Project 2 Write-up

John Abraham
Jpa18@pitt.edu
CS 449: 4pm
Part2: Passwords

FILE: jpa18_1
PASSPHRASE: "XBSzukHXyeTBNOGOUtcO"
PRODECURE & SOLUTION: For this executable using (file jpa18_1) I saw that the executable is a non-stripped binary so I used GDB to debug the program and examine its data because the symbol table is there. After analyzing the disassembler dump and with some trial and error I found the passphrase to be "XBSzukHXyeTBNOGOUtcO"
Successful Attempt: looking the disassembler dump I placed a break point at address 0x0804830c because the instruction (repz cmpsb %es:(%edi),%ds:(%esi)) seemed promising because the instruction cmsb compares bytes in memory and right after this instruction there's a comparison between $al and $dl at address 0x08048314 then a jump if not equal (jne) at address 0x08048316 after that. When reaching the break point I used x/s to eXamine the content of $esi and $edi and found that $edi contained the string I entered "something" and $esi contained a long string "XBSzukHXyeTBNOGOUtcO." I restarted the program and entered the long string in and it worked.
Failed Attempts: looking at the assembly code there were some other parts where I thought the passphrase might be stores. Such as address 0x080482fd where <chomp> is called; however, at the break point $eax and $edi contained the string I entered, and $esi contained meaningless information in terms of passphrases. Another failed attempt is when I set breakpoint at address 0x08048327 where printf is called. I placed a break point there and wanted to examine the content of registers at that break point but the program was completed before reaching the break point, so this means that the printf was trying to print "congratulations" and I entered a wrong passphrase so that part of the program was never reached.

____

FILE: jpa18_2
PASSPHRASE: "WO"
PRODECURE & SOLUTION: For this executable I was able to find the passphrase after a few failed attempts with some trial and error. I only used the GDB debugger because this executable is non-stripped binary.
Failed Attempt 1: I disassembled the program and created a break point at the only "printf" function call at address 0x080483b2 (b *0x080483b2). The program never paused at the break point. So I assumed that was the printed string for "congratulations" and I guessed incorrectly so it was never printed and that part of the program was never reached.
Failed Attempt 2: Then I put a break point at address 0x0804831b because something was moved to $edi (the string destination register) from the stack pointer mov %edi, 0x8(%esp). I examined some registers. $edi and edx contained empty strings "", and $ebx contained the string "Wednesday" which turned out to be incorrect.
Successful Attempt: Then I placed a break point at address 0x0804838e where <chomp> is called. At this break point I eXamined the contents of $eax and saw that it contained the string I just entered, "something". Then I used "info reg" to see the registers being used. After eXamining some registers I discovered that register $edi (string destination register) contained the string "WO". I restarted the program and tried this string and it worked.

____

FILE: jpa18_3
PASSPHRASE: “c c c c c”
PRODECURE & SOLUTION: For this executable I could not use the GDB debugger the same way I did for the previous two executable because using the command “file jpa18_3” I saw that this executable is a stripped binary; therefore it had no symbols table. So I still used GDB but it was more difficult; however, I was still able to find the passphrase with the help of objdump.
So for a first attempt I used the commands “strings jpa18_3,” and “mystrings jpa18_3” to see the plain text strings in the binary, and tried all the strings as passphrases and they all failed.
For another attempt I used “objdump –d .text jpa18_3”. This showed me the assembler dump of the .text section. And I looked like at address 0x080484c2 there is a test instruction and at address 0x080484cf there’s a cmp instruction. So I placed a break point at 0x080484c2 looked at the register contents. The content of $esp was letter ‘c’. Then I used “stepi” to step through the next 5 instructions to get to address 0x080484cf where the comparison occurs. And for the next 4 steps $esp had ‘c’ in it. So I exited the program and ran ./jpa18_3 and entered “ c c c c c” and the program unlocked.