Atta Ebrahimi

Program 1:

To begin looking for a password I set a break point at main and then ran the program until it hit main. At 0x080482f5 I saw a call to fgets which was most likely for user input. Two lines after a function called chomp is called. I assume that this modified the input in some way. I put a breakpoint at 0x080482fd which is where chomp is called. I entered “hello” when it asked for input and then the program stopped at the break point. After the call to fgets there’s a call to move something into $edi. I determined what was in $edi using x/s which happened to be “hello/n.”

I put a breakpoint at -x0804830c to see what happens after chomp and the string in $edi was modified to “hello”, so the escape sequence is gone. $esi is a register that was pushed on to the stack earlier so I checked what was in it and it contained “ldwldeisAQOUtrOljlOHhWB”. This register was later compared to $edi, so I assumed that this was the password. I reran the program and this worked.

Password: ldwldeisAQOUtrOljlOHhWB

Program 2:

Using break points, and the disas command I was able to get the assembly and piece out what the program is doing. I put a break at main and noticed called to strcmp, fgets, strcpy, as well as two other functions <r> and <c>. I put breakpoints at all of these functions and slowly worked my way through the problem. I noticed that after the fgets and entering abc call and before strcmp $edi and $esi contain abc and ate9\_2 respectively. I then reran the program and checked the same registers. This time around $edi and $esi contain ate9\_2, and /ate9\_2 respectively continuing to a breakpoint inserted after the comparison I check three registers $edi, $esi, and $ebx which contained “2”, “ate9\_2” and $ebx contained ate9\_2\_2. From this I took that the <c> function appended \_2 to my input and stored it in ebx. I reran the program entering just “ate9” and the program opened up.

Password: my username “ate9”

Program 3:

First thing I noticed is that this executable does not have a main function. I used objdump -d ate9 to view the assembly which I found to be very hard to trace. I set a break point in it and used the “ni” command in gdb to see how the program runs. At 0x0804846b to 0x0804847f there’s a loop that sometimes gets a string input from the user. It appears to add a number to the character in question each time. I believe the program will work for multiple passwords that follow a similar pattern. I started checking phrases that have a certain ascending and descending order. It turns out that the program can only take up to 10 characters including return character. The phrase seems to have to be at least 5 characters long and no longer than 10. If you input something longer than 10 characters, it will truncate them. Unfortunately, I couldn’t match a password to the program to get it to open up.