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

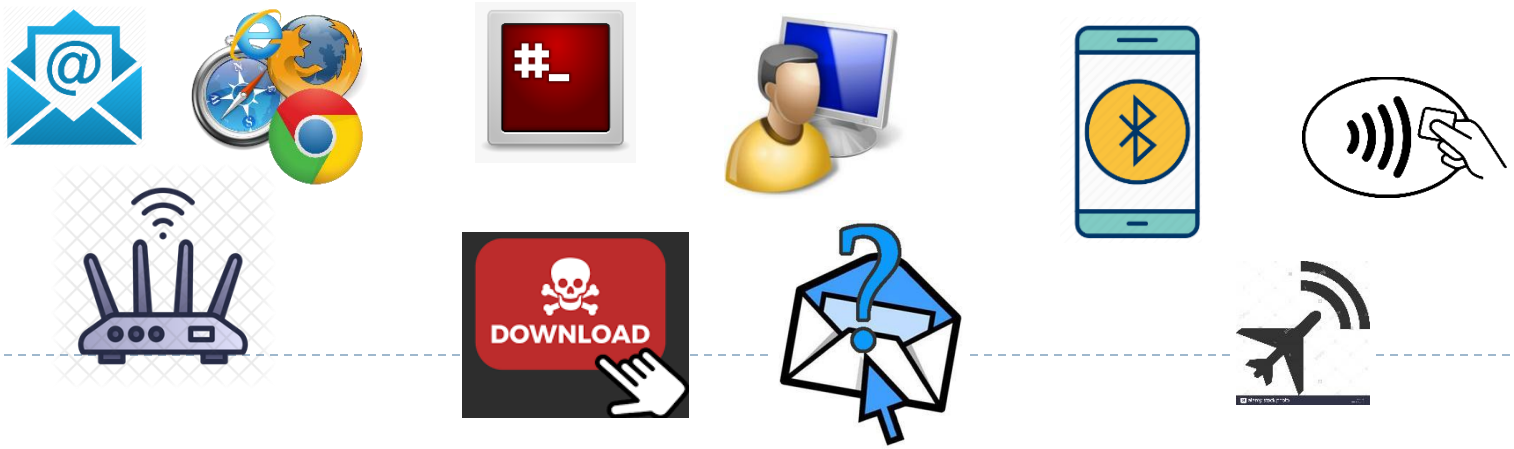
  services running with high privileges (root in Unix/Linux, SYSTEM in Windows)

- Program processing untrusted files

  downloaded files or email attachment.

- Embedded software

  mobile phones with Bluetooth, wireless smartcards, airplane navigation systems, …

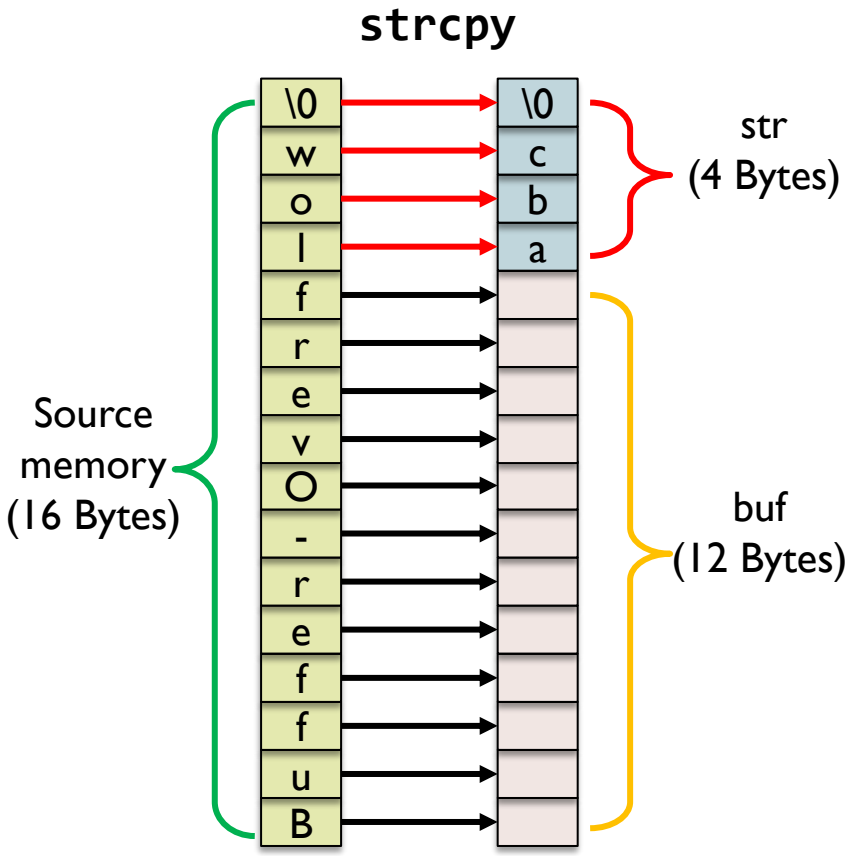


Example of Buffer Overflow

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

.....

More Vulnerability Functions

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

- Append the string *src* to the end of the string *dest*.

```
char* gets (char* str)
```

- Read data from the standard input stream (stdin) and store it into *str*.

```
int* scanf (const char* format, …)
```

- Read formatted input from standard input stream.

```
int sprintf (char* str, const char* format, …)
```

- Create strings with specified formats, and store the resulting string in *str*.

and more…

Stack Smashing

Function call convention:

- Step 2: Push the current instruction pointer (EIP) to the stack.
- Step 6: Execute the callee function within its stack frame.
- Step 9: Restore EIP from the stack.

Overwrite EIP on the stack during the execution of the callee function (step 6).

After callee function is completed (step 9), it returns to a different (malicious) function instead of the caller function!