Siyu Zhang

siz24

Project3: Password-cracking

siz24_1:

Password: qXJhErqLalkOJPKtQWyytymEbj

Procedure: Firstly, I used "gdb" to disassemble the main and I found several C functions and conditional jumps, including fgets, chomp, printf, puts, cmps, and jne. So, it's obvious that this program gets the string that the user enters by calling fgets, gets rid of the '\n' by calling chomp, and then compares the user's input with the password string; lastly, prints out the information about right or wrong accordingly. However, for the first several attempts, I used "print $register" to find out the password but failed. Therefore, I used "x/s", following with the memory address of the password string which is copied into the esi register for cmps, to get the password string and successfully unlock the program.

siz24_2:

Password: a palindrome (number or string) with the length greater than 5 (e.g. 123321, siz2442zis)

Procedure: By disassembling the main function in "gdb," I saw several function calls to c, s, and p. Hence, I disassembled each of them as well. In function c, it calls function s three times. Then I looked at function s and figured out this is used for getting the length of the string that is passed in (by finding '\0' and returning the length by eax (increasing by eax 1 if it doesn't hit '\0')). After this, I went back to function s again to figure out its meaning. In fact, function c is a "void" function which checks the conditions of the string. If the string is 0, or the length of the string is 0, or it is not ended with "\n" (0xa), the function will do nothing and return. Otherwise, it replaces '\n' at the end of the string with '\0.' And when it comes to function p. I spent a lot of time searching and learning about assembly instructions and figured it out the meaning of function p. Generally, it used two pointers pointing to the begin (ebx) and the end of the string (eax) (by calling function s to get the length). And when I saw instructions "sub edi, 0x1" and "add ebx, 0x1," I knew that those were two "moving" pointers for comparing two chars in the string from two sides (begin and

end) and moving towards the middle part. Also, function p returns 0 once the two chars it compares are not the same and returns 1 if they are the same. Therefore, I guess the password might be a palindrome. So, I tried “abcba” first but failed. Later, I realized that the program calls function s again for checking the length of the string: the length must be greater than 5. By trying a palindrome with its length larger than 5, I got the “password.”

siz24_3:

Password: a 16-character length string which contains exactly 8 of the following required chars: ‘9’, ‘c’, ‘0’, ‘4’, ‘s.’ (e.g. 99aac87c0421s660, ss4c9099AAAAAAAA) (case-sensitive and can display or reprint the password in a 16-character string form only, including ‘\n’)

Procedure: As what I did before, I started with disassembling the main function. However, this program doesn’t have a main function. Then, I tried “objdump -d -Mintel” to find the “.text” section which starts at the address 0x80480370. By making breakpoint at that address, I disassembled the entire “.text” section by using “disas $pc, +100.” After tracing the code, I realized that this program is not executed from the lower address to higher address. Instead, it runs by calling many functions and it really starts asking for user’s input at 0x08048424. In this function, there are two counters and two loops. The first loop asks for 16-character input from the user (including ‘\n’ and 16 “Enter” works!) and if the requirement is not met, it will keep asking for it. That’s why when I tried to enter 5 characters and pressed “Enter” but there was no correctness information printed on the console. The second loop reuses the counter and uses the other counter for counting the required chars. I saw many conditional jumps in the disassembled code and found that one of the counters is increased by 1 if some conditions are met. So, I used the ASCII table to find out the required characters: ‘9’, ‘c’, ‘0’, ‘4’, ‘s.’ After this loop, the counter for counting the required chars is checked whether it is exactly 8. If it is, this program will be unlocked. Otherwise, it failed. Therefore, the password must contain exactly any 8 of the 5 required chars and the rest 8 chars do not matter.