Buffer Overflow Attacks

Buffer Overflow

- **Buffers** - data storage areas that hold a predefined amount of finite data
- **Buffer Overflow** – happens when the data exceeds the size of the buffer

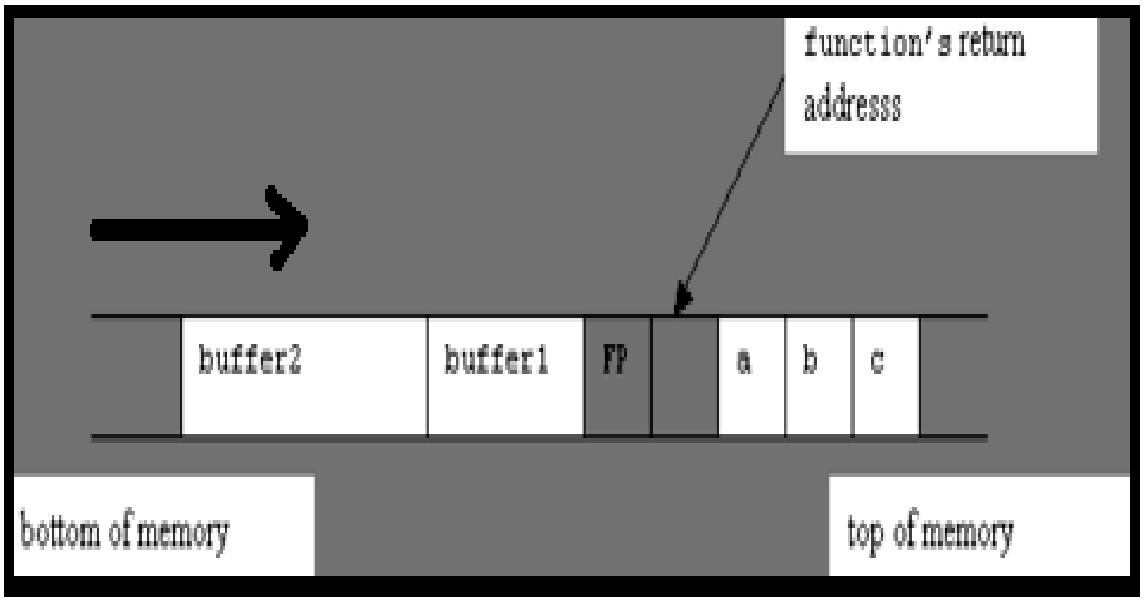
DEMO

- for simple buffer overflow

Activation Records On Stack

```
void function(int a, int b, int c)
{
    char buffer1[5];
    char buffer2[10];
    }

int main()
{
    function(1,2,3);
}
```



Stack Buffer Overflow

```
int i;
void function(void)
{
    char buffer[256]; //create a buffer
    for(i=0;i<512;i++) //iterate 512 times
    buffer[i]='A'; //copy the letter A
}
```

Stack Buffer Overflow (continued)

buffer[256]
Old EBP=0x0012FFF0
Ret EIP =0x00401000

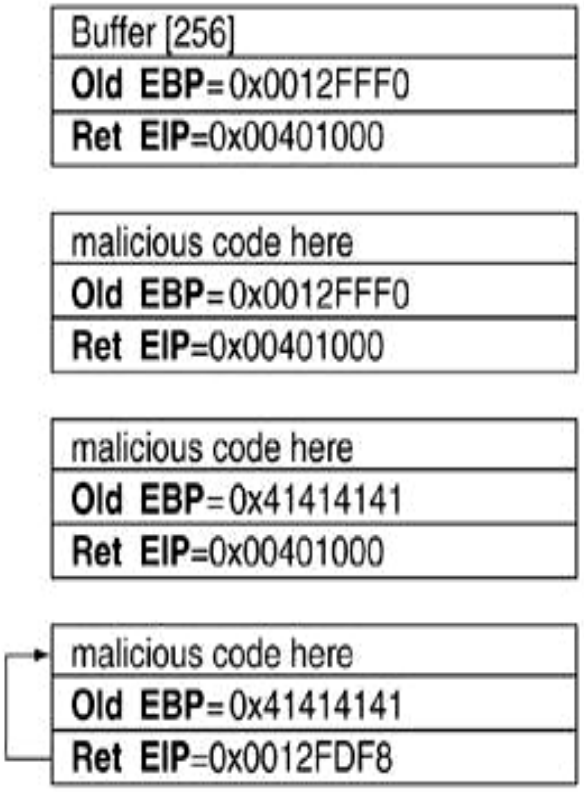
AAAAAAAAAAAAAAAA
Old EBP=0x0012FFF0
Ret EIP =0x00401000

AAAAAAAAAAAAAAAA
Old EBP=0x41414141
Ret EIP =0x00401000

AAAAAAAAAAAAAAAA
Old EBP=0x41414141
Ret EIP =0x41414141

1. A function is using a buffer 256 bytes long. The program attempts to fill the buffer with 512 As.
2. After 256 As, the buffer is full and any remaining As will begin to overflow into adjacent memory.
3. The remaining As begin to overwrite the old EBP.
4. And also overwrite the return EIP.

Exploiting Stack Buffer Overflow

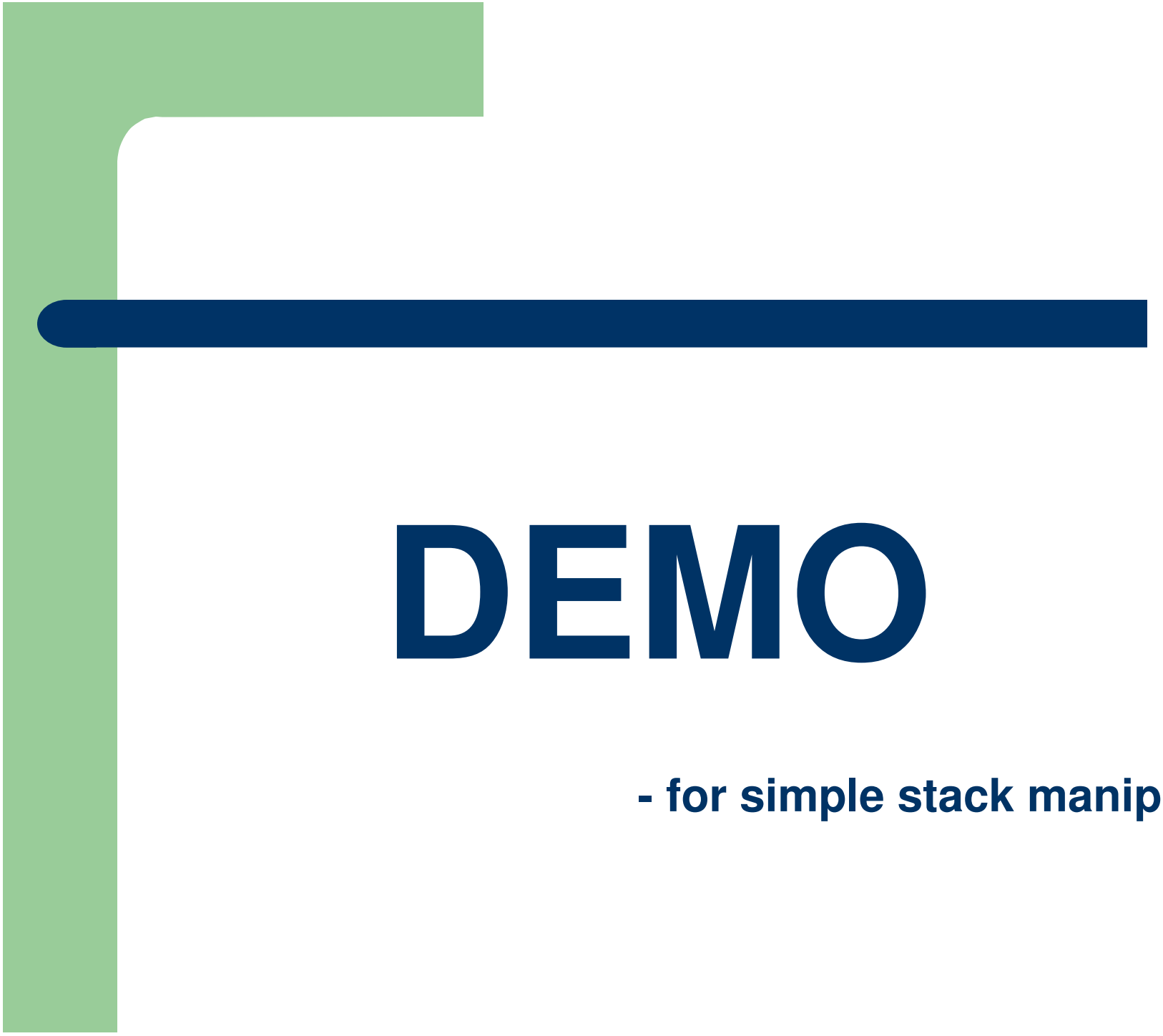


1. A function is using a buffer 256 bytes long. The program attempts to fill the buffer with the attackers code.

2. After 256 bytes, the buffer is full and any remaining bytes will begin to overflow into adjacent memory.

3. First EBP is overwritten.

4. And then EIP is overwritten with the address pointing back to the malicious code. Now, the program will begin to execute the malicious code.


DEMO
- for simple stack manipulation

Causes of Stack Buffer Overflow

- Use of common functions that do not limit the amount of data copied from one location to another
- For example, **`strcpy`** in C

Prevention

- Choice of programming language
- Use of safe libraries
- Stack-smashing protection
- Executable space protection

Thanks!

Questions?