Project 2 Write-up

Alex Vallejo

amv49@pitt.edu

Program 1

Procedure

Initially when I ran the program in gdb I placed a breakpoint in main. Because the program is doing string comparisons, at some point there must be some type of comparison or branch instruction that makes a decision with regard to the strings. I noticed that there was a ‘cmp’ instruction and inspected what was being compared. Initially%al and %dl were set to 1 or 0. These values are clearly not strings which means the strings are not being compared directly which makes sense according to what we know about string comparisons from java. I then looked to where %al and %dl were being set. I saw there was a move from a memory address to a register shortly before the compare. I set a breakpoint after this move and examined the register which gave me “LqqrELxvhvLPluhXvXHqBT”.

Solution

Lo and behold when I type this string in as a pass phrase the program tells me I have found the correct pass phrase.

Post-mortem/Notes

- The program was most likely statically linked because of it’s large file size and because running mystrings prints a large number of library function names that are probably inside the symbol table or used directly in some manner.
- The value of the pass phrase was probably hard-coded into the program because it is detectable via the mystrings program. It is also probably present in the .data segment of the application.

Program 2

Procedure

Initially I placed a breakpoint in main and disassembled the code around this area. I found that several functions were being called that related to the time such as: time, strftime and localtime. I then examined the values that were being moved after the calls to the time functions to find that “Wednesday” and “October” were being moved into the registers. After the calls to the time functions I discovered that the program was concatenating the first letters of the day and month and using those as the passphrase. I saw that there was a move onto the stack of %ebx before the strformat function so I decided to inspect %ebx after the call to strfformat. I found “Wednesday” in the register. Likewise for the second call I found “October”. I noticed that the register containing the value to compare held “WO” which I then entered as the passcode to find was it was correct.

Solution

One must enter the first letter of the current day of the week and the first letter of the day of the month. On a Wednesday in December, one would have to enter WD to unlock the program.

Notes

- This program is probably statically linked because of the large file size and the large output from the mystrings program.

Program 3

Procedure

I tried to place a breakpoint at main as I did in the previous two programs however the symbol table was not included in the executable this time and gdb could not find a main function. When I used the “info functions” command I saw that getchar was being called and I assumed that was being used to read the user input from the buffer. I placed a breakpoint on getchar and did a back trace to see the locations of the calling functions. I then disassembled each function in the backtrace with a offset of +200 to see what was around each function. I found what was the main by identifying a call to printf which prints the result of either unlocking or not unlocking the program. Next, I began running through the x86 instructions to see what was going on with the input. Each input was subtracted by 98 and was decided if it is <= 24. There was also a second condition that used a shift and bit mask against a certain constant value of 22,026,243. These conditions only allow letters that are on the bottom row of the (qwerty) keyboard through the conditional. If a letter is ‘valid’ a counter is incremented. The program then checks the counter via cmpl to see if it is equal to 7.

Solution

Any combination of the bottom row of the keyboard that is 7 letters long will work as a passcode. For example, bbbbbbb, zxcvbnm, mnmnmnc and zxcvzxc all work as valid passcodes for the program.

Notes

- Because of the small file size and the small output from the mystrings program, I can tell that this program relies on standard libraries to be present on the machine that runs it.
- Moreover, the presence of the @plt functions (which is a stub that calls the real functions) is a direct indication that this program uses dynamic loading.