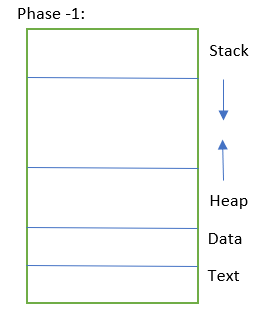
Attack lab

Attack lab is a gist about finding the vulnerabilities of the system, and attack it such that the system fails causing wrong calls to functions or giving wrong output. We used hex2raw.exe and ctarget to create an object dump file. We then used ctarget to find the address of the get\_buff() and touch1() to understand the assembly language commands.



According to the address from the assembly language 40 bytes are allocated for stack. The next 8 bytes are allocated for the return address. To cause the attack we are finding the vulnerability that, if we send 40 bytes we fill the stack, but if we send more that 4 bytes, it causes a segmentation error. By sending the address of the function we want to execute in place of the extra 8 bytes, causes to replace the return address with the wrong address.

For this we are creating a text file sending the 40 bytes data along with the next 8 bytes address.

00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00

C0 17 40 00

We then execute the command with hex2raw. The address we give matches the address in the stack and it assumes that the address is correct causing it to get attacked.

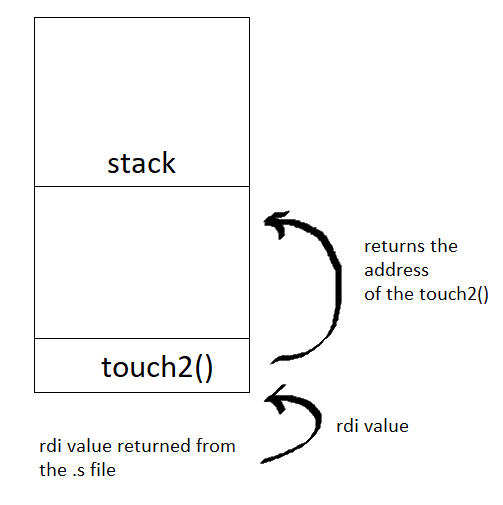
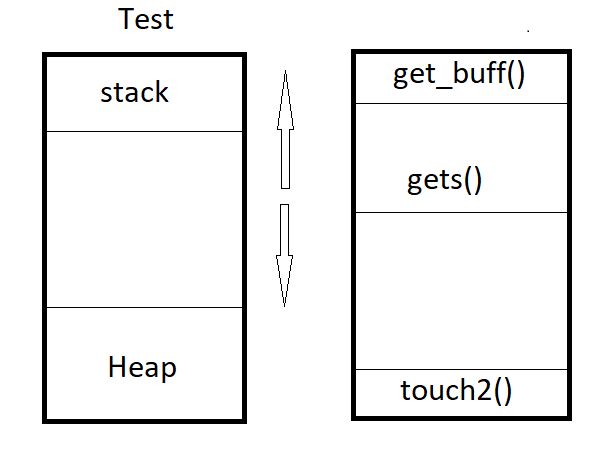
./hex2raw < output.txt | ./ctarget -q

This calls touch1().

Phase-2:

Attacking the system with the vulnerability by calling another function using a parameter. We use a .s file with an address returning the address stored in rdi register. The cookie value is required to be passed as a parameter to the touch2() for execution. This returns a hexadecimal value for the .s file.

The vulnerability is in gets() as the cookie value required sending wrong to cause wrong function call. .s file containing the assembly code creates a .O file that is executable file.



Here also we create a target.txt file containing the values of 40 bytes with 16 bytes extra. The extra 16 bytes contains the return address for the .s file and then calling the touch2() with the returned value as argument.

48 c7 c7 fa 97 b9 59 c3

00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00

78 dc 61 55 00 00 00 00

ec 17 40 00 00 00 00 00

We then execute this file with hex2raw.

./hex2raw < target.txt | ./ctarget -q

This calls touch2() with cookie value: 0x59b997fa.