

Security

CS 256/456

Dept. of Computer Science, University of Rochester

Journaling File System

- File system operations are not atomic; a sudden machine crash may leave the file system in an inconsistent state
 - consistency checking takes too long
- Journaling file system: Ext3, ReiserFS
 - maintain a dedicated journal that logs all operations
 - the logging happens before the real operation
 - each logging is made to be atomic
 - after the completion of an operation, its entry is removed from the journal
 - at the recovery time, only journal entries need to be examined ⇒ fast recovery
 - performance impact?

Log-Structured File Systems

- With CPUs faster, memory larger
 - buffer caches can also be larger
 - most of read requests can come from the memory cache
 - thus, most disk accesses will be writes
 - poor disk performance when most writes are small
- LFS Strategy [Rosenblum & Ousterhout SOSP1991]
 - structures entire disk as a log
 - always write to the end of the disk log
 - when updates are needed, simply add new copies with updated content; old copies of the blocks are still in the earlier portion of the log
 - periodically purge out useless blocks

The Security Environment

Security goals:

- Data confidentiality
- Data integrity
- System availability

Threats of intruders or adversaries:

- Exposing data
- Tampering with data
- Denial of service attacks

We focus on OS-related security issues.

Login Spoofing

- Login spoofing
 - A program running by the attacker displays a login screen (like the real one)
 - After a legitimate user types in username and password, it records those, kills itself, and a real login screen is shown
 - The user thinks she typed in a wrong password and tries again, which works
- Countermeasure?
 - Start each login session with a non-user-catchable key combination "Ctl-Alt-Delete"

User Authentication

LOGIN: ken
PASSWORD: FooBar
SUCCESSFUL LOGIN

(a)

LOGIN: carol
INVALID LOGIN NAME
LOGIN:

(b)

LOGIN: carol
PASSWORD: ldunno
INVALID LOGIN
LOGIN:

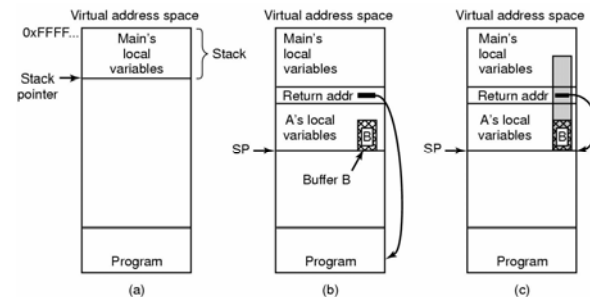
(c)

- (a) A successful login
(b) Login rejected after name entered
(c) Login rejected after name and password typed

User Authentication

- UNIX user passwords are mapped using a one-way function "e()"; and then stored in a globally readable file "/etc/passwd"
 - Bobbie, e(Dog)
 - Tony, e(6%%TaeFF)
 -
- Attacks:
 - used a precomputed common password list
 - exhaustive attack
- Countermeasure?
 - salt
 - Bobbie, 4238, e(Dog4238)
 - Tony, 2918, e(6%%TaeFF2918)

Buffer Overflow



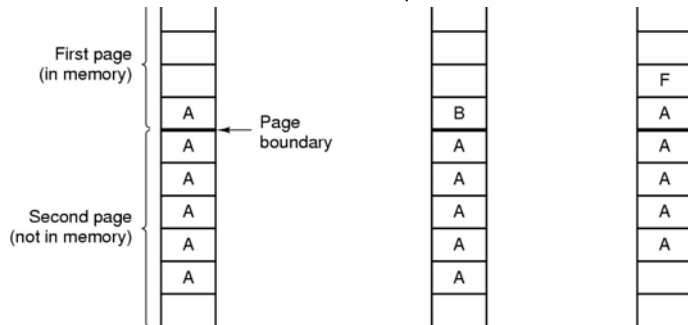
- (a) Situation when main program is running
(b) After program A called
(c) Buffer overflow shown in gray

Countermeasures:

- boundary checks, non-executable stack/data segment, ...

The TENEX Password Problem

Files are accessed with passwords. At each access, the password is checked byte-by-byte and an error is returned as soon as a byte is mismatched.



3/29/2006

CSC 256/456 - Spring 2006

9

Denial-of-service Attack

- Attacker attempts to consume all available resources at the host so no resources are left to serve legitimate users
 - attacks often come from network and they are distributed
- TCP flooding:
 - attackers establish many bogus TCP connections
 - host allocates buffer space for each connection
 - host memory being exhausted eventually
- Countermeasure?
 - **discard** flooded requests: throw out good and bad ones
 - **trace back** to source of floods
 - attack requests with spoofed identities
 - sources are most likely an innocent, compromised machines
 - **delayed processing/resource allocation**
 - stateless TCP [Shieh et al., NSDI2005]

3/29/2006

CSC 256/456 - Spring 2006

10

Virus

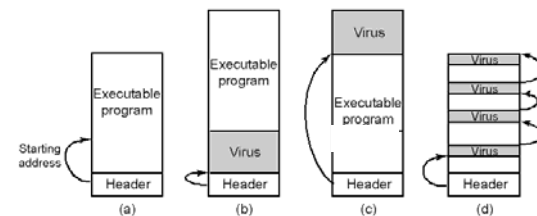
- Virus
 - program can reproduce itself
 - e.g., when invoked, traverse the file system and attach it to randomly selected executables
 - additionally, do harm
 - steal your data
 - temporarily crash the system
 - permanently damage data or hardware
 - denial of service by using all available system resources
- "Good" virus
 - quickly spreading virus
 - difficult to detect
 - hard to get rid of

3/29/2006

CSC 256/456 - Spring 2006

11

Infecting An Executable (Trojan Horses)



- (a) An executable program
- (b) With a virus at the front
- (c) With the virus at the end
- (d) With a virus spread over free space within program

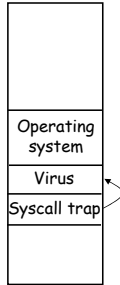
3/29/2006

CSC 256/456 - Spring 2006

12

Memory Resident Viruses

- Virus resides in memory; intercepting system calls
 - known unused memory in OS kernel
 - make the OS believe the memory that virus uses is "legitimately used"
- Where in memory to put the virus?
 - boot sector viruses
 - device driver viruses
- How to load virus there in the first place?
 - boot sector viruses
 - device driver viruses



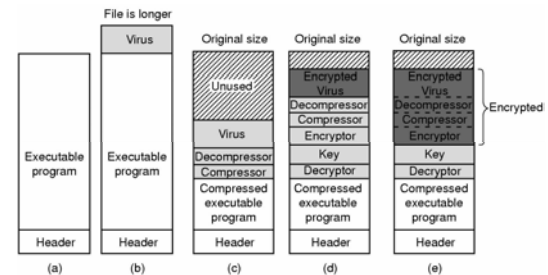
How Viruses Spread

- Try to infect programs on
 - networks: exploiting buffer overflow errors in network server daemons
 - floppy drives
- Attach to innocent looking email
 - when it runs, use mailing list to replicate

Antivirus Techniques

- Size checkers
 - keep a record on the size of disk files and scan them periodically for any size changes
 - apply on readonly executables.
- Signature scanning
 - maintain a database of patterns of common viruses
 - scan disk files for these patterns

Anti-Antivirus Techniques



- (a) A program
- (b) Infected program
- (c) Compressed infected program
- (d) Encrypted virus
- (e) Compressed virus with encrypted compression code

More Antivirus Techniques

- Integrity checkers
 - similar to size checkers, but this time we compute a checksum for file and store them somewhere; we periodically check all files to see whether the checksum still matches
- Behavioral checkers (memory-resident anti-virus program)
 - intercept system calls and detect suspicious activities: overwriting executables, ...

Disclaimer

- Parts of the lecture slides contain original work of Abraham Silberschatz, Peter B. Galvin, Greg Gagne, Andrew S. Tanenbaum, and Gary Nutt. The slides are intended for the sole purpose of instruction of operating systems at the University of Rochester. All copyrighted materials belong to their original owner(s).