CS165 – Computer Security

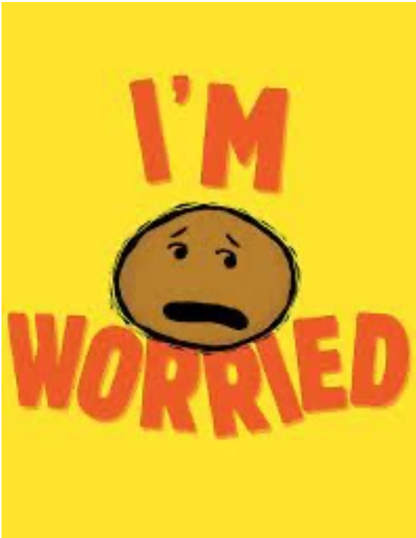
History of Software Attacks

January 16, 2024

Attacks!

- Even in the early days of computing, people were worried about attacks on computer systems

- Why were they concerned?



Early Concerns

- Significant early (1960s) computer systems were funded for government use
  - From single-user systems to timesharing, multi-user systems
  - Leakage of secrets was critical to the Allies success in World War II – and the top concern in the Cold War
- So, when the US funded the development of a general purpose, multi-user operating system
  - Considered security issues as a first-class concept

Multics Project

- Major operating systems research project
  - Information about the project is available online
  - https://multicians.org/history.html

Multics Project

- Participants: MIT, Bell Labs, General Electric
  - Bell Labs dropped out in 1969
    - Later did a system you may be familiar with…
  - General Electric sold out to Honeywell in 1970
- Started in 1965 and funded by the US government (DARPA) for over $2M per year at the time
  - Delivered systems to US Air Force
  - Later sold to various governments and to auto makers, universities, and commercial data processing services
  - Last Multics system was shut down in 2000 (Canada)

Multics Project

- Why are we discussing a system that is no longer in use?
  - And only sold 80 installations
  - But, at about $7M each

Multics Security

- Due to the interest in government deployments, security was a key goal of the Multics project from the outset
- They were concerned about two main problems
  - Secrecy
    - Prevent the unauthorized access to sensitive data
  - Integrity
    - Prevent the illicit modification of sensitive data
- Multics researchers already had a good idea about the software security problems we would face

Process Compromise

- Can an adversary provide an input payload that enables the adversary to hijack your program?
  - Multics researchers knew this was possible in theory
  - And demonstrated such attacks were possible in a vulnerability analysis of Multics in 1974
    - See retrospective in https://www.acsac.org/2002/papers/classic-multics-orig.pdf
    - Among other attacks

- Does this attack violate secrecy or integrity?

Security in Theory

- How can you ensure that your program is secure?
  - I.e., prevent process compromise

Security in Theory

- How can you ensure that your program is secure?
  - No adversary can provide an input to your program

- Works but is often not practical
  - Why not?

Program Input

- How does your program receive inputs?

System Calls Receive Input

- How does your program receive inputs?
  - System calls
    - Open and read a file
    - Open and receive packets on a socket
    - Open a pipe and receive input
    - Open a shared memory region with another process
    - Etc.
- How can an adversary impact these inputs?

At-Risk System Calls

- Which system calls does your program make that are at risk of receiving adversary input?

System Calls and Resources

- Which system calls does your program make that are at risk of receiving adversary input?
  - Ones that may receive input that can be modified by an adversary
  - Suppose there are three system calls and three resources:
    - Files A and B
    - Socket C
  - Suppose an adversary can modify File B and the send packets to Socket C
    - Which system calls are at risk?

Depends

- Suppose there are three system calls and three resources:
  - Files A and B
  - Socket C
- Suppose an adversary can modify File B and the send packets to Socket C
  - Which system calls are at risk?
- Kind of a trick question – depends on which system calls are used to access adversary-modified resources

Attack Surface

- Key term: Attack surface
- An attack surface is the set of system calls your program makes that may access adversary-controlled resources
  - I.e., receive adversary input
- You will need to protect your program's attack surface
  - More to come

Morris Worm

- Robert Morris, a 23-year-old Cornell PhD student
  - Wrote a small (99 line) program
  - Launched on November 3, 1988
  - Simply disabled the Internet
- Used a buffer overflow in a program called *fingerd*
  - To get adversary-controlled code running
- Then spread to other hosts – cracked passwords and leveraged open LAN configurations
- Covered its tracks in a variety of ways

Morris Worm

- Fingerd
  - A UNIX program you can use to determine who is logged into a computer
  - Send a network request to the daemon, which responds with who is logged in and some other metadata
  - I used this program to see if other students or my advisor were online in grad school
- The fingerd program was known to have a flaw that permitted an input payload to hijack execution
  - We'll learn this cause and its prevention later

Morris Worm

- Hijack Fingerd
  - Caused to act as a malicious program that came to be called a "computer worm"
  - The computer worm hijacks the fingerd process
    - Runs code chosen by the worm writer instead of fingerd
    - To download other malicious programs to propagate the attack to other computers in the same network (easy then)
    - And then to other networks
- Computer worm: a malware program that replicates itself to spread to multiple computers

Morris Worm

- Hijack Fingerd
  - Besides the worm behaviors, the Morris worm used multiple techniques to evade identification and ensure that its propagation was not thwarted
    - These techniques worked too well for the time
  - Change the name of the processes created by a hijacked fingerd to “sh”, avoid creating accurate “cores”
  - Tried to propagate to the same computer multiple times
- Basically, created an Internet-scale denial-of-service attack because many computers were running many copies of the Morris worm simultaneously

Morris Worm

- Other than stealing CPU cycles galore,
  - The Morris Worm did not perform any operations that stole data or modified existing data on a compromised host
    - I.e., did not attack the secrecy and integrity of host data
    - Although it certainly impacted the integrity of the fingerd process
- Nonetheless, Morris faced punishments in the forms of fines and prohibitions on computer use for a time period

Morris Worm Reaction

- It was Morris's fault
  - Hands were rung, Morris was punished, few tangible security changes happened in commercial systems
    - Exceptions: Network security research
  - And computer systems took more risks
    - E.g., executable email attachments

The Internet

- Then, the Internet "happened"
  - Actually, the World Wide Web took over in 1995 or so
- Everyone is (well, many people are) connected
  - Not everyone is nice
- It didn't take too long for new attacks like the Morris worm to emerge
  - But, these truly had malicious intent

Code Red

- Worm from 2001
  - Attacked the Windows IIS web server
  - Exploited a publicly known vulnerability
    - A patch had been available a month before
- Same type of vulnerability as the Morris worm
  - Called a buffer overflow
- Malicious activities
  - Defaced websites and launched a DDoS against several IPs, including the White House
- Code Red II later used the same vulnerability

SQL Slammer

- Worm from 2003
  - Attacked the Windows SQL server (database server)
  - Compromised approximately 75,000 hosts worldwide
    - In about 10 minutes
  - Also, exploited a publicly known vulnerability
    - A patch had been available for six months
- Also used a buffer overflow
- Malicious activities
  - None really – impact was mainly a denial of service
    - And concern that a bad actor could "own" all Internet hosts

Worm Reactions

- Problem: known vulnerabilities are exploited on unpatched machines
  - Firewall and antivirus rules target such information
- Problem: one vulnerability enables an adversary to control a host completely
  - Reduce the permissions of network-facing daemons, e.g., no longer run as “root” or “admin”
- Problem: buffer overflow allows an adversary to “inject” their code into a compromised process
  - Prevent executing data on the stack and randomize memory locations of variables and code

Take Away

- The history of software attacks is rather complex
- Early systems designers were aware of the importance of preventing software attacks (Multics)
  - Knew about attacks that were possible
  - Knew eliminating attack surfaces would prevent attacks
- The first attacks “in-the-wild” were worm attacks
  - Exploit the network attack surface
  - Defenses were proposed to protect the network attack surface – more later
- We have been in reactive mode ever since

Questions