LAB

6

Introduction to Buffer Overflow

By the end of this section, you should be able to:

- Writing a simple application with Buffer Overflow Vulnerability.
- Exploit Buffer Overflow vulnerability

6.1 Introduction to Buffer Overflow

A buffer overflow occurs when an input to a program or process have a data size more than a fixed length block of memory, or buffer, than the buffer is allocated to hold. Buffers such as array in a program code are created to have a data size defined during the initialization of the buffer, an extra data can overwrite data values in memory addresses adjacent to the buffer destination unless the program code itself have a mechanism to check the input data when too much is sent to a memory buffer. Failing to provide a mechanism to overcome this issue will result in exploitation of the vulnerability.

Exploiting a buffer overflow allows an attacker to control or crash the process or to modify its internal variables. A carefully crafted input to an application can cause the application to execute arbitrary code, possibly taking over the machine. Programming languages like C and C++ are expose to buffer overflow attacks because the programming language does not have built-in protection against accessing or overwriting data in any part of their memory and as actors can perform direct memory manipulation with common programming constructs. Modern programming languages like C#, Java and Perl reduce the chances of coding errors creating buffer overflow vulnerabilities.

6.1.1 A buffer overflow code

1. Write and compile the code below in your Kali OS using the a text editor and gcc compiler

```
#include <stdio.h>

void secretFunction()
{
    printf("Congratulations!\n");
    printf("You have entered in the secret
function!\n");
}

void echo()
{
    char buffer[20];

    printf("Enter some text:\n");
    scanf("%s", buffer);
    printf("You entered: %s\n", buffer);
}
```

```
int main()
{
   echo();

   return 0;
}
```

2. Save the code as `vuln.c` and compile the code as a 32 bit program by using the command below

```
gcc vuln.c -o vuln -fno-stack-protector -m32 -no-pie
```

3. Run the program using `./vuln`
4. Enter any value and you should only see “ You entered: [your input]

6.1.2 Exploiting the BufferOverflow

1. Enter a character more than 20 characters and report what happened to the program.
2. Run the vuln program with the command below and study the assembly code of the program

```
objdump -d vuln
```

3. Locate the buffer size and the address of the buffer starts and the function <secretFunction>.
4. From 3 you can estimate the maximum buffer sizes before the program overflow.

5. To exploit the program so that the program will run the secret function you need the determined the sizes of the buffer and the start address of the <secretFunction>
6. Once the value is determined then use the following command to inject the right amount of character and then run the <secretFunction>

```
python -c 'print "a"*32 + "\x9d\x84\x04\x08"' | ./vuln
```

- **The Instructor will explain the command use above.**

Review Question

Do a research how to prevent from writing a bufferoverflow C program :-

1. What all C functions are vulnerable to Buffer Overflow Exploit?
   a. gets
   b. scanf
   c. sprintf
   d. strcpy