**CS6008 Cryptography and Network Security.**

**Assignment No.2**

**Name:** G Vignesh **Reg No:**2018103078

**Date:** 9.4.2022

**Module-2:** Memory Corruption

**External Learning-** Implementing simple format string attacks.

**Aim:**

To construct a c program which implements has format string attacks and expose its vulnerabilities and use that vulnerability to see value of value of variables stored in stack and change the value of a variable stored in an arbitrary location in stack . By changing the values to which are not given access to user we may thereby executing some other method instead of normal execution.

**Tools-used:**

1.Linux Terminal

2.GCC compiler.

3.GNU Debugger.

4.Hexadecimal to ascii converter.

**Description:**

A Format String attack can occur when a user input string data is processed by a vulnerable function so that attacker can pass the formats to exploit the stack values with the help of format string functions/printf() family functions. By Format String vulnerability, an attacker can execute code, read the stack values, or cause a segmentation fault in the application. By passing format specifiers such as

1. %c — Formats a single character
2. %d — Formats an integer in decimal value
3. %f — Formats float in decimal value
4. %p — Formats a pointer to address location
5. %s — Formats a string
6. %x — Formats a hexadecimal value
7. %n — Number of bytes written

As a part of our input we can read values from the stack, since printf looks for such format specifier if not present it prints values matching to that specifier from the stack.

**Input:** Format specifier (%s, %n, %p, %x) as input to printf function.

**Output:** Arbitrary values from stack or changing the values of local variable thereby changing the flow of the program.

**C-program:**

**Format.c**

#include<stdio.h>

#include<string.h>

void vulnfn(){

        printf("\nhello hacker");

}

int main(int argc,char\*\* argv)

{

    char buff[5]="ABCDE";

    int pass=0;

    printf(argv[1],(int \*)&pass);

    if(pass)

    {

        vulnfn();

    }

    printf("\n%d\n",pass);

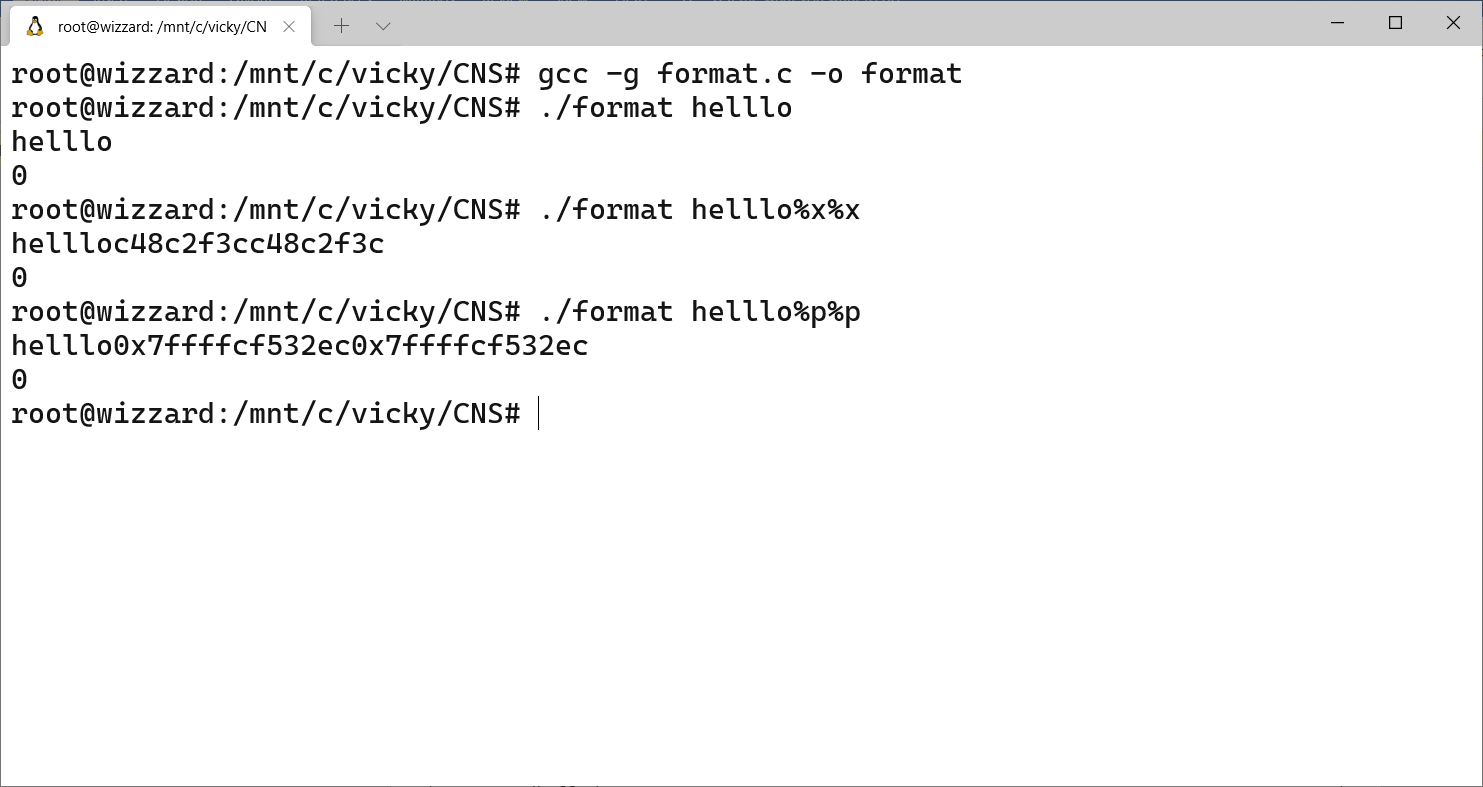
    return 0;

}

**Execution:**

**Step-1:**

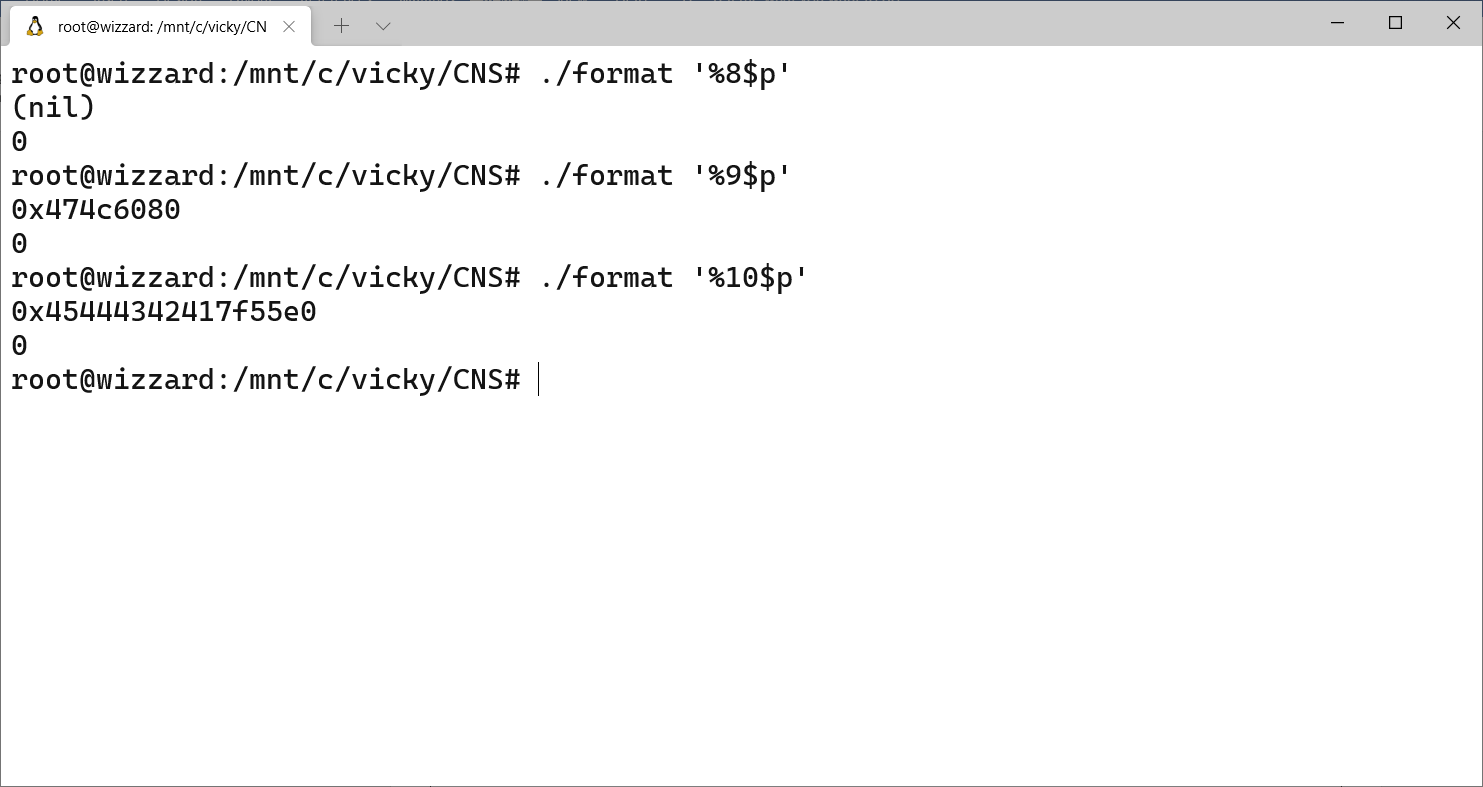
In the above program the we can see that the character array[] buff is no where printed in the function, Let’s say it may be a secret value such as password/ Important details. In this program user input argv[1] command line argument is part of format vulnerable function . on normal execution it just prints what we give as input.



But when we specify some format specifiers such as %x or %p it prints values from the stack. We have got access to the stack values due to the format string vulnerability.

**Step-2:**

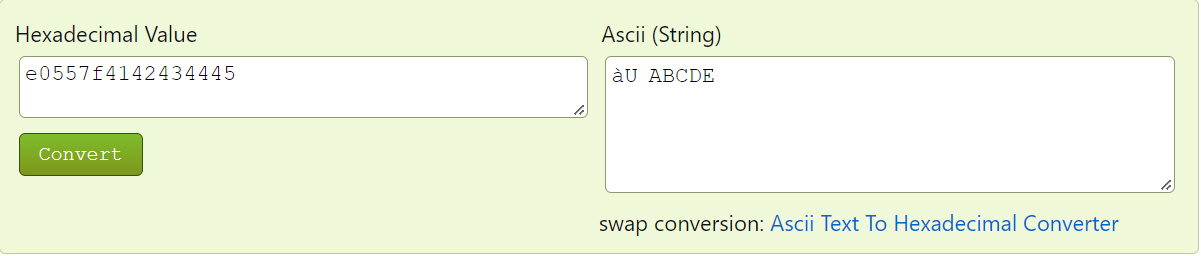
We can exploit this vulnerability to print values from the stack that are needed for us such as local variables in this character buff[] which has some important details can be accessed with out any permission by just printing values from the stack.



If we know the structure of the stack and where the local variable will be present in the stack from the base stack pointer we can easily access the location using the format specifier in printf().

Here local variable is printed in hexadecimal value:

0x45444342417f55e0 🡪 which will reverse so e0557f4142434445

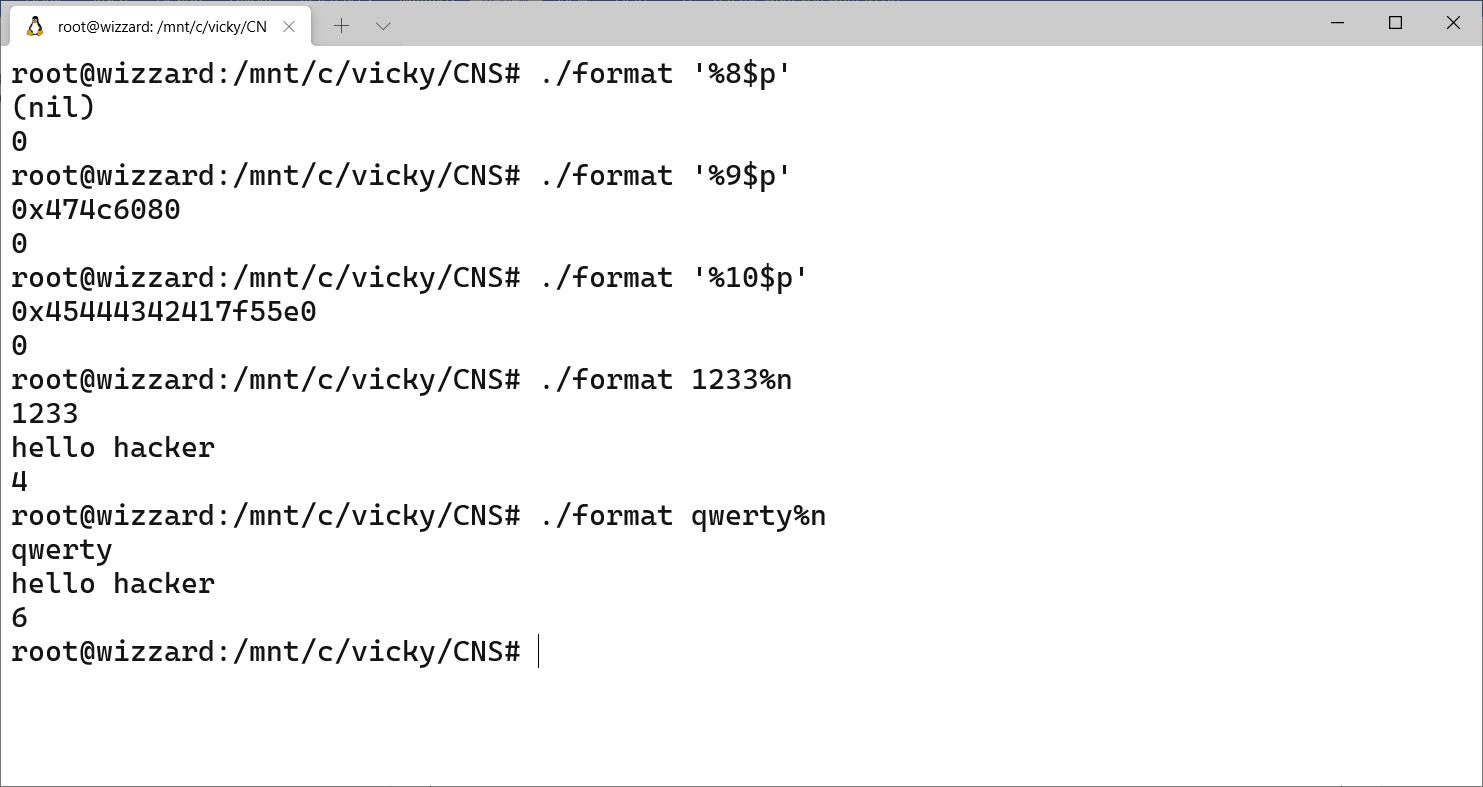


The value aU is junk and the value is ABCDE which is the local variable which may be secret by using format string vulnerability we got access to it. This is known as **Memory Leakage**.

**Step-3:**

Not just getting values from stack we can even change of the local variable and change the flow of execution using %n format specifier. Which is used to write number of bytes written so for into the argument given in printf().

In the program the variable pass is initially set to 0. But using format string attack we are going to change its values. Due to which some other function is called which might not happen in normal situations.



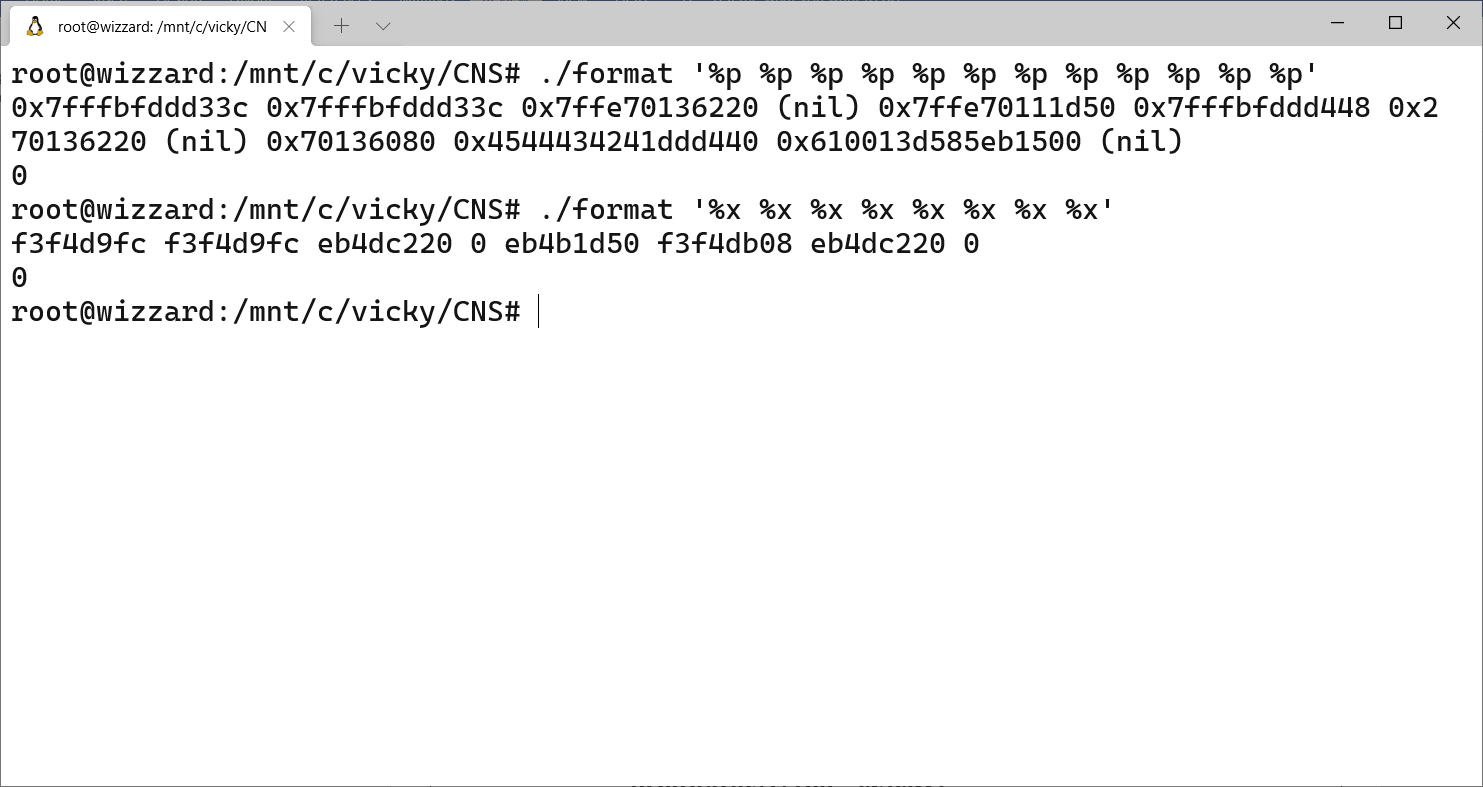
%n writes number of character before in into the pass variable. Due to which the value of pass is changed and vulnerable function vulnfn is executed, and hello hacker is printed. Which execute only if pass variable value is greater than 0.

**printf(argv[1],(int \*)&pass);**

The value pass which changes according to the number of character is printed in last line.

**Step-4:**

Using this vulnerability we can print values from the stack, such as ebp, return address, local variables etc. In this way, the attacker could execute code, read the stack, or cause a segmentation fault in the running application, causing new behaviours that could compromise the security or the stability of the system.



**Conclusion:**

Format string vulnerabilities can be over come by following some mitigations. We must always specify the format string %s/%d etc as a part of program not as a part of user input **printf(%s, argv[1]) .** Secondly use input validation always which is one way of secure coding so that we may avoid input from user if it contains unnecessary characters such as format strings. Use format guards. And not keep any important information as a part of the code unless encrypted such as password because the values in stack can easily be accessed using several vulnerabilities.