# CS 526: Information Security

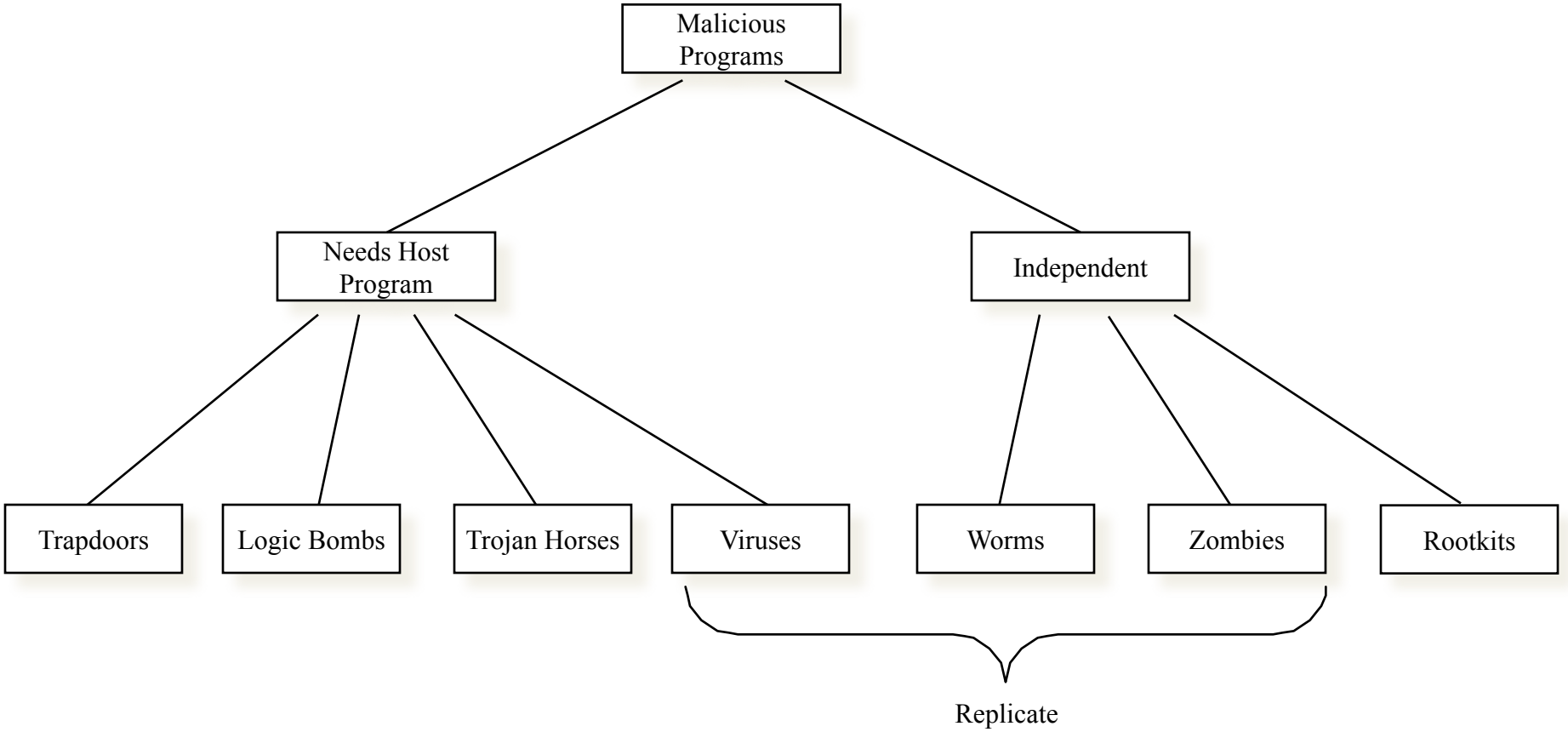
## Malicious Programs

# What is a Malicious Program

- **Malware**: software designed to infiltrate or damage a computer system without the owner's informed consent

- **Spyware:** software designed to intercept or take partial control over the user's interaction with the computer, without the user's informed consent
  - secretly monitors the user's behavior
  - collect various types of personal information

# Taxonomy of Malicious Programs



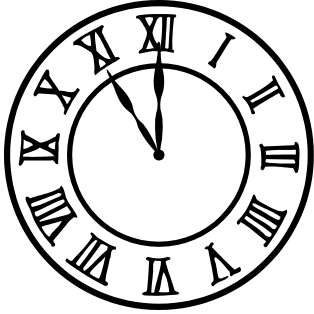
# Trapdoor

- Secret entry point into a system
  - Circumvents normal protections
  - Allows a way to sneak in
  - Often inserted by an attacker after a break-in, with malicious intent
  - Sometimes initially built into system accidentally or by design, typically without malicious intent
  - Example: In early Unix login (~1982) entering password “hastalavista” gave access to any user account
- Presents a security risk

# Trapdoor (cont'd)

- Why would a trapdoor be installed at design time?
  - And how can that be without malicious intent?
- Initial intent can be legitimate use, e.g.,
  - Debugging
  - Troubleshooting
  - Maintenance
- Initial intent is often to remove it before shipping to customers
  - But somehow it is inadvertently not removed

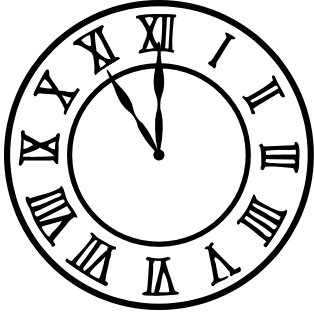
# Logic Bomb, Time Bomb



- Embedded in legitimate programs
  - Maliciously
- Activated when specified conditions met

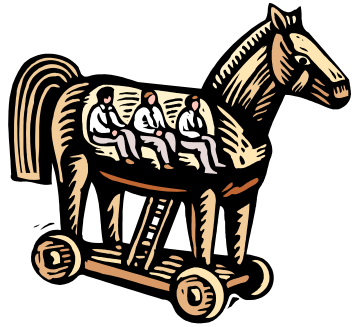
- When triggered, can damage system
  - Modify/delete files/disks

# Logic Bomb, Time Bomb



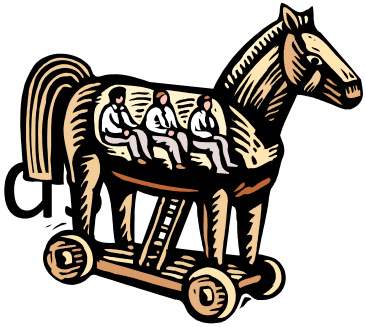
- Examples of logic bomb triggers
  - Presence/absence of some file
  - Absence of a particular user, e.g., check for employee ID in payroll, detonate if absent
- Examples of time bomb triggers
  - Particular date/time
  - Timeouts, e.g., software stop working after a target time (sometimes used by software developers, which they disable after customer action like renewing the licence)

# Trojan Horse



- Program with both an expected effect and an unexpected effect
  - Appears normal/expected (e.g., plays chess)
  - Unexpected effect (aka “payload”) that violates security policy
- User tricked into executing Trojan horse
  - Expects (and sees) overt behavior
  - Payload (usually malicious) is often covert
- Non-replicating
  - Can self-destruct before detection

# Typical Trojan Horse Payload



- Crude damage
  - Electronic (e.g., erase files)
  - Physical (if computer controls a physical plant)
- Information-stealing
  - Password, SSN, DoB, …
- Trapdoor installation
- Malware installation
  - On hard drive (“dropper”)
  - In memory (“injector”)

# Virus

- Self-replicating code
  - Like replicating Trojan horse
  - Alters normal code with “infected” version
- No overt action
  - Generally tries to remain undetected
- Operates when infected code is executed, e.g.,

  If *spread condition* then

  For *target files*

  if *not infected* then *alter to include virus*

  Perform malicious action

  Execute normal program

# Virus Types: File infectors

- Transient (aka direct-action, non-resident)
  - Executes when infected host program is executed
  - Does not linger after host program terminates
- Resident
  - Installs itself in memory when infected host prog is executed, stays there after host terminates
- Malicious code usually at beginning of legitimate prog it infects
  - Virus executes first
  - App then runs normally

# Virus Types: Boot-record infectors

- Infect code found on disk system areas
  - First sector in a HD partition
  - First sector on portable drives
- Resident
- If also file infectors: “multi-partite”, “boot-and-file”
- Most are written for Windows PCs
  - What happens if you run Linux on your PC?

# Virus Types: File system virus

- Modify directory tables
- Virus is executed before program
- Program is not altered
  - Only its directory entry is changed
  - Fools program file integrity checkers (they rely on computing a cryptographic hash of the executable file and decide “clean” if it is as expected)

# Companion virus

- Uses name of legal program
- Fool OS into executing virus instead of legal program
- When done, virus runs its companion
- E.g., modify data structure used by OS, or use precedence rules (.COM precedence over .EXE)
- Fools file integrity checkers

# Virus Evolution

- Early ones were easy to detect
  - Occurred in predictable places
  - Had unchanging code
  - Easy for antivirus programs to detect
- Encrypted virus
    - Encrypted virus body
    - Decryption code
  - Encrypt invariant code with different keys
  - Weakness: Same decryption code

# Virus Evolution: Mutation

- Polymorphic
  - Encrypted, plus:
  - Virus uses mutation engine to change decryption code
    - Equivalent, just looks different
    - Insert NOP, use equivalent instructions, re-order independent instructions, etc
  - Decryption code becomes a moving target

# Virus Evolution: More Mutation

- Polymorphic virus detection
  - Emulate target computer
  - Trick virus into decrypting
  - Get invariant code without executing it
- Metamorphic
  - Alters itself too (not just the decrypt code)
- Behavior-based detection
  - Sandboxing

# Infection Behavior

- Fast:  Infect any opened file
- Slow: Infect only created/modified files
  - So that changes reported by integrity checker are believed to be legitimate
- Sparse: Infect occasionally
  - Not every execution
  - Only certain files
  - Minimizes probability of detection

# Macro Virus

- Infected “executable” is not machine code
  - Relies on something “executed” inside application data
  - Common example:  Macros
- Otherwise similar to other viruses
  - Architecture-independent
  - Application-dependent
  - Capable of “social engineering” (use of address book)

# Worm

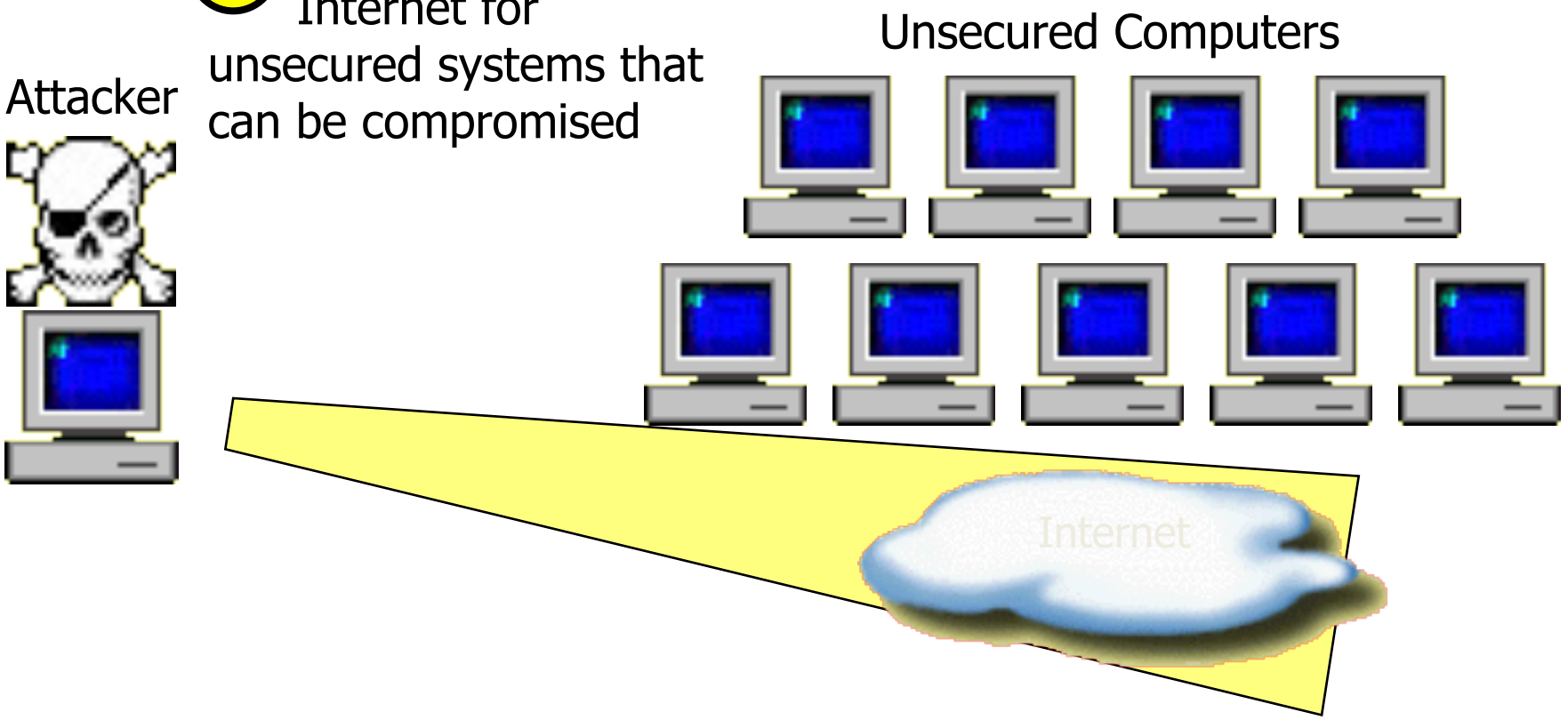
- Runs independently
  - Does not require a host program
- Propagates a fully working version of itself to other machines
- Carries a payload performing hidden tasks
  - Trapdoors, spam relays, DDoS agents; …
- Phases
  - Probing ➔ Exploitation ➔ Replication ➔ Payload

# Botnet

- Secretly take over other networked computers by exploiting software flaws
- Build the compromised computers into a zombie network or botnet (= ro**bot** + **net**work)
  - a collection of compromised machines running malware programs, under a common command and control infrastructure
- Use it to indirectly launch attacks
  - E.g., DDoS, phishing, spamming, cracking

# Detailed Steps (1)

1 Attacker scans Internet for unsecured systems that can be compromised



# Detailed Steps (2)

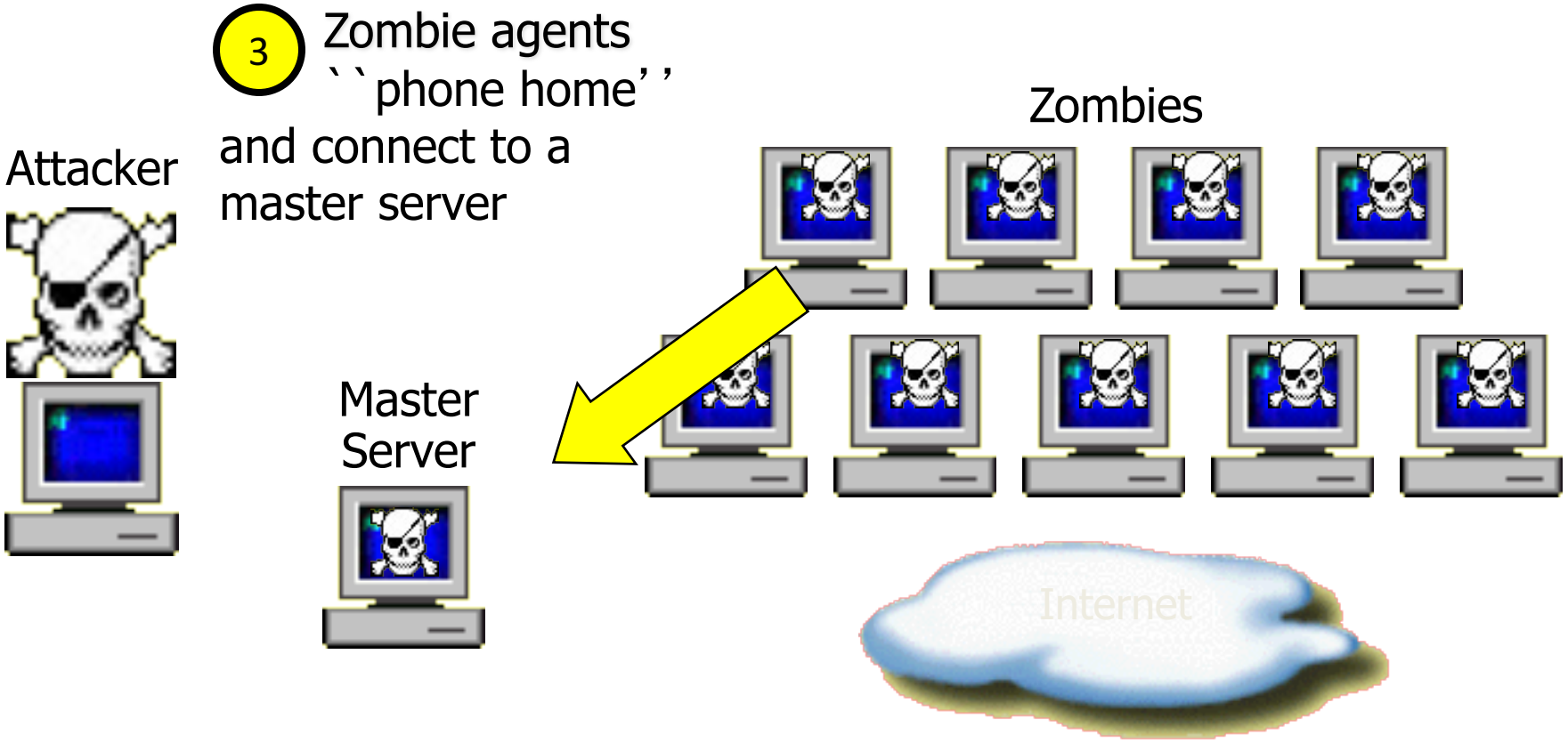
2 Attacker secretly installs zombie agent programs, turning unsecured computers into zombies

Attacker

Unsecured Computers

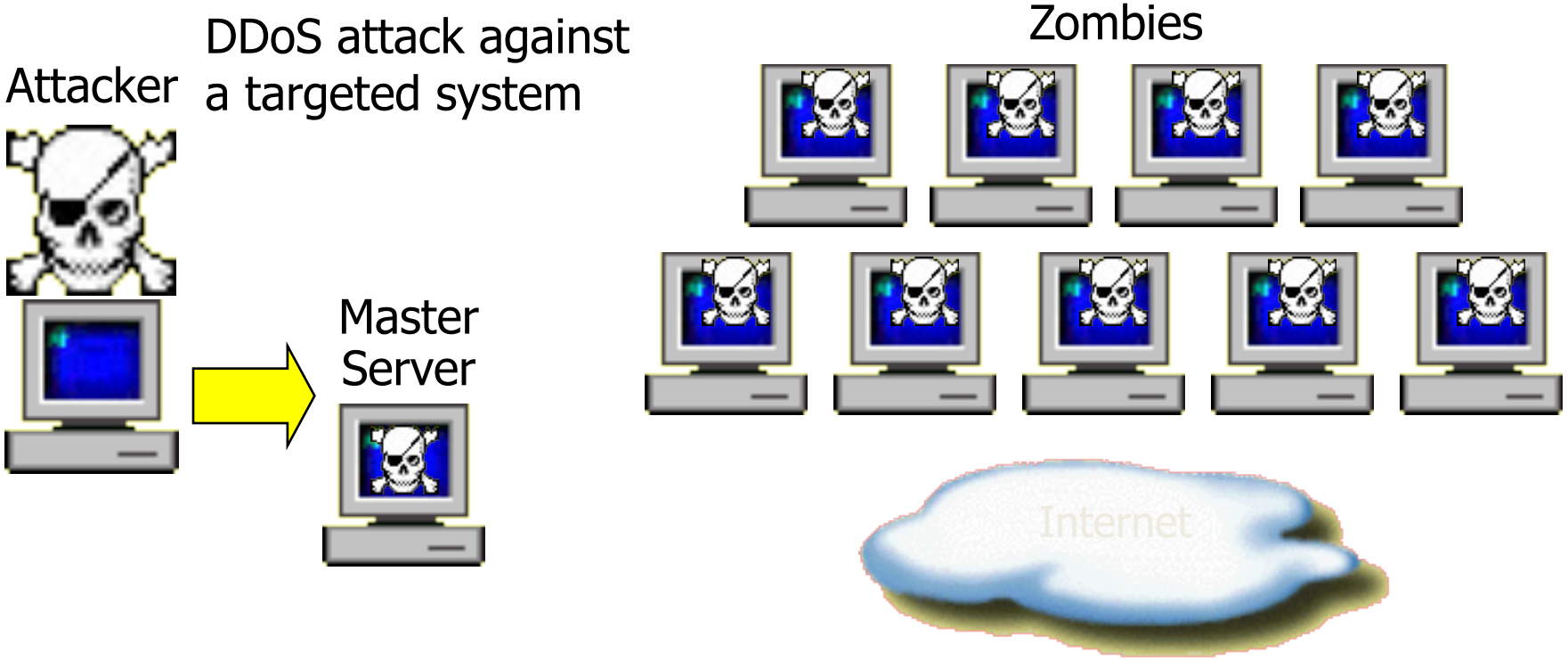
Zombies

# Detailed Steps (3)

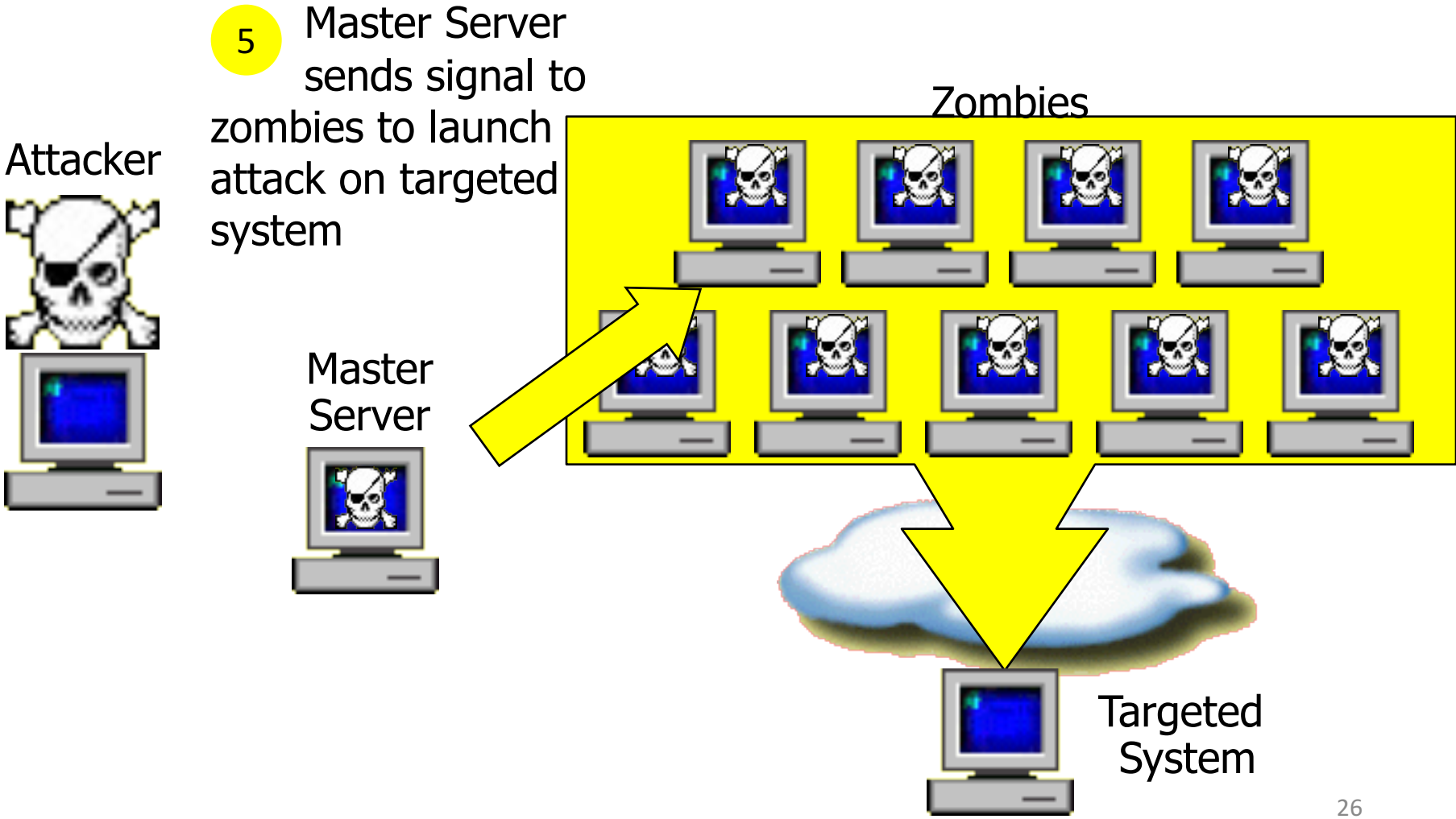


# Detailed Steps (4)

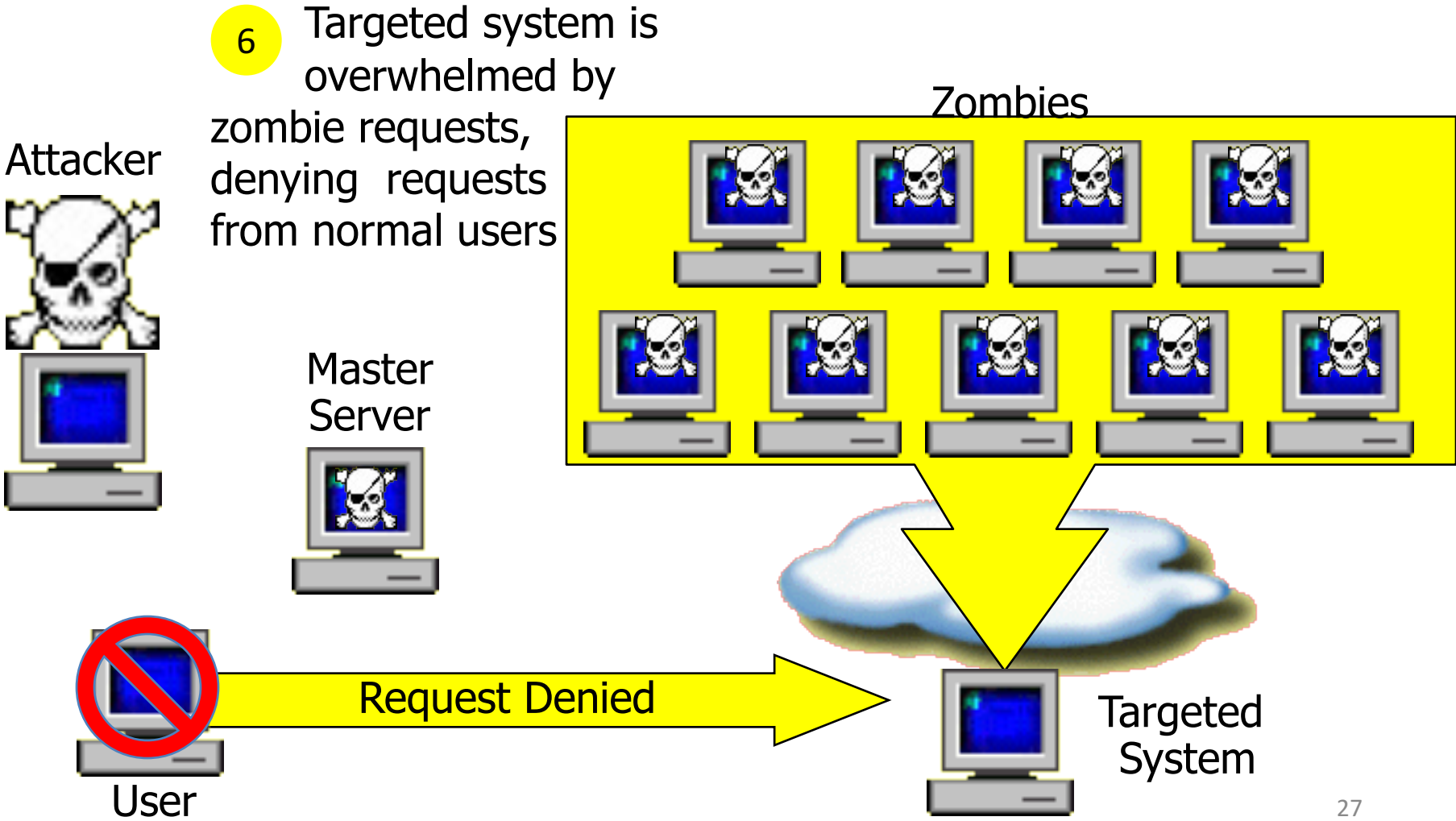
4 Attacker sends commands to Master Server to launch a DDoS attack against a targeted system



# Detailed Steps (5)



# Detailed Steps (6)

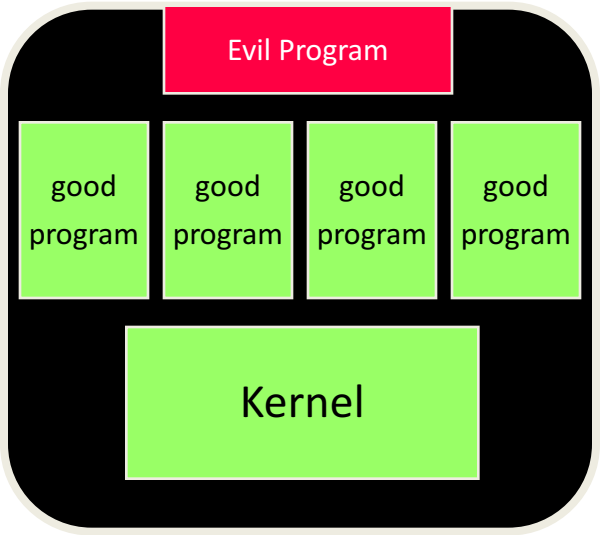


# Rootkit

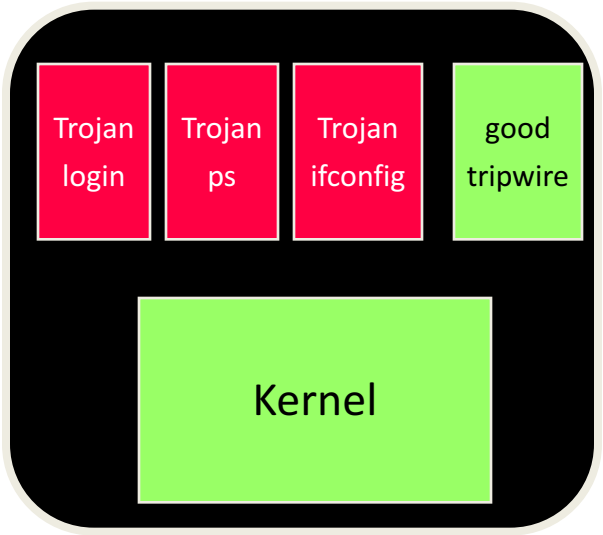
- Software used after system compromise to:
  - Hide the attacker's presence ("stealth")
  - Provide trapdoors for easy reentry ("remote control")
  - Turn laptop into physical spying device
  - Obtain information for financial fraud
- Simple rootkits:
  - Modify user programs (ls, ps)
  - Detectable by integrity checkers (change-detection)
- Sophisticated rootkits:
  - Modify the kernel itself
  - Hard to detect from userland

# Rootkit Classification

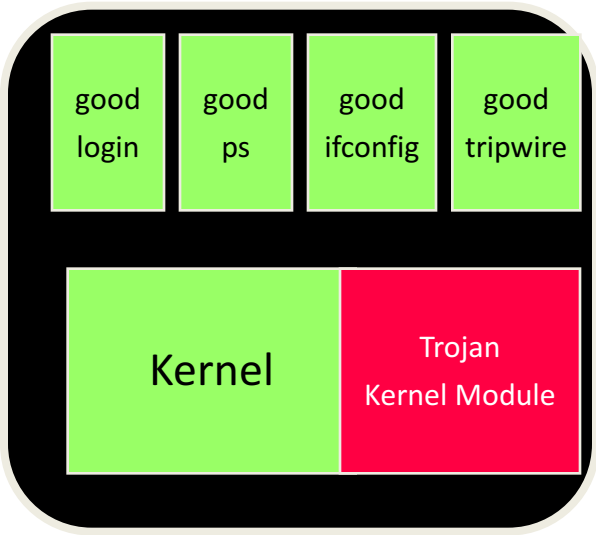
Application-level Rootkit




Traditional RootKit

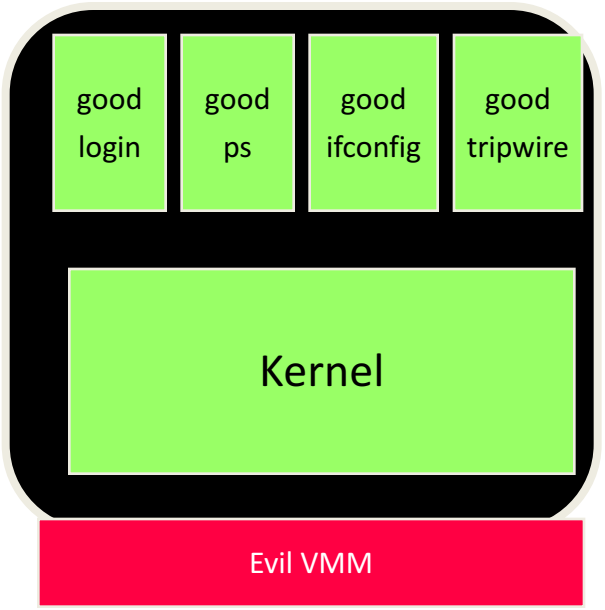



Kernel-level RootKit



# Rootkit Classification

Under-Kernel RootKit




"Blue Pill"

# Spyware

- **Spyware:** software designed to intercept or take partial control over the user's interaction with the computer, without the user's informed consent
  - secretly monitors the user's behavior
  - collect various types of personal information

# Spyware examples

- Log keystrokes
  - Looking for access credentials to bank or brokerage accounts
- Collect web history
- Scan documents on hard disk
  - Looking for SSN, DoB, and other private or confidential information

# Ransomeware

- Fastest-growing threat
- Encrypts file contents, possibly after a delay
- Requires ransom, often paid in bitcoin
  - Key may or may not be provided
  - Often has a deadline

- Countered by good backups so long as they aren’t encrypted too!

# Other unwelcome software issues

- Programs that contain a salami attack
- Web bugs
- Vulnerability to privilege escalation
  - (Mis)use of path
- Programs that carry out interface illusion
- Programs that contain a man in the middle attack
- Programs that contain covert channels

# Who writes malware?

- Employees, ex-employees
- Criminals intent on payout
- Spies
- Extortionists
- Radical activists
- …

# Who writes malware? (cont'd)

- Research by Sarah Gordon (over a decade ago)
  - Overwhelmingly, men
  - Average age ~ 20
  - Not a homogeneous group
    - Adolescent, college, adult, …
  - No "enemy"
    - Except for adult writers: "society"

# Virus writer demographics

- Effects of legal intervention
  - Inconclusive
  - Limited effect (only on specific segments)
  - Likely backlash against law viewed to limit free speech
    - “more likely to write a virus if …”
    - Dangers of unenforceable laws

# Malware detection

- Theoretically: Undecidable
  - … whether by appearance or by behavior
- Practically: Done every day
  - Detection by appearance (getting harder)
  - Detection by behavior (expensive)
- False positives/negatives
  - Almost no false positives
- Ghost positives

# Change detection

- Change in executables
  - Length
  - Content
  - Date/time in the directory listing
- Unaccounted use of resources (esp. memory)
- Unusual hardware behavior
  - Longer disk activity

# Detection (cont'd)

- Use updated anti-virus program
- Types of anti-virus packages
  - Activity monitors
    - Look for virus-like activity (e.g., write to executable, …)
  - Scanners
    - Look for known viruses
    - Include virus-removers

# Detection (cont'd)

- Types of anti-virus packages (cont' d)
  - Authentication or change-detection
    - Compute/store crypto hashes
    - Later, compute and compare with stored
    - Can catch unknown viruses, also disinfect
- Caution if using 2 scanners
  - OK if scan strings encoded in memory
  - Otherwise false positives (one of the scanners appears "infected" to the other)

# Detection (cont'd)

- Virus-checking gateways
  - Scan incoming and outgoing
    - E-mail attachments
    - Transferred files
    - Problems …
      - Unusual formats, encrypted file, …
- On-access scanning
  - As important as perimeter

# Cleanup

- Use disinfecting programs
  - Usually enough
- Restore from clean backup
  - Safer, but expensive
- Not necessary to format HD
  - Leaves partition sector (and any virus in it) untouched

# Prevention

- Keep your software up to date
  - Promptly (patch distribution problem)
- Use only clean software
  - Signed code is not necessarily safe (can contain malware, or exploitable vulnerabilities)
- If you have to take risks …
  - Do so with least privilege (limits damage)
  - Keep good backups
- File protections
  - File attributes do not protect executables

# Prevention

- Whitelisting
  - Only allow what is known good
- Blacklisting
  - Prohibit known bad

Both methods have flaws – Type I and Type II errors