AUTH

By simple execution on local machine let us examine “auth”.

As we execute it we get a message saying failed to open file. On checking the Assembly we know it is trying to read a file named “flag.txt”. Let us create a fake file “flag.txt”, with some content, on local machine and move forward. We are going to create this file in the same directory

Executing it again we do not get this error this time. We asked for the flag instead. On checking the Assembly we know it will accept only 39 characters at max. As can also be observed by giving large inputs. Evidently it can also be seen that only 39 characters are read from the file too. Later in the program over input is compared to the data read from file <0x80487eb>.

Since the comparison fails every time we're told that the flag is incorrect. Though, we can see that it returns our input too. This could be vulnerable.

How input is a controlled buffer. now checking for string format vulnerability let’s give input as “%x”. The system returns our name as an address . This is vulnerable

Now all we have to do is find the position in stack relative to this which contains our flag. Since it was read and compared it must be saved somewhere in our stack. we will be using GDB for this. I found it on ‘0xffffcecc’.

We are exploiting stack present during the “puts” function at <0x8048859>. First element leaked in my case is located at ‘0xffffce9c’. Does to calculate the position of our flag we need to find the difference between these two addresses which is nothing but [0xffffcecc-0xfffce9c]. So this will be our offset x 4 , which is 52 in decimal. Therefore our offset would be 52/4=13, because each memory block in a stack is made of 4 bytes. Thus formatting our input as “%13$x” will give us first 4 bytes of our flag stored in the stack.

The program returns us hex value as “67616c66”, which in string is “galf”. This is reverse of “flag”. Reverse due to ‘Little Endian’ storing preference used in our system. <[Endianness](https://en.wikipedia.org/wiki/Endianness)>

In similar fashion we can obtain the whole flag. Reversing every four bytes for proper order