My goal with this puzzle is to introduce some of the fundamental skills in reverse engineering; including, Static Analysis and Dynamic Analysis, such as debugging. My puzzle will consist of two parts.

The First part will be using static analysis to pull a passphrase from a direct comparison, with some simple obfuscation so it can't be found directly with strings. This is similar to the "unpackme" challenge in picoGym. Pw: "I_H8_stup1d_dum_g33se!"

The Second part will be "MFA" using dynamic analysis. I will use obfuscation techniques to make debugging the focus instead of static analysis. Pw: "Geese_B3_Gon3_for_g00d!!!!"

I completed the first part very easily, however problem 2 is where I struggled a bit. I had to figure out how to dynamically define something that wouldn't be stored in plaintext, and how to compare it to a password without the password being clearly defined. This is what I came up with.

1. Use some arithmetic to dynamically create a secret string (this secret is constant)
2. XOR the password of my choice using the secret (Then only include the encoded password in the binary) Secret = NUBIPWDKRYFMTAHO
3. XOR the user input and compare it to my encoded password
4. Decode a secret message with the same XOR

How I would Solve it

Part 1 Static Analysis

It just concats a series of 4 strings into a password order.

Part 2 Dynamic Analysis

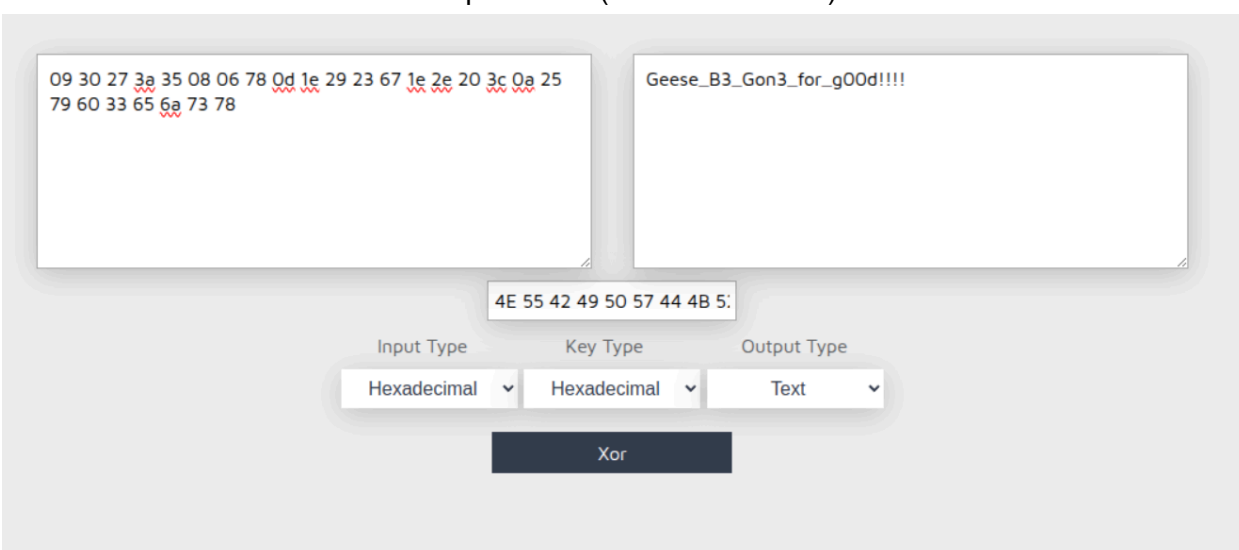
Use gdb (I also have gef installed) to pull the XOR key from memory

```
──────────────────────────────────────────────────────────────────── stack ────
0x00007fffffffdab8│+0x0000: 0x00005555555558c2  →  <CheckCheck+00c7> mov QWORD P
TR [rbp-0x150], rax       ← $rsp
0x00007fffffffdac0│+0x0008: 0x0000000029349b4a
0x00007fffffffdac8│+0x0010: 0x0000000000000000
0x00007fffffffdad0│+0x0018: 0x0000000000000000
0x00007fffffffdad8│+0x0020: 0x0000555555559ac0  →  "NUBIPWDKRYFMTAHO"
0x00007fffffffdae0│+0x0028: 0x00007ffff7e045c0  →  0x00000000fbad2a84
0x00007fffffffdae8│+0x0030: 0x000000000000001b
0x00007fffffffdaf0│+0x0038: 0x00005555555592a0  →  "Please enter a SECOND super
secret password:  for [...]"
```

Pull the encoded string breaking at the strcmp

```
thestair@thestair-VMware-Virtual-Platform: ~/Documents/CTF Challenge
      0x5555555551ca <strtol@plt+000a> nop    WORD PTR [rax+rax*1+0x0]
──────────────────────────────────────────────────────────── arguments (guessed) ──
strcmp@plt (
   $rdi = 0x0000555555559b00 → "1e 34 31 3a 27 38 36 2f",
   $rsi = 0x00005555555560f8 → "09 30 27 3a 35 08 06 78 0d 1e 29 23 67 1e 2e 20
3c[...]",
   $rdx = 0x00005555555560f8 → "09 30 27 3a 35 08 06 78 0d 1e 29 23 67 1e 2e 20
3c[...]"
)
──────────────────────────────────────────────────────────────────────── threads ──
[#0] Id 1, Name: "a.out", stopped 0x5555555559bc in CheckCheck (), reason: SINGL
E STEP
──────────────────────────────────────────────────────────────────────── trace ──
[#0] 0x5555555559bc → CheckCheck()
[#1] 0x555555555cc8 → main()
──────────────────────────────────────────────────────────────────────────────
gef➤  info register rdx
rdx            0x5555555560f8      0x5555555560f8
gef➤  info register $rdx
rdx            0x5555555560f8      0x5555555560f8
gef➤  x/s $rdx
0x5555555560f8: "09 30 27 3a 35 08 06 78 0d 1e 29 23 67 1e 2e 20 3c 0a 25 79 60
33 65 6a 73 78"
gef➤
```

Reverse the XOR for to recover the password (online tools work)



Run the program and enter both passwords to get the decoded flag (Uses the same XOR)

```
thestair@thestair-VMware-Virtual-Platform:~/Documents/CTF Challenge$ ./goose_be_
gone
If the geese are running rampant, this device is the key,
but first you'll have to provide the password for me.
Please Enter the Password: I_H8_stup1d_dum_g33se!
Success!

Wait Wait Wait, hold one a Minute...
How do I really know you aren't a GOOSE?
I will need another piece of information.
Please enter a SECOND super secret password: Geese_B3_Gon3_for_g00d!!!!
MFA correct! Access granted.
You hear a faint rumbling and the ground begins to shake
 You see a huge blimp rising out of the pond
 the geese all flee in fear!
 the park is free again!
 Here is your flag: Super_Goose_Attack_Blimp
```