

B”h

Write up 5.2

Objective:

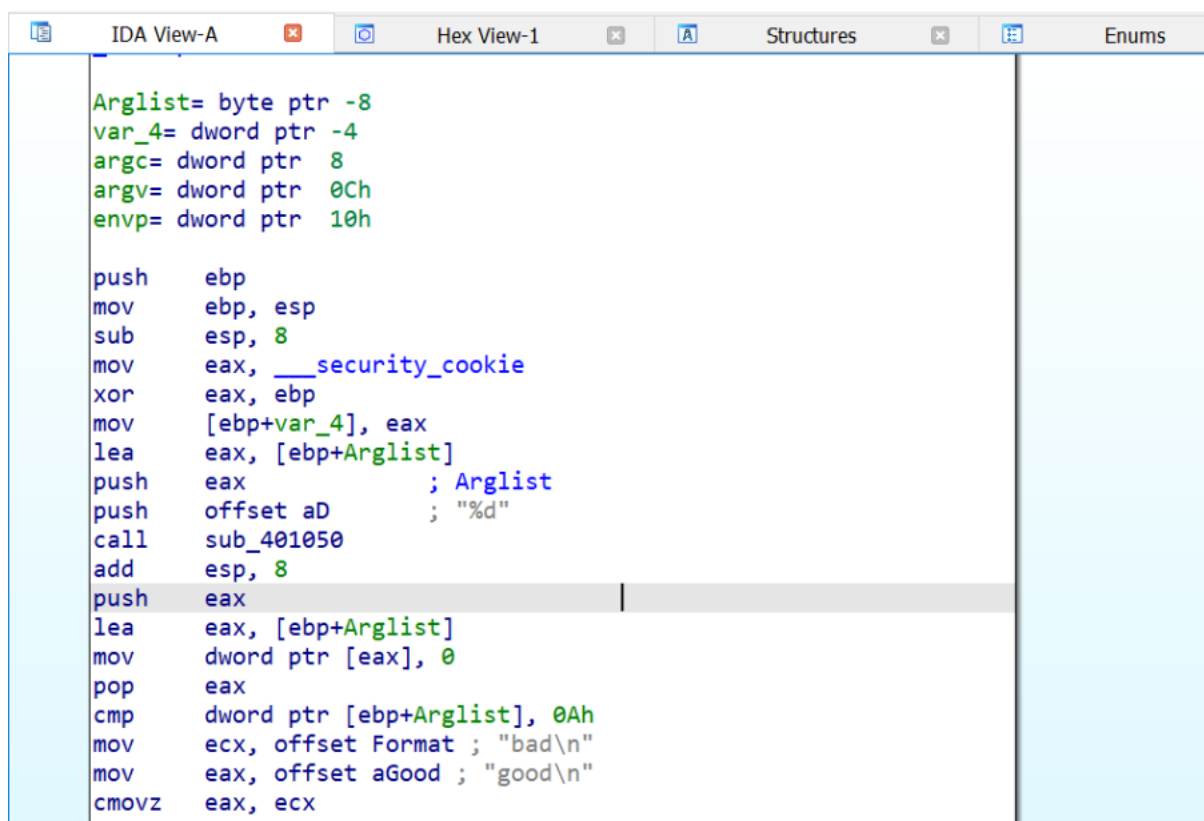
To get the code to get only the right password.

Topics Covered:

1. ida64
2. patching
3. chatgpt

Procedure:

Here is my idea on how to solve this challenge :D. Let’s perform static analysis on the binary file by using ida64 in windows machine (my favourite debugging tools).

The image shows a screenshot of the IDA View-A window. The window has several tabs at the top: 'IDA View-A', 'Hex View-1', 'Structures', and 'Enums'. The 'IDA View-A' tab is active, displaying assembly code. The code starts with variable declarations: 'Arglist= byte ptr -8', 'var\_4= dword ptr -4', 'argc= dword ptr 8', 'argv= dword ptr 0Ch', and 'envp= dword ptr 10h'. The main function body begins with 'push ebp', 'mov ebp, esp', and 'sub esp, 8'. It then moves the address of '\_\_\_security\_cookie' into 'eax', XORs it with 'ebp', and stores the result at '[ebp+var\_4]'. Next, it loads 'Arglist' into 'eax' and pushes it, followed by pushing an offset 'aD' with a format string '%d'. It then calls 'sub\_401050', adds 8 to 'esp', and pushes 'eax'. The code continues with 'lea eax, [ebp+Arglist]', 'mov dword ptr [eax], 0', 'pop eax', 'cmp dword ptr [ebp+Arglist], 0Ah', 'mov ecx, offset Format ; "bad\n"', 'mov eax, offset aGood ; "good\n"', and 'cmovz eax, ecx'. The window also shows a 'Hex View-1' tab and 'Structures' and 'Enums' tabs on the right side.

After glance through the assembly code, it looks like there is a stream (scanf) function that receive an integer from the user. After that its seems like we are compering the input we got with Ah(10) and then printing good/bad according to the compression. The problem was that compression got done with cmovle. So I checked in msdn a commend that will fit.

## Description

---

- mov if Zero Flag (ZF) is not set

```
cmovnz dest, source
```

```
IF ZF == 0:DEST = SOURCE
```

So I replaced `cmovle` with `cmovnz`.

I checked it out but the code still wasn't working.

I quickly found out the problematic part. I skipped on this:

The problem was that `eax` was getting a 0 instead of the input. So I remade this line as `{mov edx, 0}` and the problem solved.

That's all for the write up, I hope you guys did enjoy my first ever write up on reverse engineering challenge. Cheers! I'm also hope that i can continue to publish some write up for the interesting challenges in the future.