Buffer Overflow Illustration using GCC/GDB/Linux Debugging Tools

AIM:

The main aim is to write a program to illustrate buffer overflow attack.

PROCEDURE:

A buffer overflow (or buffer overrun) occurs when the volume of data exceeds the storage capacity of the memory buffer. If the transaction overwrites executable code, it can cause the program to behave unpredictably and generate incorrect results, memory access errors, or crashes.

```
#include <stdio.h>

#include <string.h>

int main(void)

{

char buff[15];

int pass = 0;

printf("\n Enter the password : \n");

gets(buff);

if(strcmp(buff, "thegeekstuff"))

{
                                                  printf ("\n Wrong Password \n");

}

else

{
                                                  p

                                                  r

                                                  i
}
                                                  n
if(pass)
                                                  t
{
                                                  f

                                                  (

                                                  "

                                                  \
n
C
o
r
r
e
c
t
P
a
s
s
w
o
r
d
\
n
"
)
;
p
a
s
s
=
1
;


/* Now Give root or admin rights to user*/
```

```
        printf ("\n Root privileges given to the user \n");
}
return 0;
}
```

The program above simulates scenario where a program expects a password from user and if the password is correct then it grants root privileges to the user.

Let‟s the run the program with correct password ie „thegeekstuff‟ :

OUTPUT

```
RUN1

Enter the password :

thegeekstuff
Correct
Password
Root privileges given to the user
```

This works as expected. The passwords match and root privileges are given. But do you know that there is a possibility of buffer overflow in this program. The gets() function does not check the array bounds and can even write string of length greater than the size of the buffer to which the string is written. Now, can you even imagine what can an attacker do with this kind of a loophole?

Here is an example :

```
RUN 2

Enter the password :

hhhhhhhhhhhhhhhhhhhh
Wrong Password
Root privileges given to the user
```

RESULT:

The main aim is to write a program to illustrate buffer overflow attack is completed successfully

Commands to execute: In bash

1. sudo apt update
2. sudo apt install –y build-essential
3. nano overflow.c (type the program and exit nano editor using Ctrl+o, enter, Ctrl+x
4. gcc -g -O0 -fno-stack-protector –no-pie overflow.c –o overflow (Debug the program with no optimization, disabled stack canary protector and disabled position Independent Executable
5. ./overflow (To check output)