CS 547: Foundation of Computer Security

S. Tripathy
IIT Patna

Previous Class

- Program security
  - Motivation and background
  - Buffer Overflow
  -

This Class

- Program security
- Buffer Overflow
  - Defense
- Incomplete Mediation
- TOCTTOU

Buffer Overflow

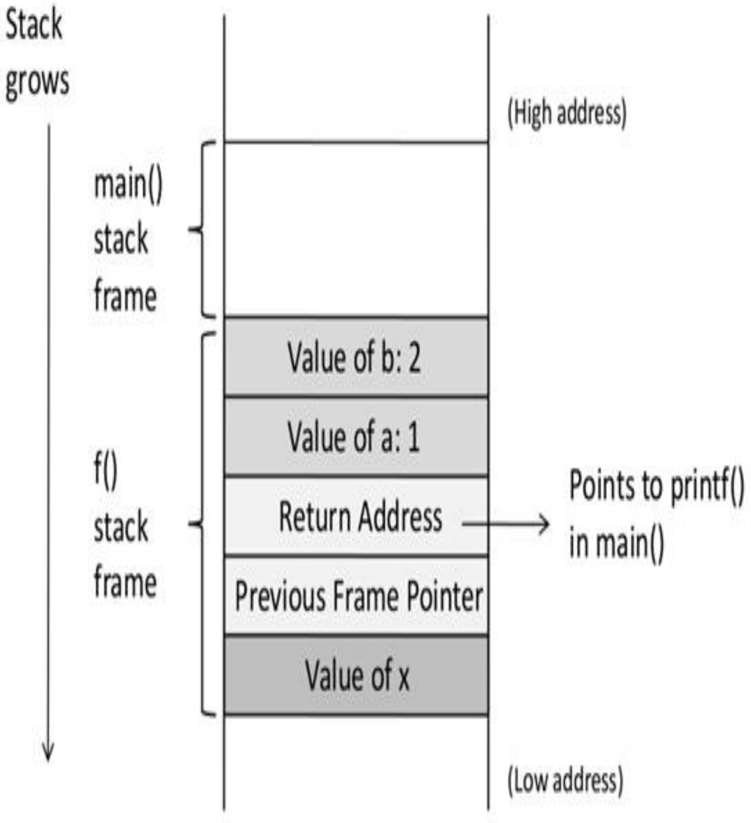
- Programming error when a process attempts to store data beyond the limits of a fixed-sized buffer
  - overwrites adjacent memory locations
  - consequences:
    - corruption of program data
    - unexpected transfer of control
    - execution of code chosen by attacker
    - memory access violations

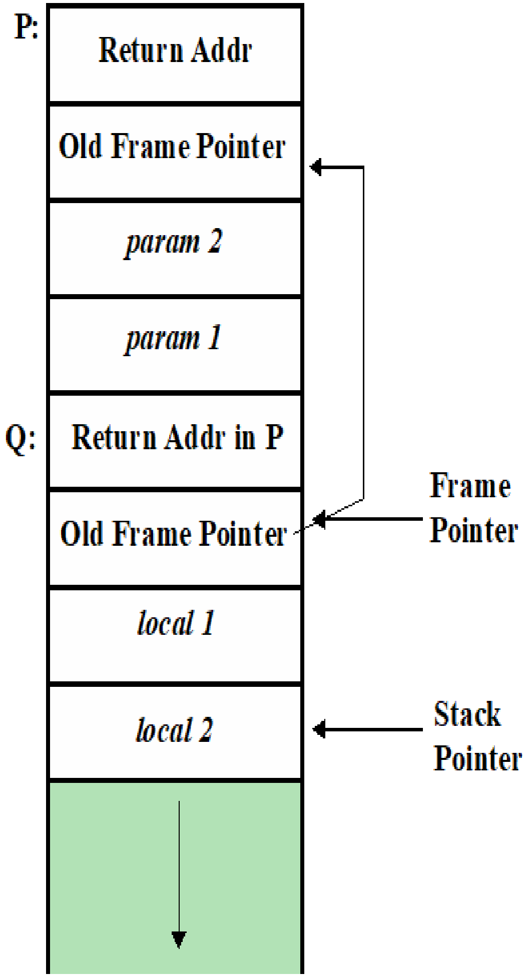
Stack Buffer Overflows

- occur when buffer is located on stack
  - also referred to as stack smashing
  - exploits included an unchecked buffer overflow
- still being widely exploited
- stack frame
  - when one function calls another it needs somewhere to save *the return address*
  - also needs locations to save the parameters to be passed in to the called function and to possibly save register values

Function Call Stack

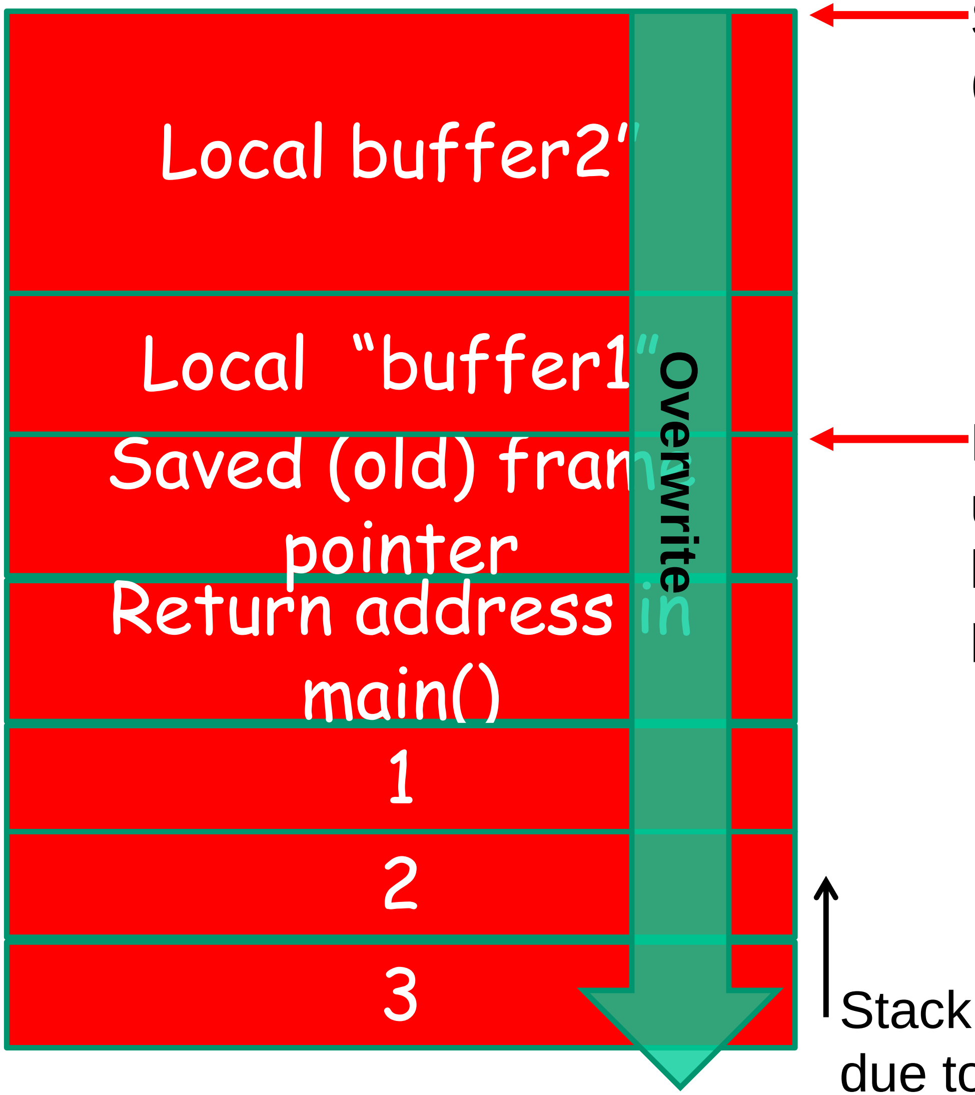
```
void f(int a, int b)
{
  int x;
}
void main()
{
  f(1,2);
  printf("hello world"
 }
```





Stack: Overflowing buffer

↑ Lower-numbered addresses



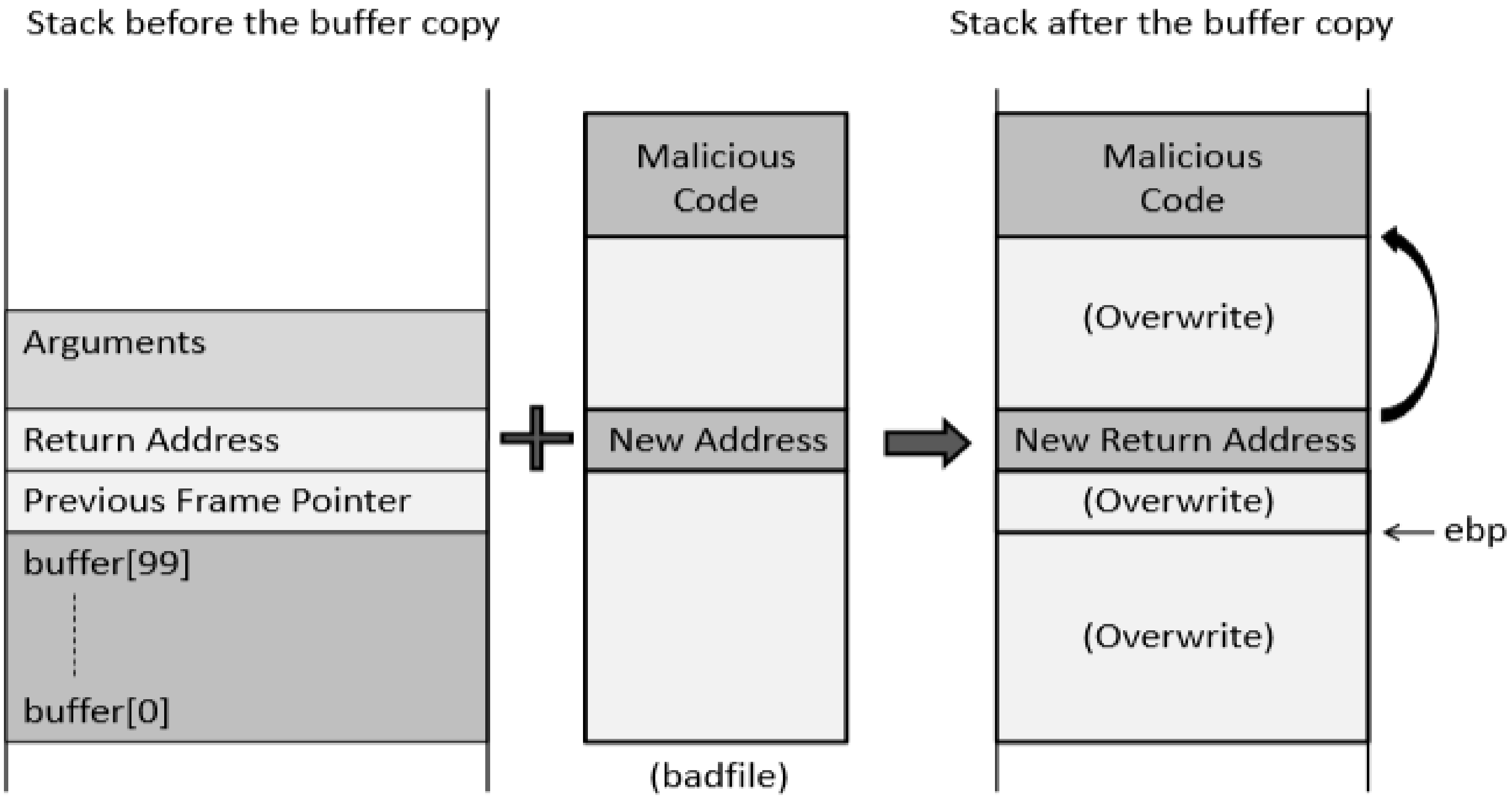
Stack pointer (SP) (current top of stack)

Frame pointer (FP) – use this to access local variables & parameters

↓ Higher-numbered addresses

↑ Stack grows, e.g., due to procedure call

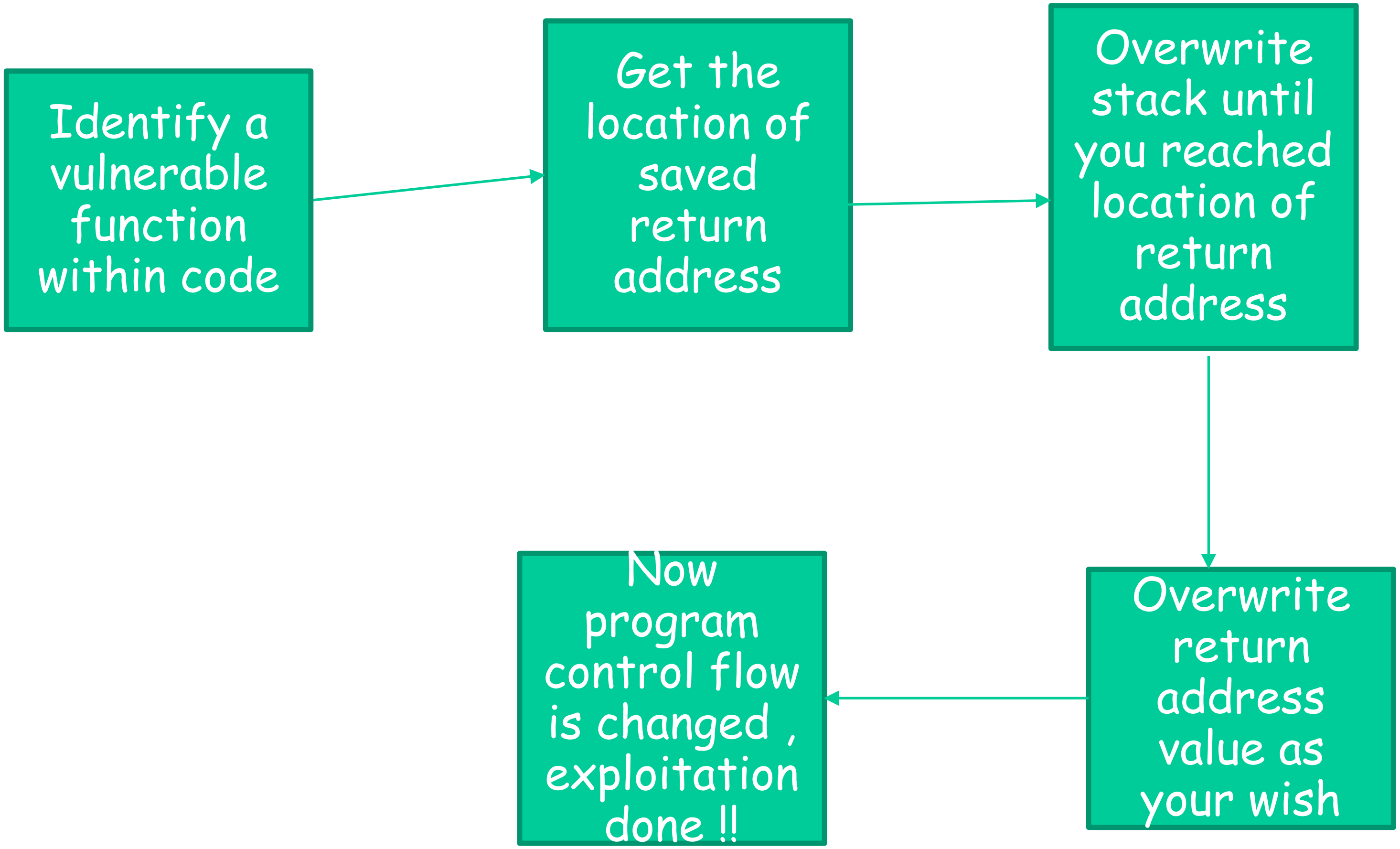
How to Run Malicious Code



Buffer Overflow Attacks

A. Find the offset distance between the base of the buffer and return address so that we can modify the return with address of the code.

B. Find the address for the malicious code. If you don't handle this, your malicious code will be stored in some random



Identify a vulnerable function within code

Get the location of saved return address

Overwrite stack until you reached location of return address

Overwrite return address value as your wish

Now program control flow is changed , exploitation done !!

- Thanks