CS165: Project 2 Report

Alexander Choi (861 268 756)
Audrey Der (861 221 280)

Due: February 12, 2019

1 Part 1

The assignment asks calls for the bypassing of authentication used in a toy application created for this class. Upon successful authentication bypass, we are given a unique string that is submitted as part of the project outcome. **Flag**: **34gdfh340234**

1.1 Methodology

Finding the Flag: Using the comments in the binary, namely "Here is your flag:", the location of the jump instruction jumping to the assembly detailing the authentication pop up was found. After scrolling up a bit, there was another comment, "Please enter your username and password:". Looking at the Strings window and all the hardcoded strings, we found a string called "admin". Immediately below it is a string of seemingly random collection of letters, symbols, and numbers. We thought it might be the password. (It was.) Now running the program given and providing this string that might be the password, the flag was printed to the console.

Patching the binary: We opted to insert a `jmp` instruction immediately after the program starts to jump to the instruction that prints the flag. See Figure 1.

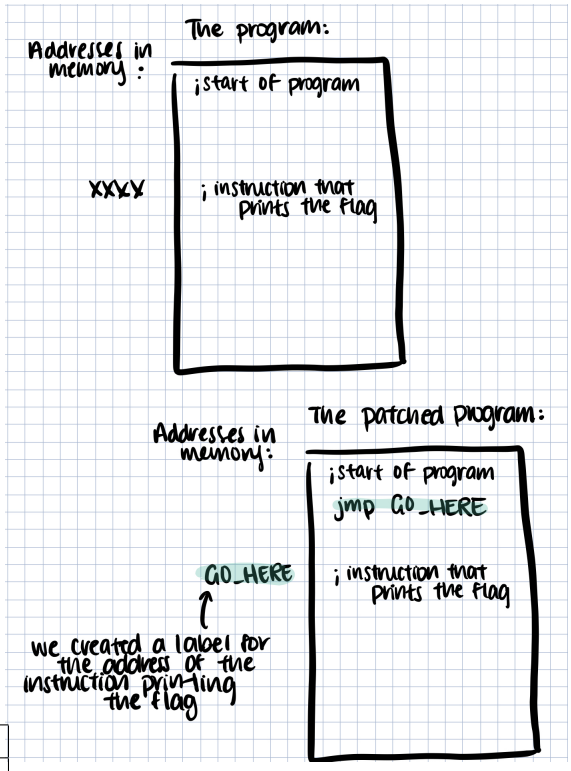

Figure 1: Part 1 Modification Diagram

1.2 Instruction Modifications

Address of Inserted Instruction	Original Instruction
4013bc	N/A
"Operation"	**New Instruction**
Added	`jmp 0x40112f`

2 Part 2

2.1 Methodology

2.1.1 Removing the Banner

Knowing that the banner included the string "Expired", we enabled unicode in the Strings window and searched for "expired". One of the results returned lead to the string that contained the banner message. We followed "up" the graph to

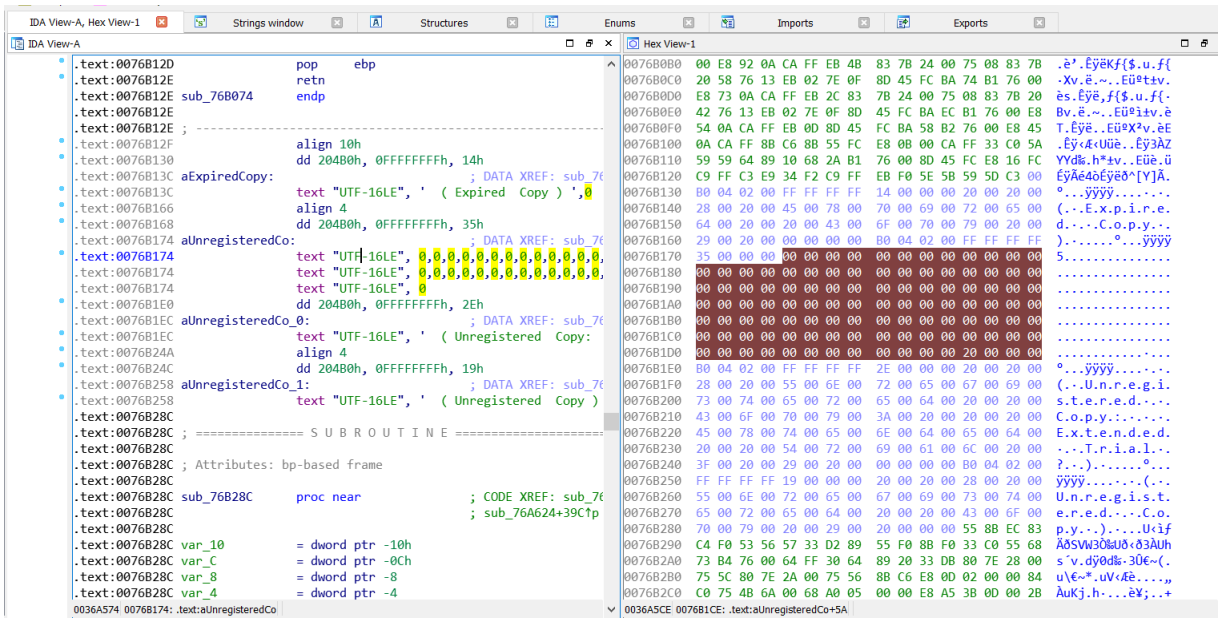

Figure 2: Replaced Banner String with Empty String

where the string was called. Synchronizing IDA View-A and
Hex View (via Hex Dump) windows, which allowed me to "find" where the string was stored in memory in the hex window. **These addresses were** `0x007613170` **through** `0x0076B1D0`. We changed the banner's string's hex values to all '0's to replace it with the empty string which "got rid of" the banner (pictured in the figure below).

2.1.2 Removing the Pop-Up

We found the label `start` and confirmed that it is the program's entry point. Then, we iterated over the code, setting breakpoints on each call instruction and stepping through the program's execution in the debugger until we found an instruction after which the pop up was created. The subroutine called by this instruction is at address `0x0081637c`. We used IDA to list the references to this subroutine. There were two, at addresses `0x00829173` and `0x00858c90`. For each of these references, we used the IDA Graph View to determine the nearest conditional branch instructions which would avoid these addresses. These were a jbe instruction at address `0x00858c83` and a jnz instruction at address `0x0082912e`. We changed these instructions to non-conditional jumps.

2.1.3 Removing the Pop-Up: Changed, Added, and Deleted Instructions

Address of Inserted Instruction	Original Instruction	"Operation"	New Instruction
0x00858c83	`jbe short loc_858CB9`	modified	`jmp short loc_858CB9`
0x0082912e	`jnz loc_8291D9`	modified	`jmp loc_8291D9`