CS165 Project 2 Part 1 Report (Truncated)

Professor Zhiyun Qian University of California, Riverside

By Sanchit Goel and Jiacheng Hou.

Is DL zero?

(Reached the end)

Continue to

check password

Yes

No

Overview

Print prompt Get username input Running the toy program with random username and password we got Push to stack (see figure 1): Print prompt Get password input PS F:\Study\2021 Fall\CS_165\project2> .\authenticate_yourself.exe Please enter your username and password to be authenticated: Push to stack Username: cnweo485yohnwev8 Password: w349p8twm4chwc Check username Incorrect username. You are not allowed to enter the system Figure 2 Original Printout Correct username? Here is an overview of the logic of the authentication function at sub 401080 (see figure 2): Yes Check password Load inputed username address to EAX Correct password? Load correct username Yes address to ECX rint incorrect password Print incorrect username Print flag message message Load the byte of Load the next byte of Figure 1 Authentication Function Overview inputed username inputed username at addr [EAX] to DL at addr [EAX+1] to DL Compare it to the same byte Compare it to the next **Bypassing Username Check** of the correct byte of the correct username at addr [ECX] username at addr [ECX+1] Here is a closer look on how the function checks No No Are they Are they if the user enters the correct username (see the same? the same? figure 3): Yes Yes Increment

Figure 3 Check Username Algorithm

EAX, ECX

by 2

ts DL NOT zero?

(Not at the end)

Continue to print incorrect

username message

Yes

We can see, the function compares the inputted username against the correct username Using a loop. Each loop, it checks if the bytes are the same and if the byte in the inputted username is zero (indication it has reached the end.)

To bypass the username check, instead of comparing the byte from the inputted username in DL to the byte from the correct username at addr[ECX], we changed it to comparing to itself, which always yields true.

Similarly, we changed the comparison on the next byte of the username to always be true (see figure 4 for table).

By making these changes, we successfully bypassed the program's username check and the program now always accepts the inputted username.

After entering a random string as username, we got (see figure 5):

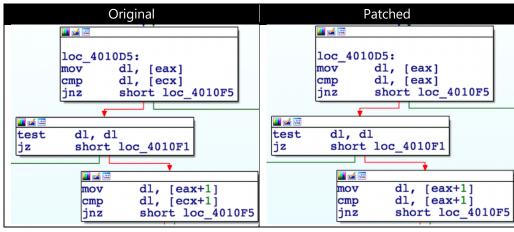


Figure 4 Patched Code to Bypass Username Check

```
PS F:\Study\2021 Fall\CS_165\project2> .\authenticate_yourself_no_username.exe
Please enter your username and password to be authenticated:
Username: sergesjgc54cmow
Password: fesu95hocy8mw
Incorrect password. You are not allowed to enter the system.

Figure 5
Printout After
Bypassing
Username Check
```

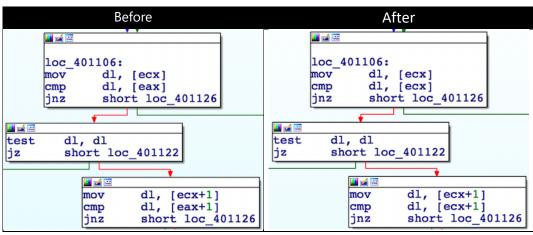
Bypassing Password Check

We found the mechanism for checking password is the same as for checking username, with only some differences:

- 1. Load the correct password address and inputted password into EAX and ECX respectively.
- 2. If the bytes are different, then print incorrect password message.
- 3. If the end of the inputted password is reached and all its bytes are the same as those of the correct one, the flag string is printed.

So, we made the similar changes to the comparisons so they would always be true (see figure 6 for table).

Figure 6
Patched Code to Bypass
Password Check



Conclusion

Since both username and

password checks are bypassed, the toy program's entire authentication is bypassed. The program now gives out the flag string no matter what username or password we give.

Running the patched program with random strings as username and password, we got (see figure 7):

```
PS F:\Study\2021 Fall\CS_165\project2> .\authenticate_yourself_bypassed.exe Please enter your username and password to be authenticated: Username: vesc5h8mo Password: fnes,so58chtw8ch Here's your flag:34gdfh340234
```

Figure 7 Printout After Bypassing All Authentication Checks

And we got the flag: 34gdfh340234