****

**EE 4550L**

**IC Hardware Security and Trust LAB**

**SPRING 2024**

**TA: Kanchan Vissamsetty**

**Lab section: 01**

**Name: Alex Yeoh**

**“I have neither given nor received aid on this assignment, nor have I observed any violation of the Honor code”**

**Signature: Alex Yeoh Date: 23rd February 2024**

**Report due date: 23rd February 2024**

1. **OBJECTIVE**

To learn how various overflows work in C on a linux machine.

1. **PROCEDURE**

Write a C program that demonstrates buffer, integer, and heap overflows. Perform a simple buffer overflow attack on simple given program. Perform a more complex buffer overflow attack on a more complex given program.

1. **RESULT**

|  |
| --- |
| #include <stdio.h>  #include <limits.h>  heap\_overflow(){  char\* nums = malloc(2\*sizeof(char));  nums[0] = '0';  nums[1] = '1';  int\* outside = &nums[1]+1;  \*outside = '2';  printf("Before %d\n",\*outside);  nums[2] = '3';  printf("After %d\n",\*outside);  }  integer\_overflow(){  int num = INT\_MAX;  printf("Before %d\n", num);  ++num;  printf("After %d\n", num);  }  stack\_overflow(){  char nums[2] = {'0','1'};  int\* outside = &nums[1]+1;  \*outside = '2';  printf("Before %d\n",\*outside);  nums[2] = '3';  printf("After %d\n",\*outside);  }  int main(int argc, char const \*argv[])  {  printf("Heap Overflow\n");  heap\_overflow();  printf("\nInteger Overflow\n");  integer\_overflow();  printf("\nStack Overflow\n");  stack\_overflow();  } |

C code for overflows

A screen shot of a computer

Description automatically generated

Output for code above.

This could be prevented by putting in checks that prevent these attacks such as first checking the length of the input before storing the information to prevent a buffer overflow attack.

A screenshot of a computer program

Description automatically generated

Output when the correct key was given.

A screen shot of a computer code

Description automatically generated

Output when an incorrect key was given.

A screen shot of a computer

Description automatically generated

Output when an incorrect key that is too long was given.

Conclusion: The program is weak to buffer overflow attacks and because xyz is an int, if anything other than a 0 was overflowed into it, it would resolve to a true in the if(xyz) conditional check.

It is weak to buffer overflow attacks because there are no protections to prevent it.

This specific problem can be fixed by removing the if(xyz) check and instead wrapping together granting access and the password check.

The endianness of the address inside the perl script is little endian.

A black screen with white text

Description automatically generated

Image of successful attack (code slightly modified from shutdown to sshutdown to prevent it)

A black background with white text

Description automatically generated

Image of code that caused successful attack.

Conclusion: The program is still vulnerable to buffer overflow attacks but not as weak as the program from part 2.

The program could be protected from this attack by first checking the length of the input and rejecting it if it does not fit in the char data[10] array.

1. **CONCLUSION**

My results satisfy the requirements. I don’t think I can improve my design to get better results. I have learned how various overflows work in C on a linux machine and how buffer overflow attacks being implemented appear.