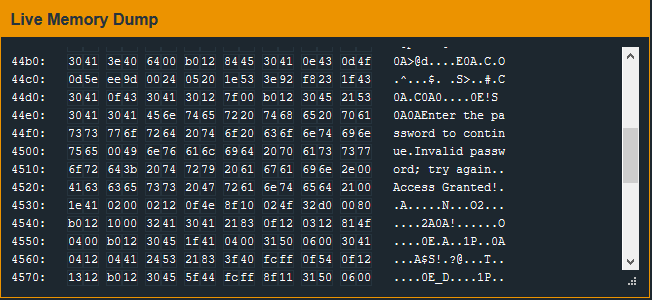
Micro Corruption – New Orleans

When starting this challenge, the first thing I did was read the Manual that pops up for the lock that is being exploited. Near the end of the manual is a sentence that immediately caught my eye. “There is no default password on the LockIT Pro---upon receiving the LockIT Pro, a new password must be set by connecting it to the LockIT Pro App and entering a password when prompted...” I knew that this bit of information would be useful, but wasn’t quite sure how I could use it.

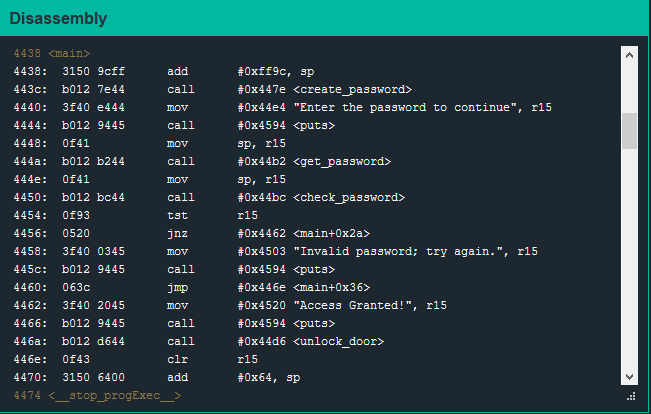
My first question was whether that means each time I attempt to solve the puzzle the password is created, or if it meant that the password had already been created and was most likely stored in memory somewhere. I decided to go with the easiest route and look through the provided memory dump to see if I could find some strings stored in plain text.

When looking through the memory I found some strings that are printed out to the user. In front of the first string “Enter the password to continue”, I noticed an odd combination of characters “0E!S0A0A”.

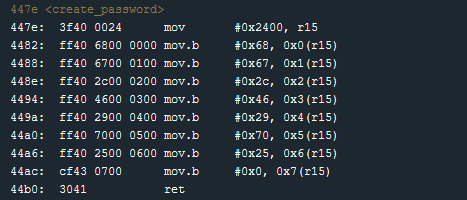


It didn’t explicitly look like a password combo, but I figured I shouldn’t expect to find something as simple as “password” being used. I decided I’d try this string and see if it would grant me access.

Of course the challenge wouldn’t be THAT easy and my provided password was rejected. With that in mind, I decided to review the provided object dump and look inside the main function.



Right away I notice a call to a function “create\_password” that I know I should go investigate.



Well that looks easy enough. It seems that r15 is being assigned to a memory address and the password is then inserted into memory one character at a time. Instead of taking each of these values and putting them into a Hex to String converter, I decided to place a breakpoint at the end of the function so I can see what the password looks like in memory after it has been created.

I run through the program and enter a nonsense password and wait to reach my breakpoint. Once there, I investigate address 2400 in memory and sure enough, there are the bytes we just put into memory. Micro Corruption’s memory dump is kind enough to give us the string representation of the bytes so the stored password is pretty easy to decipher.



I like to try and keep my work to a minimum, so before investigating the code any further I want to try this password “hg,F)p%” as the solution.

I remove my breakpoint and run through it again in debug mode. When prompted, I enter the password we found and… success! I then use the same password to solve the challenge and I’m greeted with a message that our operatives have gained access to the building.