Other topics

Chapter 16 -- Security

A big deal for OSes

- Ignoring network security which is really the responsibility of the OS
- Security for the OS -- kinds of attacks
  - breach of confidentiality -- unauthorized reading of data
  - breach of integrity -- modification of data
  - breach of availability -- resource not available
  - theft of service -- unauthorized use of resources
  - denial of service -- fork bombs (minor) ...
- Attack methods
  - masquerading
  - replay attack -- replay of valid data ...
  - message modification
  - man in the middle attack
  - session hijacking
  - privilege escalation

Security (page 2)

- Levels of security
  - physical
  - network
  - operating system
  - application
  - human

Application Level

- Malware and Trojan Horse programs
  - Major problem of "free" programs on Internet
  - not as much for open source programs
- Trap Door
- Logic Bomb
- Stack and Buffer Overflows, Code injection
  - major source of privilege escalation
  - code run on the stack
    - execvp("/bin/sh", ...)

Security (page 3)

- Viruses
  - file
  - boot
  - macro
  - rootkit
  - source code virus
  - polymorphic -- changes signature
  - encrypted
  - stealth
  - tunneling -- interrupt handler/device drivers
  - multipartite -- various locations in the system
  - armored -- hard to figure out what it does.
  - ransomware -- encrypts data, ransom for unlock code

System and Network Threats

- Default install of an OS
  - Many services enabled by default
  - Very few services enabled by default
- worms -- 1988 internet worm, Robert Morris
    - gets() buffer overflow, ...
  - Sobig worm, 2003, photo, target, MS windows
- Port Scanning -- find out what services are available
- Denial of Service -- various forms, network, CPU, ...
  - DDOS -- Distributed denial-of-service attachks

Cryptography as a Security Tool (16.4)

- encryption -- a primary tool for security
  - passwords on UNIX, ...
- Symmetric Encryption: M = D_k (E_k (M))
  - DES -- data-encryption standard, 64 bit value, 56 bit key
    - Triple DES ... 3 keys: E_k3(D_k2(E_k1(M)))
  - AES -- 2001, keys of 128, 192, or 256 bits, 128-bit blocks
  - Not good for long messages ...
- Asymmetric Encryption: RSA, public key/private key systems
- Authentication -- limiting potential senders
  - Also helps prove a message has not been modified
  - md5, SHA-1, other hash functions can be authentication
  - also digitial signatures, RSA allows anyone to verify signature
- Key Distribution
  - Symmetric encryption requires key distribution
  - reason for asymmetric encryption
    - Still can have a man-in-the-middle attack
  - Digitial certificates by a trusted, well known authority
- Implementation of Cryptography
  - Multiple layers -- networking issues here
  - Read 16.4.3 about TLS (Transport Layer Security)

User Authentication (16.5)

How do you know the user is allowed access?

- passwords
  - How to store passwords
  - Easy to guess passwords vs good passwords
  - User or System Generated (X-machine at LLNL)
  - One time passwords and two-factor authentication
  - Challenge / Response systems
- Biometrics
  - fingerprints
    - require both a fingerprint and a password
  - face recognition?
  - ear "print"?
  - other?

Total security policy is typically beyond the OS

- OS can provide tools
- Organization must use tools
- People must have buy-in for a security policy to work
- Must be a "living document"

Security Defenses

Defending from attack, both external and internal

- defense in depth -- many layers of defense are better than few
- Vulnerability Assessment:
  - Risk assessment
    - test scripts vs source code
  - Penetration testing
    - network scans
    - file system scans
    - process scans
  - US Gov ... only as secure as its most far reaching connection
- Intrusion Detection
  - honeypot -- to trap attackers
  - monitoring of system ... has some similarity to penetration testing
- Virus Protection
  - virus scanners
  - sandbox
- Read remainder of chapter (16.6.5-16.8)

Protection (Ch 17)

<< not done, may return later >>

Virtual Machines / Virtualization (Ch 18)

- Basic concept: make one machine appear as many (identical systems)
  - run two (or more) operating systems at the same time
  - want a minor performance penalty
  - have some central, trusted manager

- Past Virtual Machines
  - IBM mainframes in 1972 ...
    - VM370 -- provided several VMs
    - Each VM usually ran a single-user OS
  - Not Quite Virtual Machines: p-machine, JavaVM,

- Current VM like systems
  - Hardware based systems (IBM LPARs, Oracle LDOM)
  - VMMs -- software based VMM control, "below OS"
    - VMware ESX, XenServer, SmartOS, MS HyperV (By windows)
    - All OSes run "on top of" the hypervisor
    - One OS is normally controls the hypervisor

Current VM like systems (page 2)

- Applications that provide a VM for same architecture
  - VirtualBox, VMWare workstation, parallels, qemu-*
  - Virtual environment "inside the application"
  - Code runs at machine speed with hardware assist
  - VM and guest must be same architecture (CPU)
- Paravirtualization -- guests know they are talking to a VM
  - e.g. simplified Disk Driver module ...
- Programming environment virtualization: .net, JavaVM, GNU bc, ...
- Emulators -- run a different machine than the host
  - qemu-*, armware, spike(RISC-V)
  - Can emulate new architectures never built, e.g RISC-V (before it was built)
- Application Containment
  - BSD Jails, Solaris Zones, ...

Benefits of VMs

- Run multiple OSes
- Easy migration
- Cloud computing
- OS experimentation

Basic tools

- Trap and emulate -- "guest" OS executes a privileged instruction
  - Binary Translation ... e.g. some instructions don’t trap
- Special hardware to support virtual machines, fast VM context switches

Read the rest of the chapter for more information

One use of virtualization

isr.cmu.edu -- Internet suspend/resume project (mostly dead now)

- Mobile computing -- cuts "tight binding between PC state and PC hardware"
- Server -- a distributed FS and VM state storage
- Client runs on a VM (minor slowdown is possible)
- Client host has VM & access to network
- User authenticates to VM server, runs a VM on client host
- Client host doesn’t see anything other than encrypted files
- Client OS needs to allow ISR agent access to net and provide a device
- User OS runs on VM.
  - Can suspend VM, save to server
  - Go to different hardware, get state from server, continue
  - ISR agent doesn’t need to load ALL state from server
- Most recent work appears to stop Aug 25, 2015

Distributed vs. Parallel

- Parallel, geographically close, same or highly cooperative OSes
- Distributed, geographically distant, different or low cooperative OSes
- Not hard and fast rules
  - Amoeba -- A single OS that runs on a group of computers on a LAN
    - officially dead.
  - Inferno -- Plan 9 based, last release March 28, 2015
    - Applications in "Limbo" language, target for Dis VM
      - Dis code could easily be translated for host
    - Produces a homogeneous environment accross multiple platforms

Why?

- Resource Sharing
- Computation Speedup
- Reliability
- Communication
  - Multi-site systems
  - Migration from large mainframes -> Network of Workstations

Distributed OS

- Two different world views
  - Each workstation has its own user base ... need to login on every machine
  - One login on distributed system, don’t know which machine you are using
- Issues
  - Data migration
  - Computation migration
  - Process migration
    - Load balancing
    - Computation speedup
    - Hardware preference
    - Software preference
    - Data access / Resource need
  - Kind of network -- LAN vs WAN
    - naming -- somewhat solved by DNS
    - routing -- static vs dynamic
    - Connection: circuit, message, packet ...
  - Not going to do to much networking here ...

Robustness

- Failure Detection
- Reconfiguration
- Recovery from Failure
- Fault Tolerance

Scalability and Transparency

- how does the system scale to larger and larger systems
- does the "user" notice how big the system is?

Hadoop -- Open source map/reduce engine with a distributed FS

- currently part of the Apache family
- designed to run on a cluster of commodity computers

Lustre -- parallel file system, lustre.org.

- Used on many high end HPC systems.

DCE/DFS -- not a full distributed OS, tools for distributed systems

- Remote Procedure Calls, Distributed Objects, Security, Web ...
- DFS essentially AFS with modifications

Distributed File Systems (19.6)

A method to share the same files across a distributed system

- Issues:
  - Model?   client/server vs peer-to-peer
  - Naming of files
    - location transparency
    - location independence
  - File migration
  - Caching, block vs whole file
    - write-through policy
    - consistency
    - client vs server updates to cache
  - Replication, storing files on multiple servers/hosts
    - replication coherence
    - replication updating
  - Semantics (see storage slides): UNIX, session, immutable shared file

Distributed File Systems (page 2)

AFS -- Andrew File System

- C/S model, read replica servers, one write server
- whole file caching, session semantics
- client requires access to at least one server to continue
- Typical Unix Name space:
  - /afs/cs.cmu.edu/user/....

Coda -- Play AFS again with

- All servers R/W, conflicts and resolution
- disconnected operation ... assume cache is correct if can’t contact server
- hoarding in cache

NFS - Network File System

- Not a true DFS
- Used to provide files across a collection of workstations
- Stateless vs. Stateful
  - NFS (original) uses UDP (NFS V4 is stateful, uses TCP)
  - Server going down and back up doesn’t "kill" the connection
- Very little local caching

Distributed Coordination (Not in current Text)

Previous versions of Silberschatz had this in it. Slides from 6e:
http://www.wiley.com/college/silberschatz6e/0471417432/slides/pdf2/mod17.2.pdf

Process Coordination across a distributed system

- On a single machine (even multi-core) it is easy to determine the order of events
- Can get things like locks done "easily" on a distributed system?

Event ordering in a message passing situation

- Happened-Before Relation
  - A and B are events in the same process, A --> B (A before B)
  - A sending, B receiving a message, A --> B
  - Transitive: A --> B, B --> C, then A --> C
- Notice dependence on messages for inter-process ordering.
- Also, A --> A can never hold ... irreflexive, partial ordering

Event Ordering

- Implementation
  - Each process has a "clock"
  - Each "event" increments this clock
  - Each message is tagged with this clock
  - If message M = <D, C> (D = data, C = clock) and C > local C
    - set local C to C+1
  - Total ordering by ordering events in process order if all Cs are the same

Mutual Exclusion in a distributed environment

Centralized Algorithm

- One process becomes the "coordinator"
  - Messages: request, reply, release
  - reply is not sent back until we can assure mutual exclusion
- Coordinator process dies?
  - pick a new one, reconstruct the queue

Mutual Exclusion

Fully Distributed Algorithm

- Much harder!
- Using Ordering from above ...
- Entry on process i
  - sends message request(P_i, TS) to all processes (TS timestamp)
  - A process receives the request sends back a reply
    - if not in a critical section
    - or when it completes its critical section
    - or if waiting, compare TS in request with self request TS
      - send if request TS < self TS
  - When replies from ALL processes are received, start CR