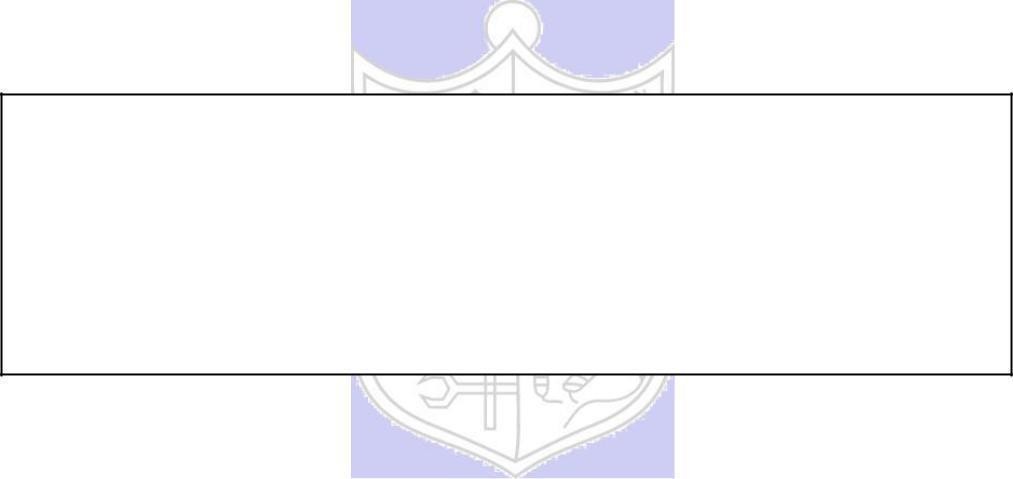
BKJSCE/IT/TY/SEM V/INS/2020-21



**Experiment No. 6**

**Title: Simulation of Buffer Overflow Attack**

**Batch:B1` Roll No.:1814078 Experiment No.:6**

**Aim:** To simulate Buffer Overflow Attack.

**Resources needed:** Windows/ Kali Linux OS, C/JAVA language

## Theory:

A buffer, in terms of a program in execution, can be thought of as a region of computer’s main memory that has certain boundaries in context with the program variable that references this memory.

For example:

char buff[10]

In the above example, „buff‟ represents an array of 10 bytes where buff[0] is the left boundary and buff[9] is the right boundary of the buffer.

A buffer is said to be overflown when the data (meant to be written into memory buffer) gets written past the left or the right boundary of the buffer. This way the data gets written to a portion of memory which does not belong to the program variable that references the buffer.

Here is an example:

char buff[10]; buff[10] = 'a';

In the above example, we declared an array of size 10 bytes. Please note that index 0 to index 9 can used to refer these 10 bytes of buffer. But, in the next line, we index 10

was used to store the value „a‟. This is the point where buffer overrun happens

Buffer overflows, if undetected, can cause your program to crash or produce unexpected results. int main(){

char buff[10] = {0};

strcpy(buff, "This String Will Overflow the Buffer");}

As you can see that the program will write the complete string in the array „buff‟ but as the size of „buff‟ is less than the size of string so the data will get written past the right boundary of array „buff‟. Now, depending on the compiler you are using, chances are high that this will get unnoticed during compilation and would not crash during execution. The simple reason being that stack memory belongs to program so any buffer overflow in this memory could get unnoticed.

So in these kinds of scenarios, buffer over flow quietly corrupts the neighbouring memory and if the corrupted memory is being used by the program then it can cause unexpected results.

Consider for example the software is used for authentication and the authentication resides in a single bit in the memory. If a buffer overflow overwrites this authentication bit, then any attacker can get authenticated.

Buffer overflow errors are characterized by the overwriting of memory fragments of the process, which should have never been modified intentionally or unintentionally. Overwriting values of the IP (Instruction Pointer), BP (Base Pointer) and other registers causes exceptions, segmentation faults, and other errors to occur. Usually these errors end execution of the application in an unexpected way. Buffer overflow errors occur when we operate on buffers of char type.

## Activity:

1. Students should find some vulnerability in c programming.
2. Students should use the vulnerability to implement buffer overflow on Kali Linux. Hint: Explore Stack smash

**Results:** (Program printout with output / Document printout as per the format)

**1.** Finding some vulnerability in c programming:

#include <stdio.h>

#include <string.h>

void main(){

char a[10];

char pass[] = "abcdxyzw";

printf("Enter The Password: ");

gets(a);

if (strcmp(a, pass)==0){

    printf("Correct Password");

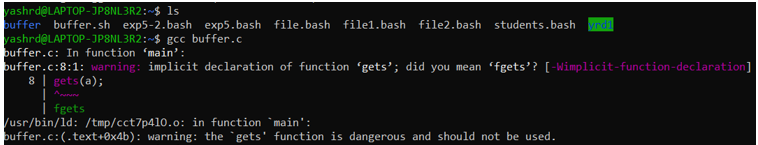
}

else{

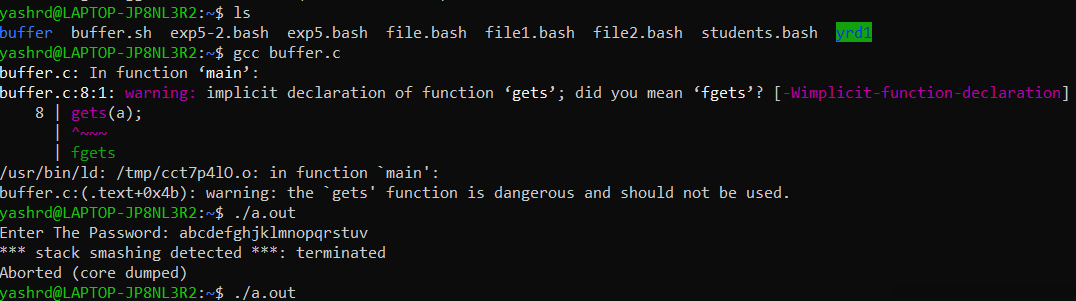
    printf("Incorrect Password");

}

}



**2.** Using the vulnerability to implement buffer overflow on Kali Linux.



# Questions:

1. **Explain buffer overflow in terms of web server attack?**

**Ans:**

A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold or when a program attempts to put data in a memory area past a buffer. In this case, a buffer is a sequential section of memory allocated to contain anything from a character string to an array of integers. Writing outside the bounds of a block of allocated memory can corrupt data, crash the program, or cause the execution of malicious code.

1. **What are the other nonmalicous program errors or attacks?**

**Ans:**

**Time-of-Check to Time-of-Use Errors:**

Access control is a fundamental part of computer security; we want to make sure that only those who should access an object are allowed that access. Every requested access must be governed by an access policy stating who is allowed access to what; then the request must be mediated by an access policy enforcement agent. But an incomplete mediation problem occurs when access is not checked universally. The time-of-check to time-of-use(TOCTTOU) flaw concerns mediation that is performed with a "bait and switch" in the middle. It is also known as a serialization or synchronization flaw.

**Incomplete Mediation**:

It occurs when the app accepts incorrect data from user. need to know that any user input falls withing specified values. Client-‐Side Mediaeon. many websites rely on the client to keep state for them

# Outcomes: Understand Security issues related to Software, Web and Networks.

**Conclusion:** We studied and implemented Buffer overflow in this experiment.

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of faculty in-charge with date References: Books/ Journals/ Websites:**

1. Charles P. Pfleeger, “Security in Computing”, Pearson Education
2. BehrouzA. Forouzan, “Cryptography and Network Security”, Tata McGraw Hill
3. William Stalling, “Cryptography and Network Security”, Prentice Hall