Malicious Software

• There are many types of security problems in software
  – often such holes are exploited by malicious software or malware

• There are many types of malware
  – backdoors
  – logic bombs
  – Trojan horses
  – viruses
  – worms
  – bots
  – rootkits
  – …
• Taxonomy of malicious software

Malicious Software

- Malicious software
  - Need host program
    - Trapdoors
    - Logic bombs
    - Trojan horses
  - Independent
    - Viruses
    - Worms
    - Zombies
    - Rootkits

Replicate
Malicious Software

• Another way to classify malicious software

  – Propagation
    • infected content: viruses
    • vulnerable exploit: worms
    • social engineering: spam email, trojans

  – Payload
    • system corruption: ransomware, logic bomb
    • attack agent: zombie, bots
    • information theft: keyloggers, phishing, spyware
    • stealthing: backdoors, rootkits
Malicious Software Types

- **Trapdoor (or backdoor)**
  - a secret point entry into a program
  - it allows one who knows of the trapdoor existence to get around the normal security access procedures and gain access

- **Trapdoors were commonly used by developers to debug and test programs**
  - the program recognizes a special user ID or a combination of events and grants the user special privileges
  - programmers can easily gain access to the program for debugging purposes or in case something goes wrong

- **Trapdoors have been used to gain unauthorized access to systems**
• Logic bomb

  – code embedded in a legitimate program
  – the code is set to activate when certain conditions are met
  – example conditions
    • presence or absence of particular files
    • particular date or time
    • particular user
  – when a logic bomb is triggered, it typically damages the system
    • modify/delete data, files, or even disks
    • cause the system to halt
Malicious Software Types

- Trojan horse
  - a program with overt (expected) and covert function
    - the overt functionality appears normal and useful
    - when invoked, covert functionality violates security policy
  - user is tricked into executing Trojan horse
    - user sees overt behavior
    - covert function is performed with user’s privileges

- Examples of Trojan horses
  - accomplishing a task that an unauthorized user could not perform
    - Trojan directory listing program `ls` lists files and makes them world readable
Malicious Software Types

• Examples of Trojan horses
  – accomplishing a task an authorized user could not perform
    • login program stores passwords and sends them to a specific address
    • compiler inserts extra code into programs
  – performing data destruction
    • listing directory contents and then removing the files
    • reporting the weather and quietly deleting files

• Covert functionality can be related or unrelated to the overt functionality
Malicious Software Types

- **Viruses**
  - a self-replicating code that attaches itself to a host program
  - the virus contained in an “infected” program will have the ability to infect other programs
  - there is no overt action, it generally tries to remain undetected

- A virus is activated when the host program is executed
  - often the virus attaches itself in the beginning of the program
  - i.e., first virus code is executed and then the original program is run
Viruses

• A virus contains an **infection mechanism** and **payload**
  - the infection mechanism is code responsible for virus replication
  - the payload is other functionality the virus has, including any damage and benign activity

• Example operation of an infected program

  ```
  if (spread condition) then
      for target files
          if not infected, then alter to include virus
      if (activate payload) then
          perform malicious action (payload)
  execute the host program
  ```
• **Virus lifetime phases**
  - the virus can be *dormant* while the spread condition is false
  - then it enters the *propagation phase* and infects other programs or system areas
  - when the payload is *activated*, it performs its main function
  - propagation and execution phases can be activated based on any event
    - date, system utilization, presence/absence of some object, etc.

• Often virus’s code starts with a specific label that indicates that a program has already been infected
  - the virus checks for presence of this label before infecting
Viruses can be classified in many different ways

Virus types based on the target of infection

- boot sector viruses
  - how do we ensure that virus carrier get executed?
  - solution: place the code in boot sector of disk
  - the code is run on each boot and propagates by altering boot disk creation

- executable infectors
  - malicious code is placed at beginning of a legitimate program
  - the code is run when the program is executed, followed by the normal program execution
• Virus types based on the target (cont.)
  – macro viruses
    • non-executable files with macro code are infected
    • the code is interpreted by the application that opens the file
    • example: Microsoft Office documents that can carry macros

• There is a constant battle between virus writers and antivirus software writers
  – both viruses and antivirus software are getting increasingly sophisticated

• Viruses can employ a number of strategies to conceal their presence
• **Compression**
  
  – goal: avoid detection based on increased length of the host program

  – solution: store main program in compressed form

  • when the virus is added to the program, the rest of it is compressed

  • when the program is executed, the virus code uncompresses the program and runs it
**Viruses**

- **Encryption**
  - part of the virus creates an encryption key and encrypts the rest of the virus with it
  - the key is stored with the virus and is used to decrypt it when necessary
  - each copy of the virus uses a different randomly chosen key
  - virus scanners no longer can look for a specific signature

- **Polymorphism**
  - the virus mutates with each infection by introducing differences in the virus code
  - virus scanners cannot look for a specific signature
Viruses

- **Virus evolution**
  - boot sector and executables
    - early systems had poor access control protection mechanisms
  - macro viruses
    - became prevalent in 1990s
    - now MS Office applications have greater protection
  - email viruses
    - prevalent today and allow for faster spreading speeds
    - email virus sends infected contents to all email addresses found on the infected machine
    - first opening infected attachment was necessary to get infected, now simply opening the email could be sufficient
Types of antivirus software

- first generation: simple scanners
  - the simplest technique is to identify a virus “signature”
  - antivirus software then searches for this specific bit pattern
- heuristic scanners
  - identify common behavior of a virus
  - look for traces of such behavior
  - examples: viruses that use encryption, integrity checking of executables
- activity monitors
  - identify a set of actions that indicate that infection is attempted
Viruses

• Types of antivirus software
  – activity monitors (cont.)
    • intervene when such actions are performed
  – combination of the above techniques
  – advanced detection through program simulation
    • an executable file is run on a CPU emulator in controlled environment
    • code scanning is performed to detect a virus (which could be stored encrypted, but is decrypted during execution)

• Antivirus software can have the ability to communicate information about new viruses to a central server
  – allows for timely dissemination of new information to all clients
Worms

- **Worm**
  - a program that self-replicates, but runs independently
  - it propagates by copying itself to other machines through network connection
  - like viruses, it carries a payload for performing hidden tasks
    - e.g., backdoors, spam relays, DDoS agents, etc.

- **A worm can use any network-based mechanism for propagation**
  - e.g., through email, remote exploits, remote logins
  - often a worm is programmed to use more than one propagation method
Worms

• Worm lifetime has similar phases to that of a virus
  – probing: search for potential hosts to infect by inspecting host tables and other files
  – exploitation: find a way to gain access to a remote host
  – replication: copy itself to the remote host and cause it to run
  – payload execution: payload can be executed immediately or triggered by some event

• The first well-known worm is Morris worm which was released in 1988

• Many other large-scale worms appeared afterwards
Worms

• Cost of worm attacks
  – Morris worm (1988)
    • infected approx. 6,000 machines (10% of computers connected to the internet)
    • cost approx. $10 million in downtime and cleanup
  – Code Red worm (2001)
    • infected more than 500,000 servers
    • caused approx. $2.6 billion in damages
    • cost approx. $8.75 billion
• **Morris worm** (1988) – first major attack
  
  – exploited Unix security vulnerabilities, as well as tried password cracking
  
  – no immediate damage from the program itself
    
    • most of the code was to ensure spread of the worm (find other machines, attempt to gain access)
    
    • another part was to copy the worm, compile, and activate on a new machine
  
  – replication and threat of damage
    
    • load on network and systems used in attack
    
    • many systems shut down to prevent further attack
Morris worm propagation mechanisms

- buffer overflow problem in fingerd (Unix finger daemon)
  - fingerd is written in C and runs continuously
  - the worm exploited fgets through a buffer boundary attack
  - somehow this was the most successful propagation mechanism
- trapdoor in the debug option of sendmail (e-mail distribution program)
  - this option allowed the worm to obtain shell access
- remote logins through rsh
  - trusted logins found in .rhosts
  - cracking of weak passwords (using /etc/passwd and its own database of about 400 common passwords)
• More on Morris worm
  – the program was called ‘sh’ to remain undetected
  – the program opens its files and unlinks (deletes) them so that they cannot be found
  – it tried to infect as many hosts as possible
    • when worm successfully connects, it forks a child to continue infection while the parent process keeps trying other hosts
  – the worm did not modify or delete existing files, install Trojan horses, capture superuser privileges, etc.
  – the author was quickly found and charged
  – system admins were busy for several days
    • machines got reinfected and overloaded
Worms

• Lessons learned from Morris worm?
  – security vulnerabilities come from system flaws
  – diversity is useful for resisting attack
  – “experiments” can be dangerous

• More resources
    http://www.ee.ryerson.ca/~elf/hack/iworm.html
Worms

- Code Red
  - initial version was released on July 13, 2001
  - the worm sends its code as an HTTP requests and exploits buffer overflow in MS Internet Information Server (IIS)
  - when it is executed
    - first checks whether the machine is already infected
    - 1st–20th of each month: spreads via random scan of 32-bit IP address space
    - 20th–end of each month: performs a flooding attack against 198.137.240.91 (www.whitehouse.gov)
  - 1st release had an error in seeding PRNG and had a linear growth
  - 2nd release (July 19, 2001) infected almost 360K servers in 14 hours
Wittig worm
- released on March 19, 2004
- exploited buffer overflow in firewall (ISS) products
- first widely propagated worm with destructive payload
  - corrupted hard disk
- shortest interval between vulnerability disclosure and worm release
  - 1 day
- demonstrated that security devices can open doors to attacks
  - other examples include antivirus software and IDS
Worms

- **Stuxnet worm**
  - was detected in 2010, but has been spreading quietly for some time
  - deliberately restricted the rate to reduce detection
  - targeted industrial control systems (Iranian nuclear program)
  - supported a range of propagation mechanisms

- **WannaCry ransomware**
  - released in May 2017
  - quickly spread infecting hundreds of thousands of machines
  - organizations in 150 countries were affected
  - encrypted files and demanded payment
Worms

- How do worms propagate?
  - scanning worms
    - worm chooses random address
    - model propagation as infectious epidemic
  - coordinated scanning
    - different worm instances scan different addresses
  - flash worms
    - assemble tree of vulnerable hosts in advance, propagate along tree
  - topological worms
    - use information from infected hosts (web server logs, email address books, config files, SSH known hosts, etc.)
• **Code red**: theory meets practice (based on number of scans)

![Probes Recorded During Code Red's Reoutbreak](image)

**X-axis**: Hour of the day

**Y-axis**: Number Seen in an hour

- **Blue line**: # of scans
- **Red line**: Predicted # of scans

*Data from CAIDA*
Worm Propagation

- **SQL Slammer** (2003): infects 90% vulnerable machines in 10 minutes, reaches network saturation

---

**DShield Probe Data**

- Probes in 2 second bucket vs. Seconds after 5am UTC
- **Equation:** $K=6.7/m, T=1808.7s, Peak=2050, Const. 28$

(Data from CAIDA)
Worms

- For more information, see
  - “How to Own the Internet in Your Spare Time”, USENIX Security Symposium, 2002

- Challenges in defending against worms
  - small interval between vulnerability disclosure and worm release
    - Witty worm: 1 day; zero-day exploits
  - ultrafast spreading
    - Slammer: 10 minutes, Flashworm: seconds
  - large scale
    - Slammer: 75,000 machines, Code Red: 500,000 machines
• **Need for automation**
  
  – current threats can spread faster than defenses can react
  
  – manual capture/analysis/signature generation/rollout model is slow

• **Worm detection and defense by traffic monitoring**
  
  – observe all traffic between your network and the internet
  
  – **approach 1**: apply throttling/rate limiting
    
    • detect superspreaders by finding hosts that make failed connected attempts to too many other hosts
    
    • limit the number of connections and/or number of hosts scanned
**Worms**

- **Worm detection and defense by traffic monitoring** (cont.)
  - **approach 2:** identify worm patterns
    - look for strings common to traffic with worm-like behavior in monitored traffic
    - signature-based approach
    - content-sifting by detecting the same bitstring pattern
      - main observation: strings of (say) 40 bytes repeat rarely in normally generated traffic
      - disadvantages: large computation and memory requirements, false positives and negatives

- **Worm defenses can also be semantic-based**
  - focus on the root cause (vulnerability); detect exploits, diagnose, generate antibodies
Botnets

• Bot (or zombie)
  – a program that secretly takes over another networked computer by exploiting software flaws
  – it uses that machine to launch attacks that don’t trace back to the creator of the bot
  – each infected machine receives and executes remote commands

• Worm vs. bot
  – a worm propagates itself and executes itself
  – a bot is controlled by a central server (or servers)

• A collection of bots often acts in a coordinated manner and forms a botnet
- Construction and operation of botnets (steps 1 and 2)

1. Attacker scans internet for unsecured systems that can be compromised

2. Attacker secretly installs zombie agent programs on compromised computers
• Construction and operation of botnets (steps 3 and 4)

3. Zombie agents connect to a master server

4. Attacker sends command to launch DDoS attack against a target system
Botnets

• Construction and operation of botnets (steps 5 and 6)

5. Master server sends signal to launch attack on target system

6. Target system is overwhelmed and denies requests from normal users
Botnets

- **How bots are used**
  - launch attacks that are hard to trace to the originator
    - DDoS
    - phishing, spamming
    - traffic sniffing or keylogging, stealing data
    - spreading new malware

- **IRC servers were popular as the master server**
  - bots join a specific chat channel and wait for commands
  - distributed control mechanisms can be used to minimize failure

- **The main objective in defending against botnets** is to detect and disable it at construction phase
Rootkits

- **Rootkit** is software used on a compromised machine to maintain superuser access
  - it is used to hide attacker’s presence
  - it also provides a reentry mechanism into the system

- Since attacker has full access to the system, a rootkit might
  - add/change programs, files, and system utilities
  - monitor processes and network traffic
  - modify the kernel
  - install backdoors for reentry
  - carry any type of malicious payload
Rootkits

- **Types of rootkits**
  - *user mode*
    - modifies results returned by various programs to hide its presence
  - *kernel mode*
    - patches the kernel to modify results returned by native APIs and/or hide some running processes

- Rootkits can also be **persistent** (survive reboot) or **memory-based**
  - persistent rootkit stores code in a persistent store and finds a way to execute it after reboot

- Rootkits are independent of the way of gaining root privileges
  - any mechanism suffices (Trojan horse program, password guessing, system vulnerability, malware, etc.)
• **Reentry** can be performed through any mechanism that works
  – modified login program, accepting connections on a specific port, etc.

• Rootkit’s **payload** can include running sniffers, mounting attacks, compromising other machines, etc.

• **Rootkits are often difficult to detect**
  – since we cannot rely on system’s tools for rootkit detection, other mechanisms must be used
  – can combine network-based monitoring with host-based view
  – the only reliable way to recover from a kernel-based rootkit is to reinstall the OS
Conclusions

- A large number of malicious software types exist
  - Trojan horses, viruses, worms, bots, keyloggers, etc.

- Malware results in large losses

- Malware evolves as better countermeasures become available

- Effective defenses often require substantial efforts and must adopt to constantly changing malware techniques