work through the terminal, I got the flag.