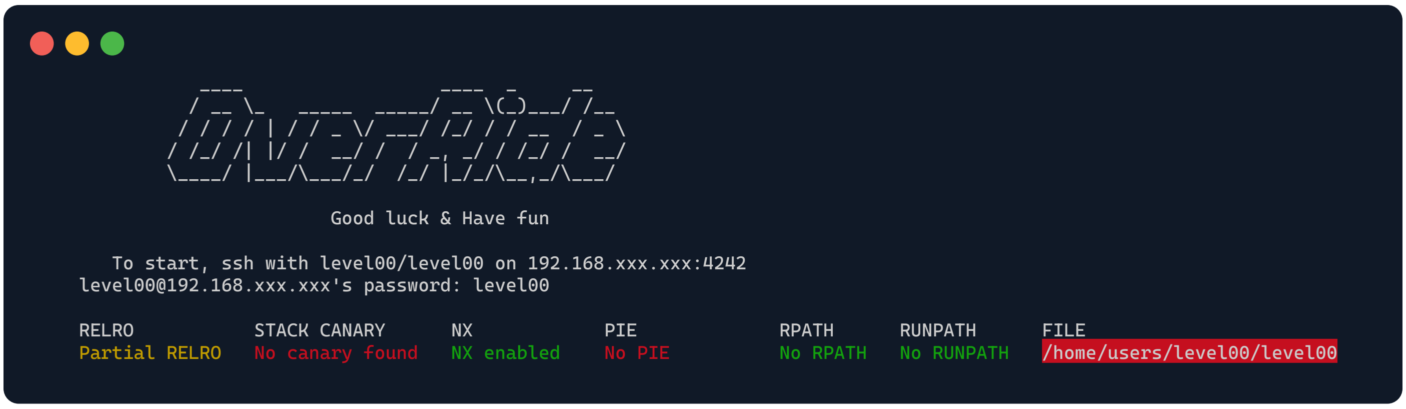
./level00



```
           ____                  ____  _     __
          / __ \_   _____  _____/ __ \(_)___/ /__
         / / / / | / / _ \/ ___/ /_/ / / __  / _ \
        / /_/ /| |/ /  __/ /  / _, _/ / /_/ /  __/
        \____/ |___/\___/_/  /_/ |_/_/\__,_/\___/

                       Good luck & Have fun

   To start, ssh with level00/level00 on 192.168.xxx.xxx:4242
level00@192.168.xxx.xxx's password: level00

RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      FILE
Partial RELRO   No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   /home/users/level00/level00
```

In the home directories of the **users** in the **OverRide** project, each user possesses an **executable file** formatted as **ELF 32-bit** or **64-bit**. To transfer these files to our local system, we consistently utilized the **scp** command with the following syntax:

```
scp -P 4242 user@192.168.xxx.xxx:filename localfilename
```

We decompiled each file with ***Ghidra***. Given that the direct translation from **assembly** can be nebulous at times, we took the liberty of renaming variables and making slight code adjustments for better readability.

In the different levels of the project, every time we establish an **SSH** connection to a **levelx** user, the terminal presents us with a comprehensive list of security protections:

RELRO: Ensures certain memory sections, including the Global Offset Table, are read-only post program initialization, making overwrites tough.

STACK CANARY: any small random value placed on the stack to detect buffer overflows. If a buffer overflow occurs, the canary value will likely be overwritten

NX (No-eXecute): A CPU feature that designates memory areas as non-executable, hindering exploits relying on executing code from these regions.

PIE: Allows executables to operate at various memory addresses, enhancing memory unpredictability when paired with ASLR.

RPATH/RUNPATH: ELF binary attributes dictating dynamic library search paths. Misconfigurations can lead to library hijacking.

./level00^{2}

```c
int main(void)
{
    int inputValue;

    puts("***********************************");
    puts("*            -Level00 -           *");
    puts("***********************************");
    printf("Password: ");

    scanf("%d", &inputValue);

    if (inputValue != 0x149c) // 5276
    {
        puts("\nInvalid Password!");
    }
    else
    {
        puts("\nAuthenticated!");
        system("/bin/sh");
    }

    return inputValue != 0x149c;
}
```

To successfully enter the conditional **if** statement in the code, the program must receive 5276 as input. If this condition is met, the program spawns a **shell** that allows us to operate as user **level01**.

```
level00@OverRide:~$ {
    python -c 'print("5276")';
    cat <<< "cd ../level01 && cat .pass";
} | ./level00

***********************************
*            -Level00 -           *
***********************************
Password:
Authenticated!
uSq2ehEGT6c9S24zbshexZQBXUGrncxn5sD5QfGL

level00@OverRide:~$ su level01
Password: uSq2ehEGT6c9S24zbshexZQBXUGrncxn5sD5QfGL

level01@OverRide:~$
```