Dosyayı file komutu ile incelediğimizde ELF çalıştırılabilir dosyası olduğunu görüyoruz.

```
reti@reti:~/cyber sec/stm-ctf$ file password
password: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, not stripped
reti@reti:~/cyber sec/stm-ctf$
```

Dosyayı çalıştırdığımızda parola istiyordu. Deneme yaptığımızda Wrong, Try Again! mesajı çıkıyordu.

```
reti@reti:~/cyber sec/stm-ctf$ ./password
Enter your password : 1234
Wrong, Try Again!
```

Bu noktadan sonra hemen dosyayı analiz etmek için Ghidra'yı açtım. Fonksiyon isimlerine, stringlere baktığımızda dosyanın Go ile yazılmış olduğu aklıma geldi. Daha önceki senelerde de go ile yazılmış dosyalarla ilgili sorular sorulduğunu da okumuştum.
Go derlendiği zaman main fonksiyonu main.main adında bulunur. Ghidra ile hemen burayı incelediğimizde güzel bir karşılasştırma görüyoruz.

```
void main.main(void)

{
  undefined1 **ppuVar1;
  long in_FS_OFFSET;
  long *local_88;
  undefined1 *local_10;

  ppuVar1 = (undefined1 **)(*(long *)(in_FS_OFFSET + 0xfffffff8) + 0x10);
  if (*ppuVar1 <= &local_10 && &local_10 != (undefined1 **)*ppuVar1) {
    runtime.newobject();
    local_10 = main.statictmp_0;
    fmt.Fprint();
    fmt.Fscan();
    if (*local_88 == 0xd6216b41fa3e22f) {
      fmt.Fprintln();
    }
    else {
      fmt.Fprintln();
    }
    return;
  }
  runtime.morestack_noctxt();
  main.main();
  return;
}
```

Bu sayıyı denediğimizde Succes :) mesajını alıyoruz. STMCTF{0xd6216b41fa3e22f} veya STMCTF{d6216b41fa3e22f} denediğimizde flag'i kabul etmedi. Decimale çevirip verdiğimizde puanımızı almış olduk.

```
reti@reti:~/cyber sec/stm-ctf$ ./password
Enter your password : 0xd6216b41fa3e22f
Success :)
```