Overflow countermeasures

- Staying within bounds
  - Check lengths before writing
  - Confirm that array subscripts are within limits
  - Limit input to the number of acceptable characters
  - Limit programs' privileges to reduce potential harm
- Many languages have overflow protections
- **Canary values in stack to signal modification**
- Leveraging protections of platforms
- Code analyzers can identify many overflow vulnerabilities

StackGuard

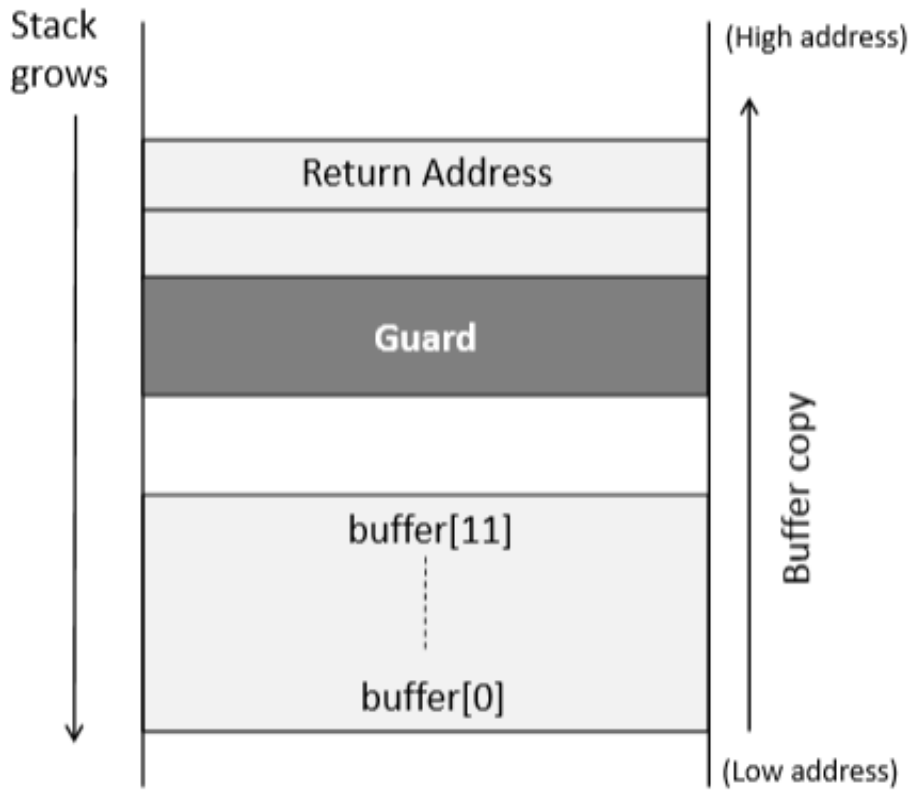
- Runtime tests for stack integrity.
- Embed “canaries” (or guards) in stack frames and verify their integrity prior to function return.
- StackGuard implemented as a GCC patch
  - Program must be recompiled (now enabled by default)
- Note: Canaries do not provide full protection
  - Some buffer overflow attacks leave canaries unchanged

StackGuard (Cond.)

```
void foo (char *str)
{
    int guard;
    guard = secret;

    char buffer[12];
    strcpy (buffer, str);

    if (guard == secret)
        return;
    else
        exit(1);
}
```



Execution with StackGuard

```
seed@ubuntu:~$ gcc -o prog prog.c
seed@ubuntu:~$ ./prog hello
Returned Properly

seed@ubuntu:~$ ./prog hello00000000000
*** stack smashing detected ***:  ./prog terminated
```

Canary check done by compiler.

Prog.c is a vulnerable program which takes input from the user and calls a function foo() which copies to a buffer.

```
foo:
.LFB0:
    .cfi_startproc
    pushl    %ebp
    .cfi_def_cfa_offset 8
    .cfi_offset 5, -8
    movl    %esp, %ebp
    .cfi_def_cfa_register 5
    subl    $56, %esp
    movl    8(%ebp), %eax
    movl    %eax, -28(%ebp)
    // Canary Set Start
    movl %gs:20, %eax
    movl %eax, -12(%ebp)
    xorl %eax, %eax
    // Canary Set End
    movl    -28(%ebp), %eax
    movl    %eax, 4(%esp)
    leal    -24(%ebp), %eax
    movl    %eax, (%esp)
    call    strcpy
    // Canary Check Start
    movl -12(%ebp), %eax
    xorl %gs:20, %eax
    je .L2
    call __stack_chk_fail
    // Canary Check End
```

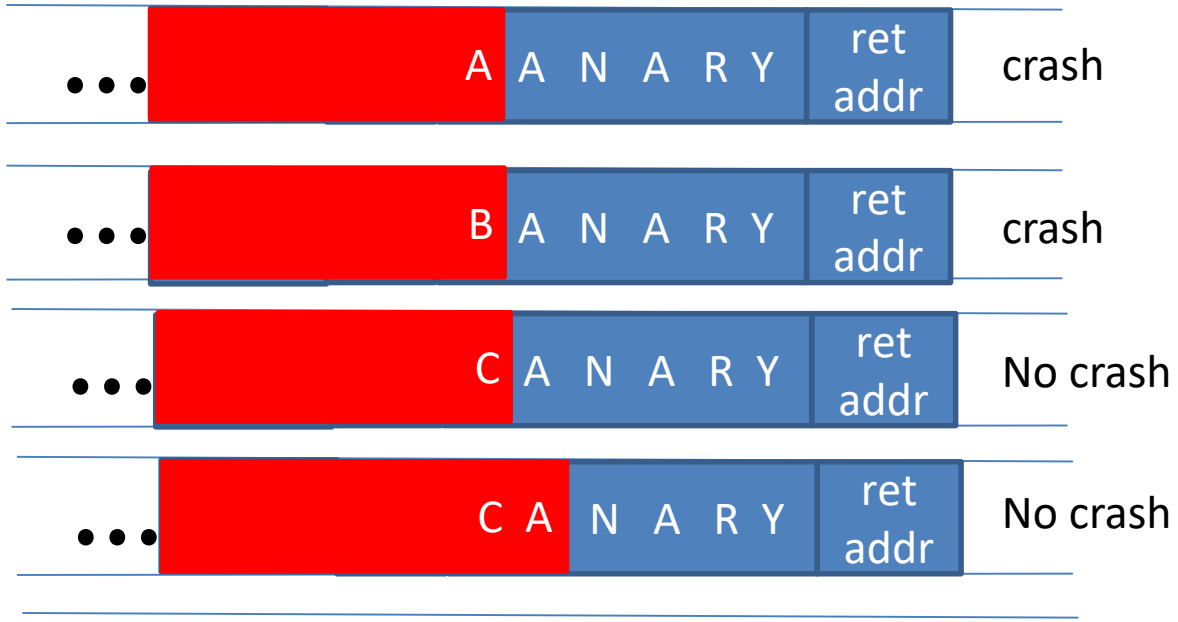
Evasion: canary extraction

A common design for crash recovery:

- When process crashes, restart automatically (for availability)
- Often canary is unchanged (reason: relaunch using fork)

Danger:

- canary extraction byte by byte

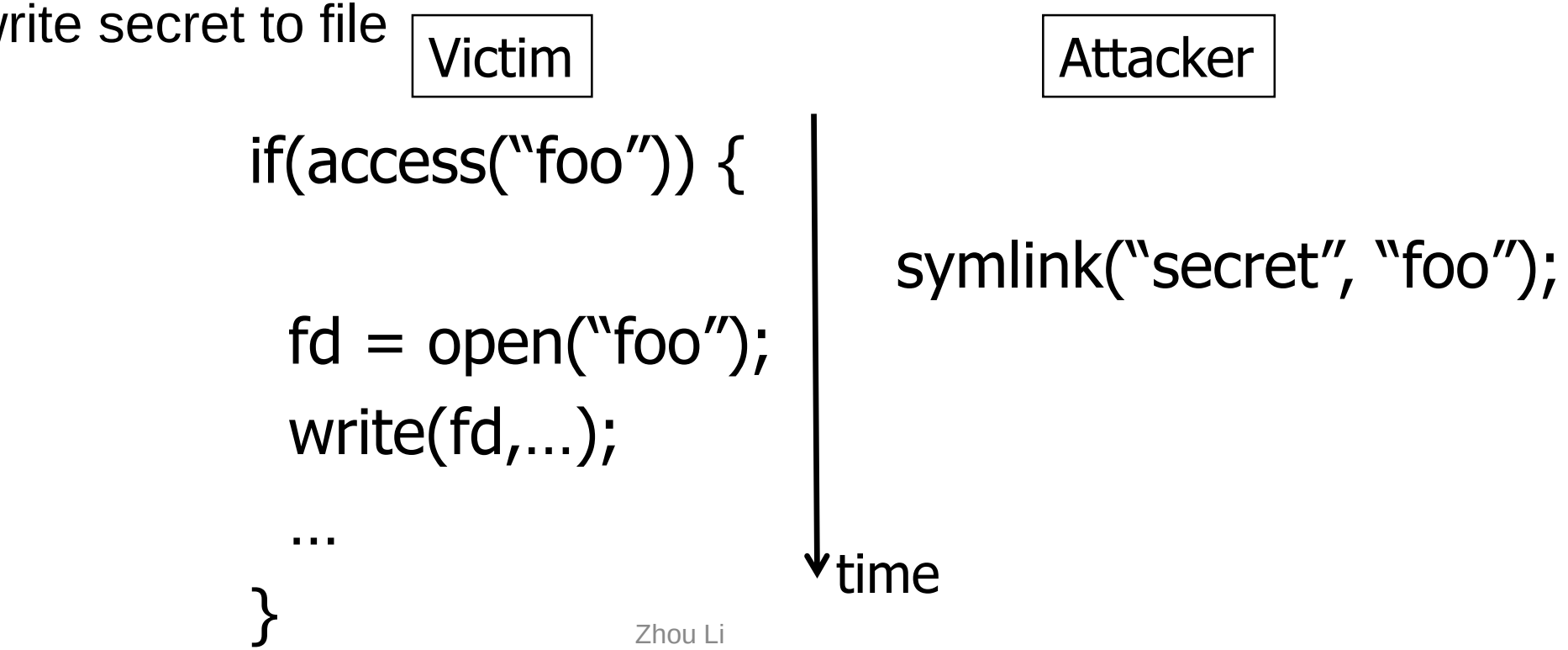


Time-of-Check to Time-of-Use (TOCTTOU)

- A class of software bugs caused by changes in a system between the *checking* of a condition (such as a security credential) and the *use* of the results of that check.
- Security implication
  - Access control is ineffective
  - Confidentiality and integrity failure
- Countermeasures
  - Access-checking software must own the request data until the requested action is complete.
  - Ensure serial integrity, or no interruption during validation.

TOCTTOU example

- Victim checks file, if its good, opens it
- Attacker changes interpretation of file name
- Victim write secret to file



Undocumented Access Point (Backdoor)

- Created by programmer but forgot to remove in release
  - Interrogate data values during execution
  - Special debug mode to test conditions
- Could be exploited to compromise a machine
  - Security should not depend on secrecy of backdoor
  - Barracuda Networks products accepted remote logins from username "product" and no password, if coming from a specific address range

Integer overflows

Problem: what happens when int exceeds max value?

int m; (32 bits) **short s; (16 bits)** **char c; (8 bits)**

c = 0x80 + 0x80 = 128 + 128 ⇒ c = 0

s = 0xff80 + 0x80 ⇒ s = 0

m = 0xffffff80 + 0x80 ⇒ m = 0

Can this be exploited?

An example

```
void  func( char *buf1, *buf2,    unsigned int len1, len2) {
      char temp[256];
      if  (len1 + len2 > 256)  {return -1}          // length check
      memcpy(temp, buf1, len1);                    // cat buffers
      memcpy(temp+len1, buf2, len2);
      do-something(temp);                          // do stuff
}
```

What if **len1 = 0x80, len2 = 0xffffff80** ?

⇒ len1+len2 = 0

Second memcpy() will overflow heap !!

Unsafe utility program

- Unsafe libc functions

  strcpy (char *dest, const char *src)

  strcat (char *dest, const char *src)

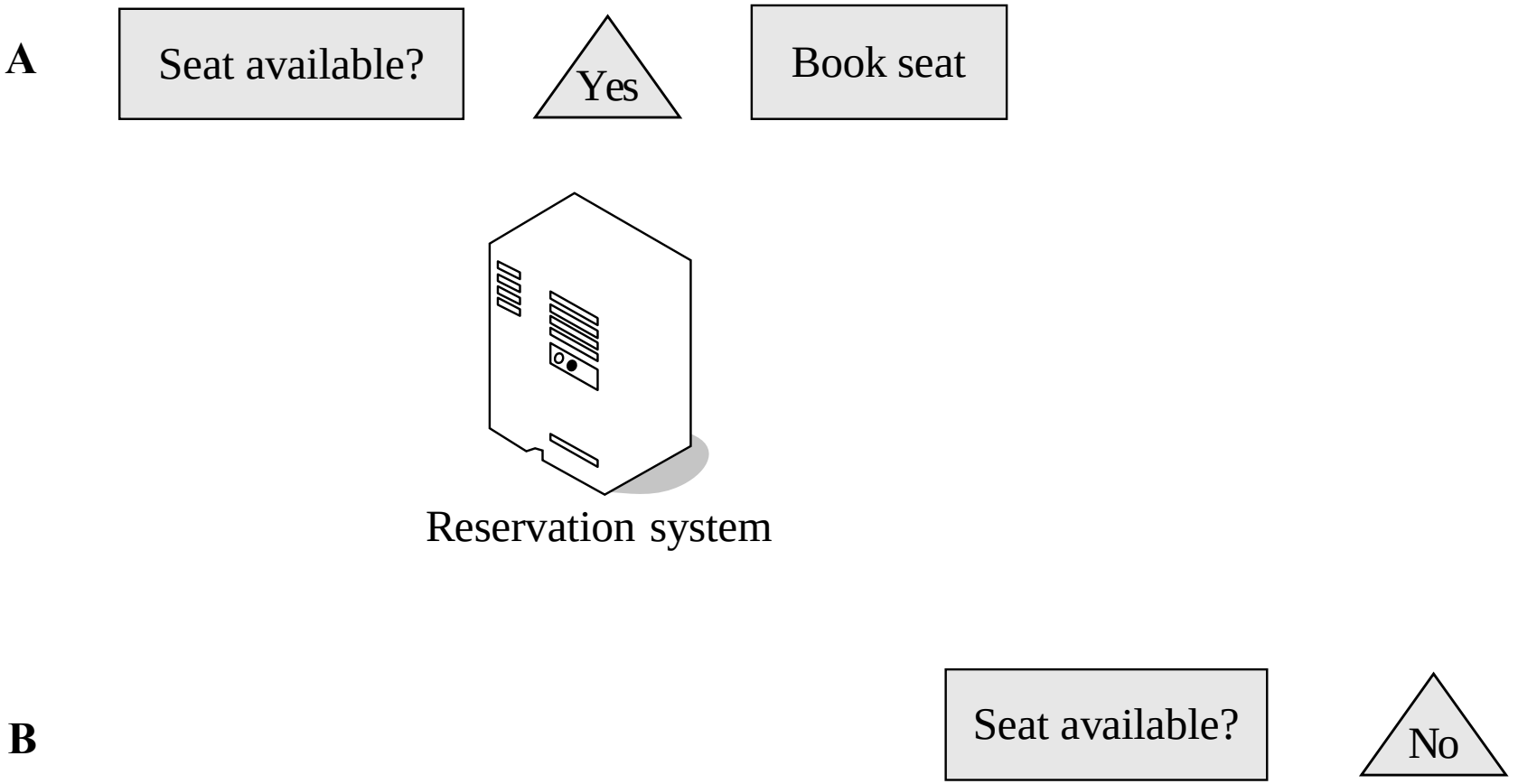
  gets (char *s)

  scanf (const char *format, …) and many more.

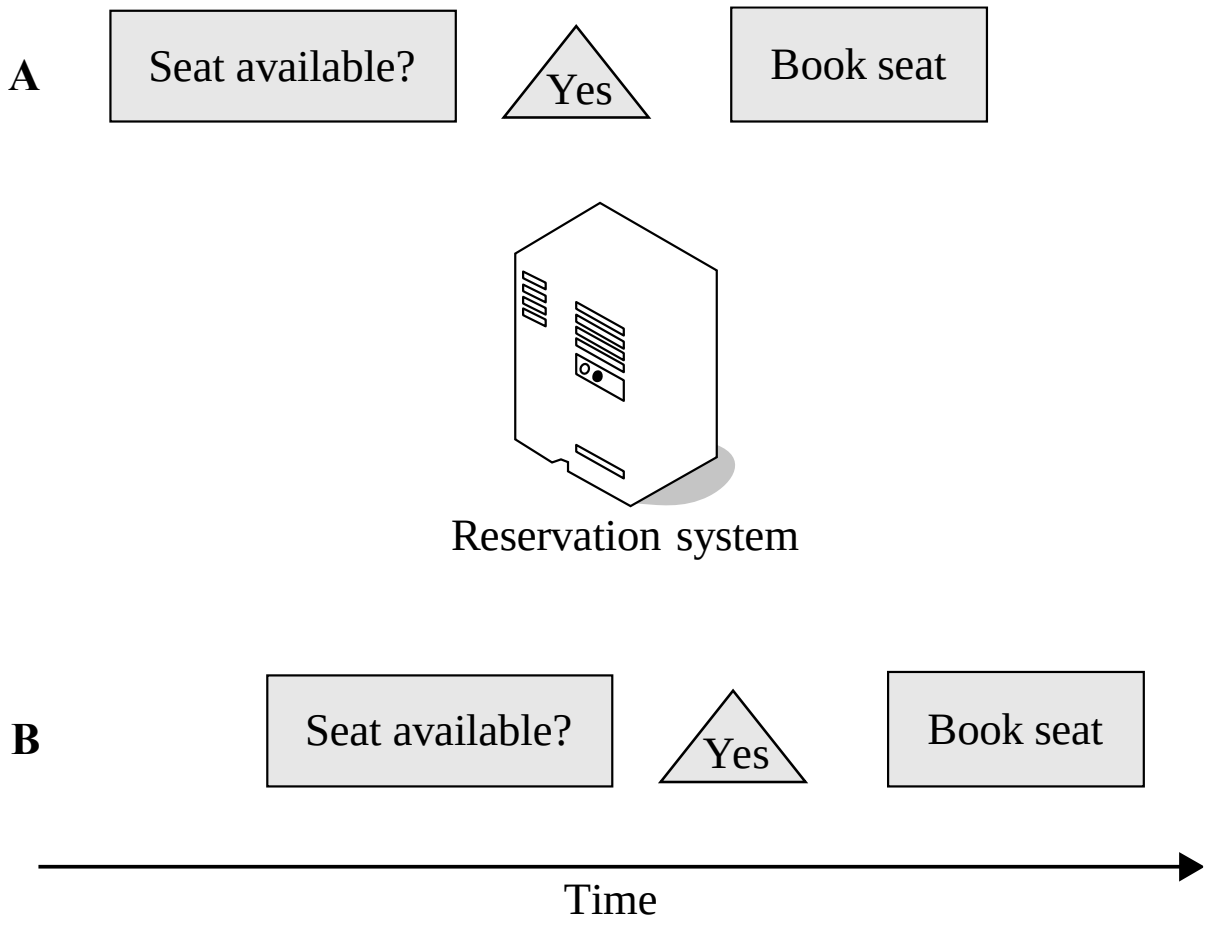
- "Safe" libc versions strncpy(), strncat()
  - But they may leave string unterminated.

- Windows C run time (CRT):
  - strcpy_s (*dest, DestSize, *src): ensures proper termination

No race condition: Example

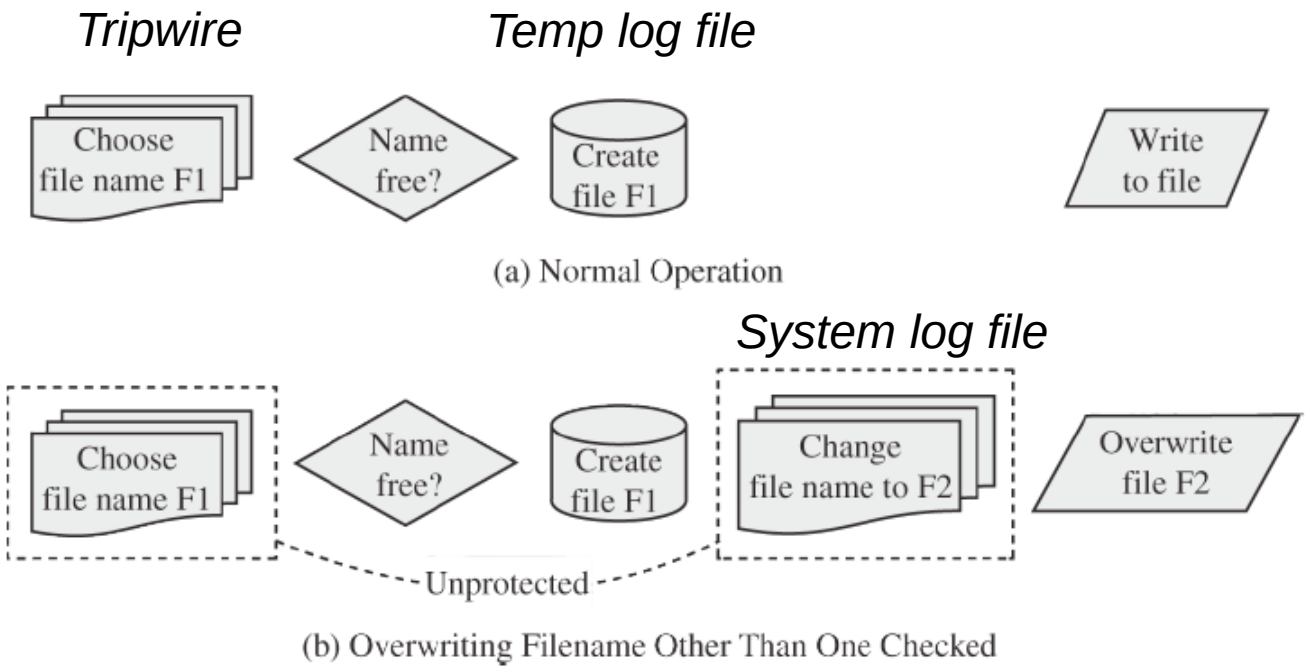


Race condition: Example



Race condition

- Two processes execute concurrently and the outcome of computation depends on the order of instructions
- Security implications: overwriting file (e.g., system log)



Filename race condition

Malware

Malware

- Programs planted by an agent with malicious intent to cause unanticipated or undesired effects
- A threat exploiting vulnerabilities
- Virus
  - A program that can replicate itself and pass on malicious code to other nonmalicious programs by modifying them
- Worm
  - A program that spreads copies of itself through a network
- Trojan horse
  - Code that, in addition to its stated effect, has a second, nonobvious, malicious effect

Type of malware

Code Type	Characteristics
Virus	Code that causes malicious behavior and propagates copies of itself to other programs
Trojan horse	Code that contains unexpected, undocumented, additional functionality
Worm	Code that propagates copies of itself through a network; impact is usually degraded performance
Rabbit	Code that replicates itself without limit to exhaust resources
Logic bomb	Code that triggers action when a predetermined condition occurs
Time bomb	Code that triggers action when a predetermined time occurs
Dropper	Transfer agent code only to drop other malicious code, such as virus or Trojan horse
Hostile mobile code agent	Code communicated semi-autonomously by programs transmitted through the web
Script attack, JavaScript, Active code attack	Malicious code communicated in JavaScript, ActiveX, or another scripting language, downloaded as part of displaying a web page

Types of malware (cond.)

Code Type	Characteristics
RAT (remote access Trojan)	Trojan horse that, once planted, gives access from remote location
Spyware	Program that intercepts and covertly communicates data on the user or the user's activity
Bot	Semi-autonomous agent, under control of a (usually remote) controller or "herder"; not necessarily malicious
Zombie	Code or entire computer under control of a (usually remote) program
Browser hijacker	Code that changes browser settings, disallows access to certain sites, or redirects browser to others
Rootkit	Code installed in "root" or most privileged section of operating system; hard to detect
Trapdoor or backdoor	Code feature that allows unauthorized access to a machine or program; bypasses normal access control and authentication
Tool or toolkit	Program containing a set of tests for vulnerabilities; not dangerous itself, but each successful test identifies a vulnerable host that can be attacked
Scareware	Not code; false warning of malicious code attack

Harm from malicious code

- Harm to users and systems:
  - Deleting or encrypting files
  - Modifying system information, such as the Windows registry
  - Stealing sensitive information, such as passwords
  - Attaching to critical system files
- Harm to the world:
  - Some malware has been known to infect millions of systems, growing at a geometric rate
  - Infected systems often become staging areas for new infections

Ransomware: Trojan Horse? Worm? Virus?

Is it a malware?