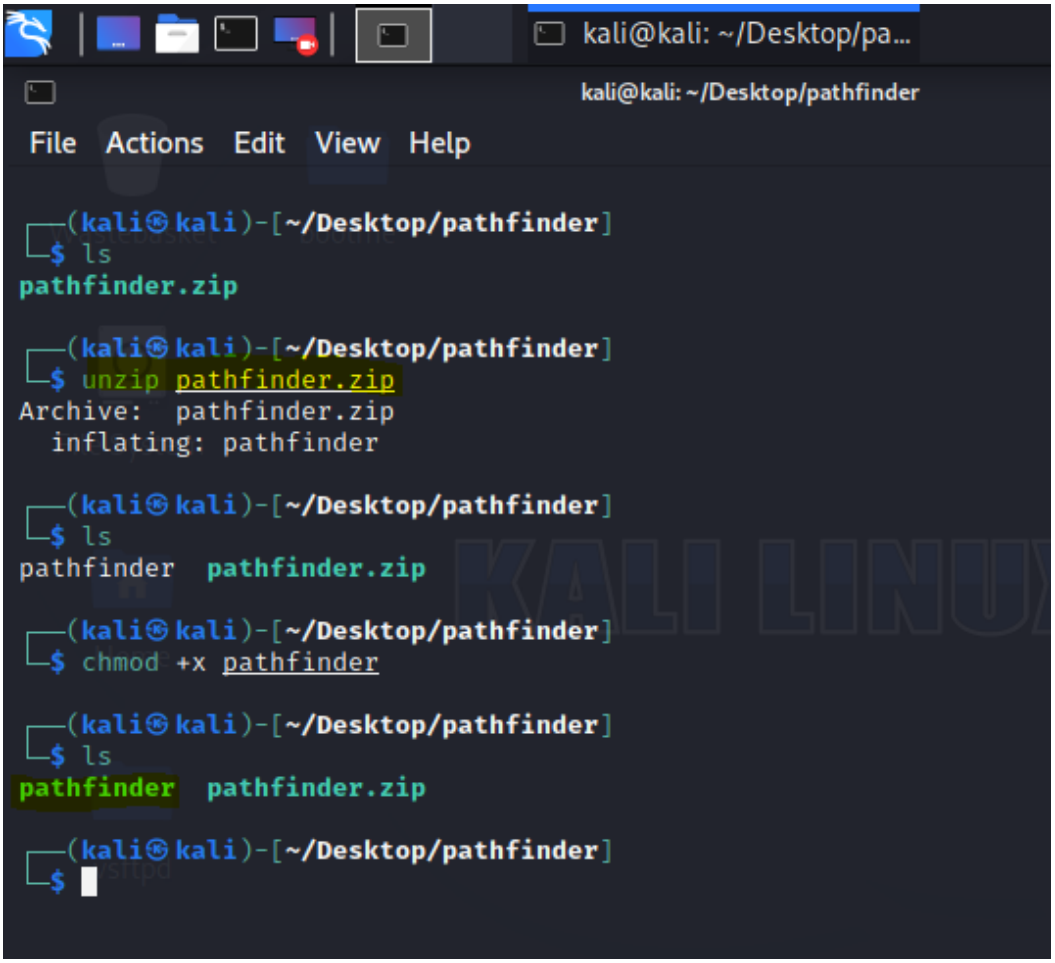
1. Unzip the file & make the extracted ELF file executable.



2. Before running the program, the file can be opened in Ghidra (https://ghidra-sre.org/) & the following is noticeable when analyzing the function FUN_*080485c8* (Screenshot 1) & *FUN_08048569 (*Screenshot 2):
    a. The Input must be 8 Characters, due to the length of the while loop
    b. There is a String Comparison (*strcmp*) with the input & a variable
    c. The Input String must be in capital letters due to the condition of the if statement (**0x41 = A / 0x5a = Z**)

```
local_28 = 0x53485943;
local_24 = 0x55425a5a;
printf("Enter the password: ");
__isoc99_scanf(&DAT_0804e332,local_3c);
local_44 = 0;
while (local_44 < 8) {
  cVar1 = FUN_08048569((int)local_3c[local_44],local_44 + 8);
  local_3c[local_44] = cVar1;
  local_44 = local_44 + 1;
}
iVar2 = strcmp(local_3c,(char *)&local_28);
if (iVar2 == 0) {
  puts("You reached your destination.");
  puts("A massive abandoned spacestation appears in front of you");
}
else {
  puts("You died.");
}
```

```
int FUN_08048569(int param_1,int param_2)

{
  if ((0x40 < param_1) && (param_1 < 0x5b)) {
    return (param_1 + -0x41 + param_2 * 0x1f) % 0x1a + 0x41;
  }
  puts("You died.");
                    /* WARNING: Subroutine does not return */
  exit(1);
}
```

3. From the information gathered in Step 2., the file can be executed with *ltrace* to track the system calls & parameters of any invoked functions. When prompted for a password, enter a String with 8 Characters & all caps (here: *OOOOOOOO*).

```
┌──(kali㉿kali)-[~/Desktop/pathfinder]
└─$ ltrace ./pathfinder
__libc_start_main(0x80485c8, 1, 0xffd0e004, 0x804e290 <unfinished ...>
printf("Enter the password: ")                          = 20
__isoc99_scanf(0x804e332, 0xffd0df14, 2, 0xf7fded66Enter the password: OOOOOOOO
)      = 1
strcmp("CHMRWBGL", "CYHSZZBU")                          = -1
puts("You died."You died.
)                                          = 10
+++ exited (status 0) +++

┌──(kali㉿kali)-[~/Desktop/pathfinder]
└─$
```

We see the Input is manipulated (with the function *FUN_08048569*) & compared to the **concatenated & reversed** variables (*local_28* & *local_24*) that are hex representations of "SHYC" & "UBZZ".

4. Bruteforce the password by trying a character until the respective inputted character matches the character that it's compared to.
   This could/should have been done with a python script, but I assumed that it would take less time to do it by hand. This assumption was reinforced by the fact, that the alphabet order was kept intact, so that not every character appeared randomly and therefore not the entire alphabet would need to be tested.

```
┌──(kali㉿kali)-[~/Desktop/pathfinder]
└─$ ltrace ./pathfinder
__libc_start_main(0x80485c8, 1, 0xfff19434, 0x804e290 <unfinished ...>
printf("Enter the password: ")                          = 20
__isoc99_scanf(0x804e332, 0xfff19344, 2, 0xf7f77d66Enter the password: OFJPRMJW
)      = 1
strcmp("CYHSZZBT", "CYHSZZBU")                          = -1
puts("You died."You died.
)                                          = 10
+++ exited (status 0) +++

┌──(kali㉿kali)-[~/Desktop/pathfinder]
└─$ ltrace ./pathfinder
__libc_start_main(0x80485c8, 1, 0xffa0ade4, 0x804e290 <unfinished ...>
printf("Enter the password: ")                          = 20
__isoc99_scanf(0x804e332, 0xffa0acf4, 2, 0xf7fbcd66Enter the password: OFJPRMJX
)      = 1
strcmp("CYHSZZBU", "CYHSZZBU")                          = 0
puts("You reached your destination."You reached your destination.
)                          = 30
puts("A massive abandoned spacestation"...A massive abandoned spacestation appears in front of
you
)                  = 57
+++ exited (status 0) +++
```

5. At some point, you'll get the flag as the inputted string.