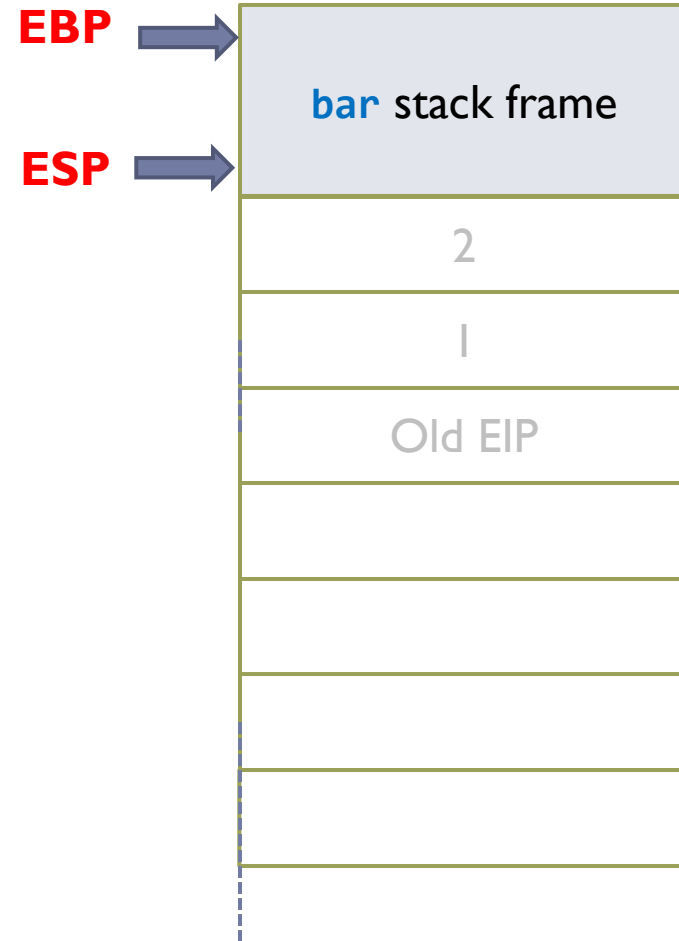


Function Call Convention

Step 10: Delete function parameters.

- ▶ Pop values from the stack (1, 2).
- ▶ **ESP** is also updated (old **ESP**) due to the pop operation.
- ▶ Function parameters (1, 2) are deleted from the stack.
- ▶ Continue the execution in function **bar**.

```
void bar( ) {  
    foo(1, 2);  
}  
int foo(int x, int y){  
    int z = x + y;  
    return z;  
}
```



Outline

- ▶ Review: Memory Layout and Function Call Convention
- ▶ **Buffer Overflow Vulnerability**

A Common Vulnerability in C Language

String

- ▶ An array of characters (1 Byte).
- ▶ Must end with NULL (or `'\0'`). A string of length n can hold only $n-1$ characters, while the last character is reserved for NULL.

`char* strcpy (char* dest, char* src)`

- ▶ Copy string `src` to `dest`
- ▶ No checks on the length of the destination string.

What if the source string is larger than destination string?

```
char str[6] = "Hello";
```

	X
	X
[5]	\0
[4]	o
[3]	l
[2]	l
[1]	e
[0]	H

```
char* strcpy (char* dest, const char* src) {  
    unsigned i;  
    for (i=0; src[i] != '\0'; ++i)  
        dest[i] = src[i];  
    dest[i] = '\0';  
    return dest;  
}
```

General Idea

More data into a memory buffer than the capacity allocated.

Overwriting other information adjacent to that memory buffer.

Key reason: C does not check boundaries when copying data to the memory.



High coverage

Any system implemented using C or C++ can be vulnerable.

- ▶ Program receiving input data from untrusted network
sendmail, web browser, wireless network driver, ...
- ▶ Program receiving input data from untrusted users or multi-user systems
services running with high privileges (root in Unix/Linux, SYSTEM in Windows)
- ▶ Program processing untrusted files
downloaded files or email attachment.
- ▶ Embedded software
mobile phones with Bluetooth, wireless smartcards, airplane navigation systems, ...

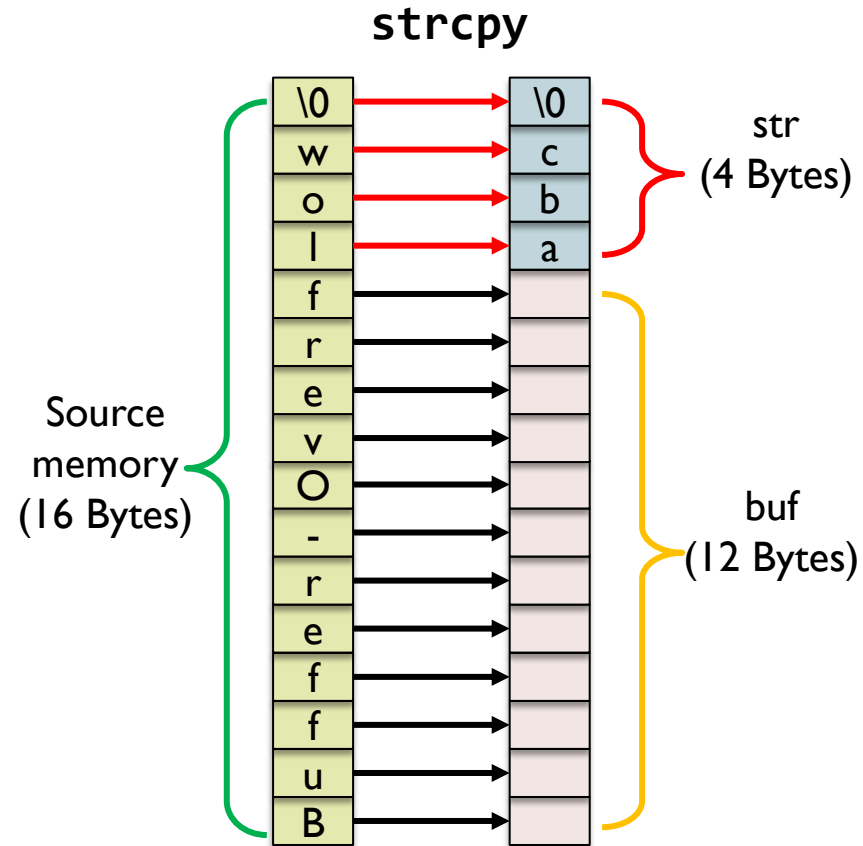


Example of Buffer Overflow

Corruption of program data

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

int main(int argc, char* argv[]) {
    char str[4] = "abc";
    char buf[12];
    strcpy(buf, "Buffer-Overflow");
    printf("str is %s\n", str);
    return 0;
}
```



Potential Consequences

```
int Privilege-Level = 3;  
char buf[12];  
strcpy(buf, ".....");
```

Privilege escalation

```
int Authenticated = 0;  
char buf[12];  
strcpy(buf, ".....");
```

Bypass authentication

```
char command[] = "/usr/bin/ls";  
char buf[12];  
strcpy(buf, ".....");  
execv(command, ...);
```

Execute arbitrary command

```
int (*foo)(void);  
char buf[12];  
strcpy(buf, ".....");  
foo();
```

Hijack the program control

.....