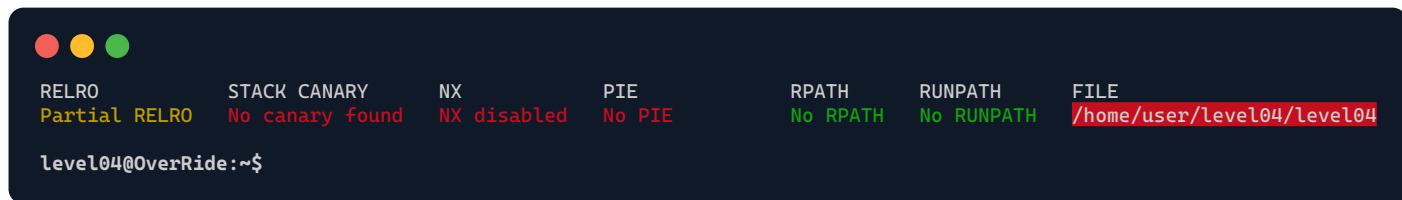


./level04



Decompiled file with **Ghidra**:



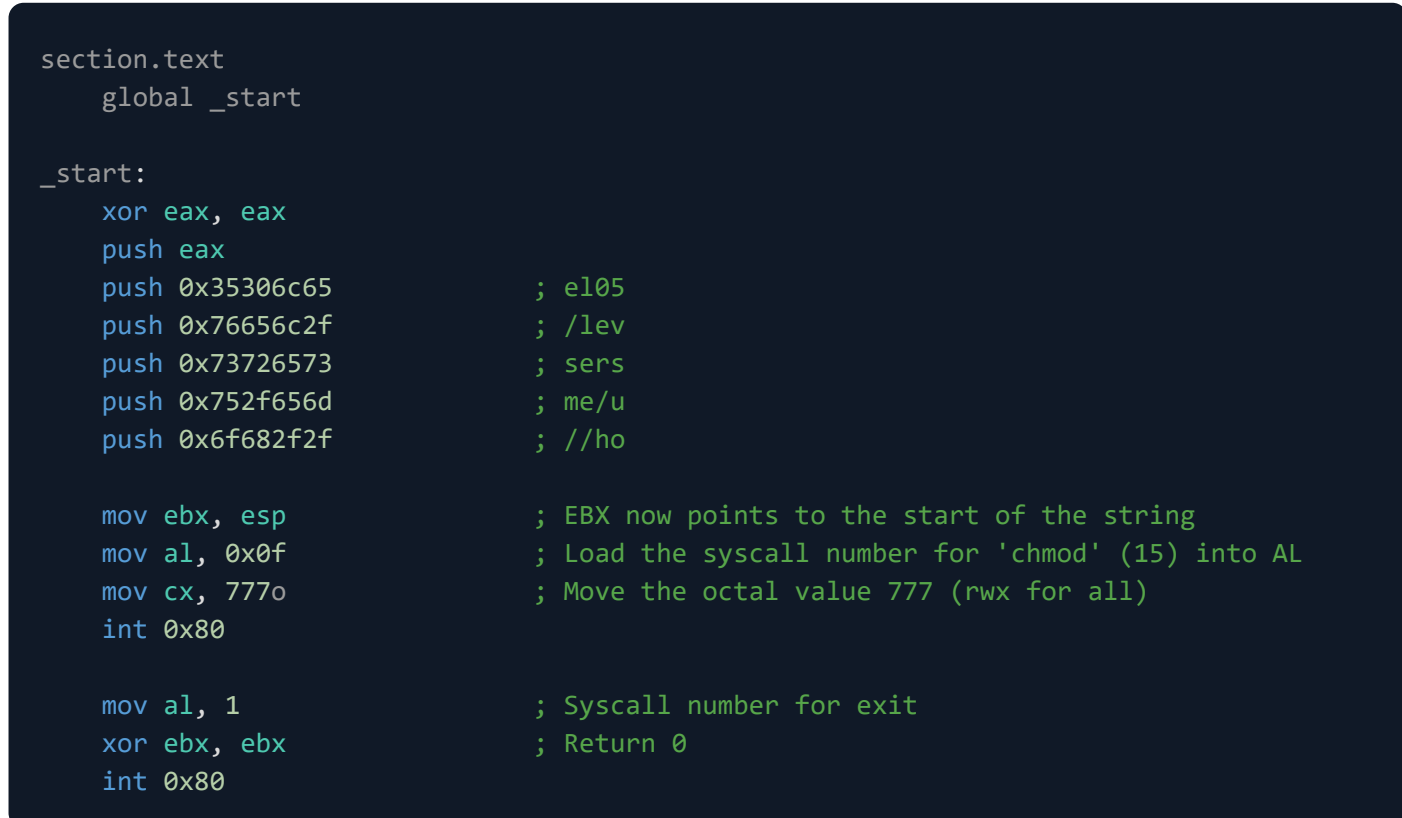
This program establishes a simple debugging environment that prevents the execution of the **exec()** system call within a child process.

It employs the **ptrace** system call to **trace system call** invocations by the child. When the child process attempts to execute **exec()**, which is identified by the **syscall 11**, the parent process terminates the child. This effectively prevents the typical exploitation technique where **shellcode** would use **exec()** to spawn a **shell**, thus mitigating a common security threat.

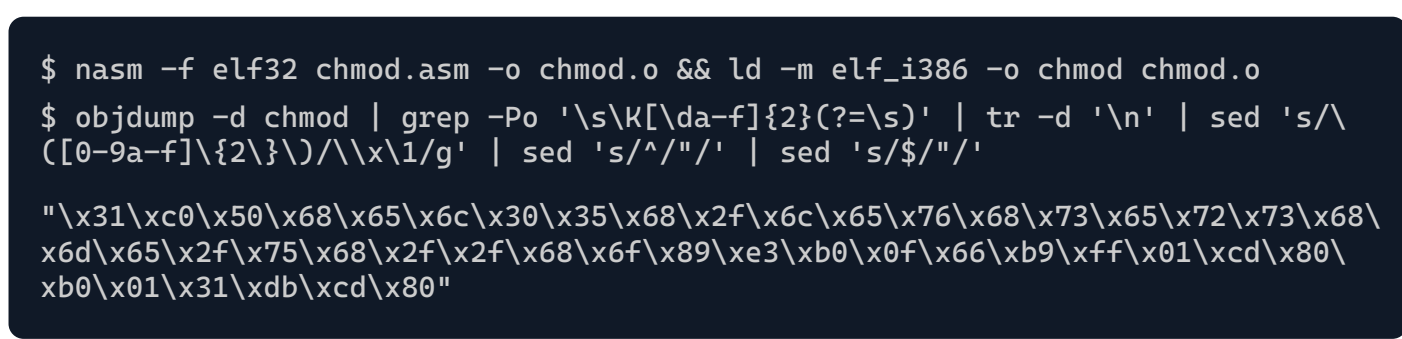
However, the program's security measures are focused narrowly on the **exec()** system call. It does not account for other system calls, for example the child process is still capable of using the **chmod()** system call to change file permissions.

We can exploit this to alter permissions of the level05 home folder.

To achieve this, we'll craft a **shellcode**—derived from our **assembly** program—that, when injected, will change the level05 directory's access rights:



We assemble the code with **nasm** and link it with **ld**:



With our **shellcode** ready, we'll exploit the vulnerable **gets(buffer)** function to trigger a *buffer overflow*, thereby overwriting the **main** function's **return address** to redirect execution flow to our **shellcode**'s entry point. With the help of **gdb**, we'll determine the buffer's starting position and the correct offset:

