Cryptography and Network Security
System Security : Memory Integrity Attack and Defenses

Km Simran Jaiswal(20CS60R58)

18-Oct-2020

1 Introduction

Network security tell us how to protect computer against network attacks. Attackers ultimate goal is to get access to your machine. System security deals to protect our system from these attacks.

2 Memory Integrity

Property of Computer memory (RAM or cache) remaining unaltered ,except by authorized parties. Our program resides in memory ; hence memory contain information such as code or password.
If attacker can read your memory then security is breached.

2.1 How memory Integrity works ?

- Idea is to take over victim machine. Attacker execute arbitrary code on victim machine by hijacking application control flow.
  Control flow means which instruction will be executed next.
- Instruction are loaded into memory and executed one after the another.
  Attacker's job is to change what should be executed next.

3 Control Hijacking Attacks : Buffer overflow attack

Stack is a Data structure that store local program code and data.
Consider the code below:

```
void foo (char *s){
char buf[10];
```

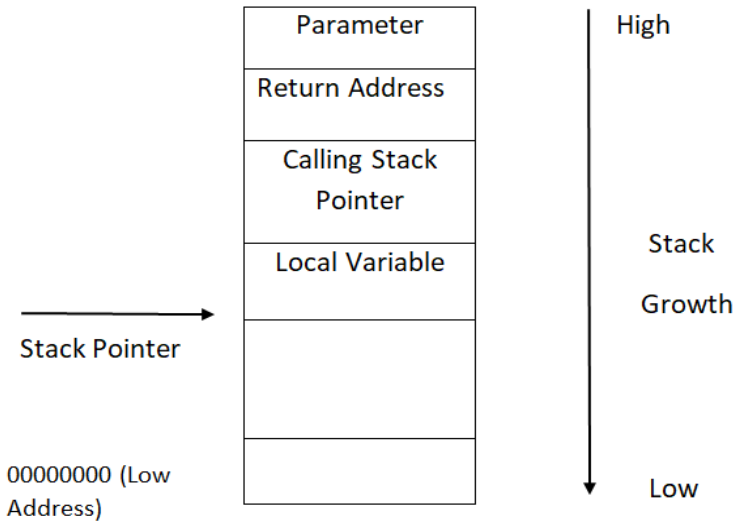
```
strcpy(buf,s);
printf("%s\n",s);
}
.....
.....
foo("thisstringistoolong");
```

The issue in the above code is that strcpy function never check size of copying data. Copying *s* into *buf* require bounds check otherwise it will result into **out of bounds** memory write.

- The general idea is give system a very large string that will overflow buffer. Buffer is a finite array.
- System with sloppy code (where bounds are not checked) are easy target of buffer overflow attack.

3.1 C call stack structure

- When we call a function the return address , values of parameters , stack frame pointer and local variable are stored in stack.
- Stack grows from high address to low address.



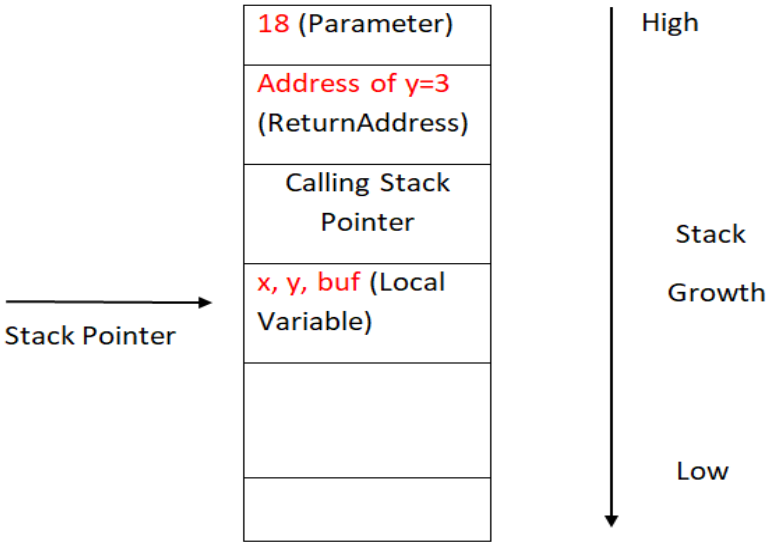
- Consider the code below:

```
x=2;
foo(18);
```

```
y=3;

void foo(int y)
{
int x,int y;
char buf[10];
x=j;
......
......
}
```

Stack structure of above code is :



3.2 General idea of Buffer Overflow

- Buffer overflow means modify return address . When function complete its execution it will jump to modified return address.
  Modified return address will point to buffer which has malicious code.

3.3 Issues In Buffer Overflow Attack

3.3.1 How do we know what value should the new "return address" be?

The new return address is the address of buffer (because buffer is under attacker's control) but how do we **Guess Address** of buffer.

Guessing address :

- We need to know source code of app to estimate both address of buffer and new return address.
  If stack start from 'x' address then we can perform reverse engineering and find address of buffer.
- Simply, we can guess million time address of the buffer.

3.3.2 How do we build "small program" and put it in a string?

Constraints for small program is:

- Small program cannot contain null character '$\backslash 0'$ because once OS encounters '$\backslash 0'$ it stop executing.

Generating String :
The small program into buffer does an exec() system call which changes password database or other file.

- Compile the exec() code and generate machine language code.
- Take this individual byte and build a string.

3.4 NOP's (No Operation Instruction)

- It perform no operation but simply advance instruction pointer.
- Usually we put sequence of NOP instruction at start of our program.
- It solves the problem of finding exact address of buffer overflow . Hence , it is preferable for attacker.

3.5 Unsafe C function

None of the below function is safe because it never check array bounds. These function do not compare size of destination and source . Some unsafe function are:

- strcpy (char *dest , const char *src)
- strcat (char *dest , const char *src)
- gets (char *s)
- scanf (const char *format ,. . .)

4 Conclusion

A buffer overflow attack happens when a program write more data to a fixed length buffer . This attack changes control flow of the program and violates memory integrity . Many famous software such WeChat , Realtek , GNU screen are vulnerable to this attack.