Craig Kodman

[Cmk126@pittt.edu](mailto:Cmk126@pittt.edu)

2/28/2016

Password 1:

As in the puzzle lab we did earlier in the semester, I loaded my program into gdb and looked for compare functions as a starting point. I put a break point at main. From there I disassembled and looked for interesting spots to put a break point. One break point was for a cmpsb which was comparing $edi and $esi. When that break point came up I entered “x/s $esi” because when I displayed edi, esi, edx, ecx, ebx and eax, it was the only one with an interesting high number which I came to learn shows something is in there. When I looked at esi I got “RaZpnqLdDZNlhBqHKbXzZEydGuf” which looked like it could be a password. I then entered that password when I ran the program, and it was the password.

It could be also found using strings cmk126\_1. When doing it that way I found:

9\_<v0

[^\_]

[^\_]

[^\_]

\[^\_]

[^\_]

ffffff.

RaZpnqLdDZNlhBqHKbXzZEydGuf

Sorry! Not correct!

Congratulations!

Unlocked with passphrase %s

FATAL: kernel too old

FATAL: cannot determine kernel version

/dev/full

Above I also found that odd looking string that looked like a possible password. As it was hardcoded here, I could have also found the password this way.

Part B

As I did to find the first password, I ran the program and put a break point in main. From there I disassembled the code and place breakpoints at many spots of interest. The breakpoint I first made after main was at repnz scas. When that breakpoint in main was reached I checked edi and found exactly what I had typed when prompted (“lkjsdf”). So this is where my entry is stored. I then checked many other breakpoints until I found “\_2” and “cmk126\_2”. I tried “cmk126\_2” as my password and it failed. I then found that $eax in 0x080485cd has “\_2” in it and that “\_2” was being used to append input. I then checked $esi and found “/cmk126\_2” being stored. So when I entered the program name, it stored this. This might be stored to build the password, so I took note of that. I later found that “cmk126” was also stored by itself. This led me to deduce that the “\_2” was being appended to my guessed password and was being compared to “cmk126\_2”. So if I entered “cmk126” the program would append “\_2” and it would then make an equal comparison with “cmk126\_2” and it would work. And it did! So the second password is “cmk126” since that is my username and is used to make the name of the file – cmk126\_2. So if someone else was using this program and their file name was “dlb12\_2” then their password would be “dlb12”.

Part C

There is no main() in this program so I had to do an object dump to view all of the assembly code. I printed this out so I could look at all of the code. After many hours of tracing I realized that this program only cared about 10 characters for input. If I was short of that 10, then it still waited for more input. I also disassembled .text to search through that. In there, I made a ton of breakpoints around nearly anything that looked interesting, especially any cmpb or cmp operations. I also saw that there was a lowercase function that would make this case insensitive.

I ran the program and at each breakpoint I entered “display $ebx” or “display $eax” etc to see if there were any high numbers in there. I found that when there were low numbers, it was out of bounds or not interesting. If there were high numbers, then I would look directly at the register with x/s to see what I found.

At one point, “0x8049718:” there was “\r” which I thought might have something to do with my password. I tried all kinds of combinations of “\r” and any other 10 characters but could never find the password. I even tried to reverse them with “r\” or “/r” and nothing would work. See picture below.

