Protection and Security

CS 256/456

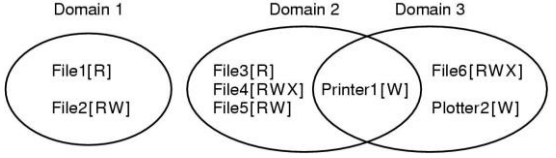
Dept. of Computer Science, University of Rochester

Security and Protection

- Goals:
  - Data confidentiality
  - Data integrity
  - System availability
- Threats:
  - Malicious intruders
  - Accidental data loss

Operating Systems Protection

- Operating system consists of a collection of objects, hardware or software (e.g., files, printers)
- Protection problem - ensure that each object is accessed correctly and only by those processes that are allowed to do so
  - a specific type of security problem
- Domain = set of (object, rights) pairs
  - what is a domain in traditional UNIX?
  - when do process domain switches occur in traditional UNIX?



How to Track Objects/Rights in Domains?

- View protection as a matrix (*protection matrix*)
  - Rows represent domains
  - Columns represent objects
  - *Access(i, j)* is the set of operations that a process executing in $Domain_i$ can invoke on $Object_j$
- Dynamic protection
  - Operations to add, delete access rights

Domain \ Object	File1	File2	File3	File4	File5	File6	Printer1	Plotter2
1	Read	Read Write						
2			Read	Read Write Execute	Read Write		Write	
3						Read Write Execute	Write	Write

Domain Structure

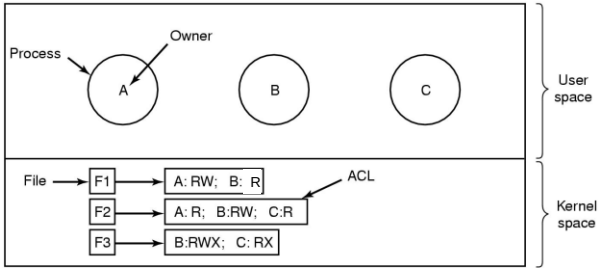
- Principle of protection: least privilege
- Alternatives for domain of protection:
  - Unix: users, groups
  - Multics (1960s): rings of capabilities/protection

Implementation of Protection

- The complete protection matrix consumes too much space and is usually very sparse.
  - two ways to condense it
- Each column = Access-control list for one object
  Defines who can perform what operation.
  Domain 1 = Read, Write
  Domain 2 = Read
  Domain 3 = Read
  ⋮
- Each Row = Capability List for each domain
  Define what operations allowed on what objects.
  Object 1 – Read
  Object 4 – Read, Write, Execute
  Object 5 – Read, Write, Delete, Copy
- Alternative: Lock-Key Mechanism

What does traditional UNIX do?

Access Control Lists



Use of access control lists to manage file access

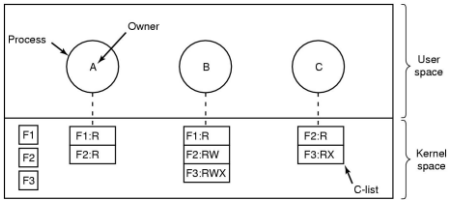
Revocation of access rights?

Capabilities

- Capability for a domain
  - abstraction that indicates access rights to objects for the domain
- Comparing against ACL
  - may be built-in with handle to objects (more efficient access right checking)
  - efficient mechanism to pass it around
- Important concern
  - a process should not be able to tamper with its capabilities

Implementing Capabilities

- Kernel-space capability list
  - user programs use handles (e.g., file descriptors) to refer to them



- Tagged architecture
  - memory words containing capabilities are tagged; user programs can only read those words; only kernel programs can change those words
  - the protection system kernel does not need to track capabilities and capabilities can be passed by memory sharing

Cryptographically-protected Capabilities

- Cryptographically-protected capability
  - a random number (called "check") is generated for each file at creation time and maintained in secrecy
  - capability is formed cryptographically

Server	Object	Rights	f(Object, Rights, Check)

- Function f() is a one-way function such that:
  - it is computationally infeasible to guess what "check" is even when object, rights, and f(object, rights, check) is known.
- Compared with tagged capabilities, cryptographically-protected capability
  - does not require new hardware support
  - can be passed around between distributed machines

Revocation of Access Rights

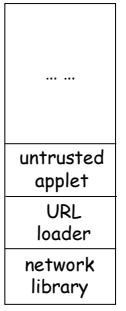
- Immediate or delayed
- *Selective vs. general*
  - *selective:* revoking rights to a select group of domains (e.g., all processes belonging to a particular user)
- Partial vs. total
  - Subset or all rights to object
- Temporary vs. permanent
- Access control lists: immediate, selective or general, total or partial, permanent or temporary
- Capabilities: distributed throughout the system; possible solutions
  - Reacquisition
  - Back-pointers
  - Indirection
  - Keys

Protection in UNIX

- Protection domains: users
- Access matrix for files:
  - a simplified access control list
- Protection commands for files:
  - each user can change protection on files it owns
  - superuser can do everything

Protection in Java

- A Java class can be loaded remotely; therefore can be dangerous
  - A class is assigned a protection domain when it is loaded by the JVM
  - The protection domain indicates what operations the class can (and cannot) perform
- If a library method is invoked that performs a privileged operation, the stack is inspected for access right violation
- Other techniques
  - Sandboxing
  - interpretation



The Security Environment

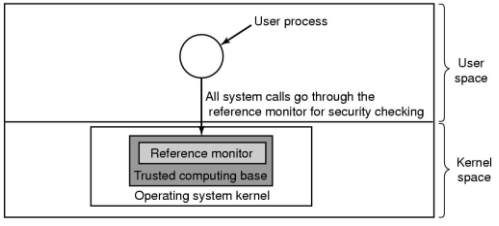
We focus on OS-related security issues …

Security goals:
- Authentication
- Data confidentiality
- Data integrity
- System availability

Threats of intruders or adversaries:
- Identity hijacking
- Exposing data
- Tampering with data
- Denial of service attacks

Trusted Systems

- Trusted Computing Base
  - hardware and software for enforcing security rules
- If the TCB works according to specification, the system security cannot be compromised
- A reference monitor-based TCB:



Formal Models of Secure Systems

Objects

	Compiler	Mailbox 7	Secret
Eric	Read Execute		
Henry	Read Execute	Read Write	
Robert	Read Execute		Read Write

(a)

Objects

	Compiler	Mailbox 7	Secret
Eric	Read Execute		
Henry	Read Execute	Read Write	
Robert	Read Execute	Read	Read Write

(b)

- Protection commands: operations that can change protection matrix
- TCB must ensure that protection commands do not move the protection matrix from an authorized state to an unauthorized state

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 "Ctrl-Alt-Delete"

Leaking Unnecessary Information

LOGIN: ken
PASSWORD: FooBar
SUCCESSFUL LOGIN

(a)

LOGIN: carol
INVALID LOGIN NAME
LOGIN:

(b)

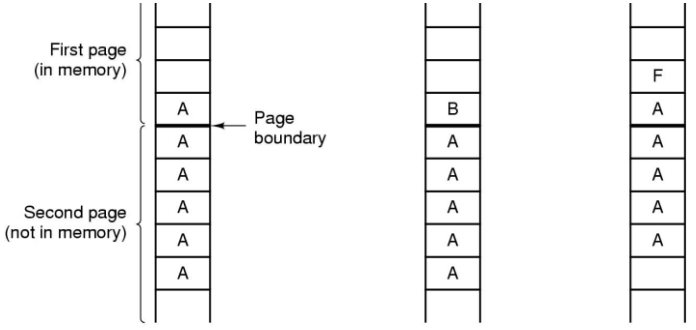
LOGIN: carol
PASSWORD: Idunno
INVALID LOGIN
LOGIN:

(c)

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

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.



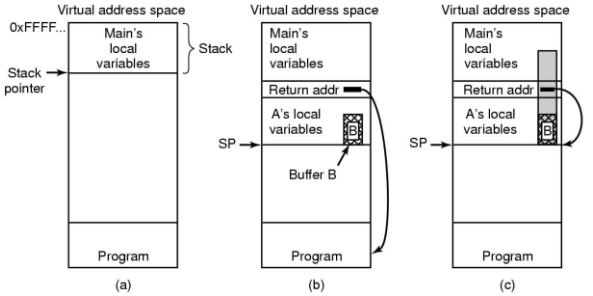
User Authentication

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

Improving Password Security

- One-time passwords
  - Challenge-response authentication
- Authentication using a physical object (e.g., ATM card)
- Authentication using biometrics

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 Morris Worm [1988]

- Three methods of infection
  - rsh from trusted machine
  - Buffer overflow attack in finger
  - Bug in sendmail
- Once a machine was infected, used password cracking for further proliferation
- If a copy of the worm already existed, 1 in 7 exited
  - Downfall since it brought the Internet down

Security Services

- Privacy (prevent others from obtaining information)
- Authentication (verify the identity of an object owner)
- Data integrity (make sure the data is not altered)

→ Cryptography as a Security Tool

Cryptographic Algorithms

- Encryption/decryption algorithms typically public knowledge
- Key shared between sender and receiver is the difference
  - Secret key – symmetric – both participants share a single key
  - Public key (e.g., RSA)
    - Public key published for all to know
    - Private key that is shared with no one
  - Hash or message digest (e.g., MD5) – no keys
    - Map a potentially large piece of data into a small fixed-length number
    - More efficient to compute than the above techniques

Encryption Algorithm Requirement

- Assumption – only the key is kept secret or you will have to keep re-inventing new encryption algorithms!
- Algorithm must be safe from chosen plain-text attacks (if the attacker knows both plaintext and ciphertext)
  - Make sure none of the structure of plaintext remains

→ Key distribution a central problem in security

DES – Date Encryption Standard

- Secret key algorithm
- Encrypts 64-bit blocks of plaintext using a 64-bit key
- Three distinct phases
  - Permute (shuffle bits in the block)
  - Perform 16 rounds of an identical operation
  - Perform the inverse of the original permutation
- Decryption – apply the same algorithm, except that the keys are applied in reverse

Encrypting Large Data

- Cipher block chaining (CBC) – XOR ciphertext for block i with plaintext for block i+1 before running it through DES
- Use initialization vector in lieu of non-existent ciphertext for block 0
- Use Triple-DES to increase security

RSA (Rivest, Shamir, and Adleman) Algorithm

- Public key for encryption, private key for decryption
- Grounded in number theory
- Encryption/decryption function requires enormous computational power

Key Selection

- Key length of 512 bits
- Choose two large prime numbers p and q, and multiply them together to get n
- Choose encryption key e such that e and (p-1)x(q-1) are relatively prime
- Compute decryption key d such that
- Public key constructed from the pair <e,n> and private key by the pair <d,n>

Authentication: Simple Three-Way Handshake

- Participants already share a secret key
- Client sends encrypted (using client handshake key) random number x along with clientId
- Server decrypts and responds with x+1, along with random number y, both encrypted with server handshake key
- Client validates legitimacy of server by checking return value x+1 and returns encrypted y+1
- Server validates legitimacy of client based on response and then sends encrypted session key

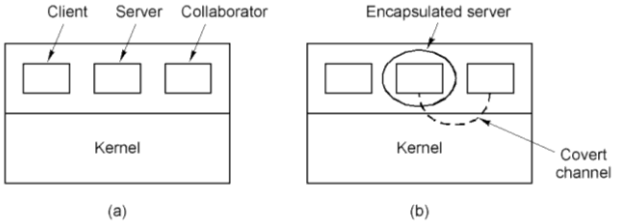
Authentication: Trusted Third Party (e.g., Kerberos)

- The authentication server S is trusted and shares a secret key with A and B, K_A and K_B respectively
- A identifies A and B to S
- S sends back a two-part message, encoding session key K and a timestamp T, using K_A and K_B
- A decodes first part of message and forwards second part to B along with timestamp encoded using K
- B decodes second part to recover K and T and uses K to decode second part and compare
- B replies with T+1 encoded using K

PGP – Pretty Good Privacy

- Provides privacy, authentication, integrity, compression
  - Hash message using MD5
  - Encrypt hash using private RSA key
  - Compress using ZIP
  - Use random key to encrypt message using IDEA (similar to DES)
  - Encrypt random key using receiver's public key

Covert Channels: Information leaking



Client, server, and collaborator processes

Encapsulated server can still leak to collaborator via covert channels

- Covert channels
  - Modulating CPU usage
  - Locking/unlocking files

Information Leaking Through Side Channels

- Side channels
  - performance observations
  - program execution signals (e.g., cache usage, memory bus usage)
- Side channel attack on hyper-threading processors [Percival 2005]
  - OpenSSH running DES encryption on one hyper-thread
  - attacker running on the other hyper-thread
  - attacker and OpenSSH share hardware cache, so attacker can monitor its own cache miss pattern to infer the execution of OpenSSH (and its DES encryption key)

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 is exhausted eventually
- Countermeasure?
  - discard flooded requests: throws out good and bad ones
  - trace back to source of floods
    - sources are most likely an innocent, compromised machines
  - delayed processing/resource allocation
    - stateless TCP [Shieh et al., NSDI2005]

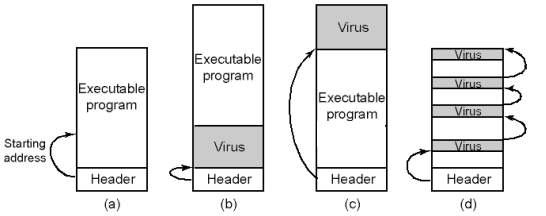
Virus

- Virus (fragment of code embedded in a legitimate program)
  - 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

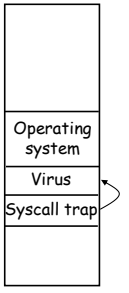
Infecting An Executable (Trojan Horses)



(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

Memory Resident Viruses

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



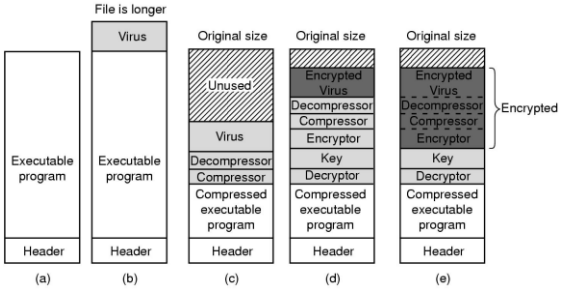
How Viruses Spread

- 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 of the size of disk files and scan them periodically for any size changes
  - apply on read-only 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 by 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).

So far …

- Login and password security
- Exploiting system weaknesses and worms
  - Login spoofing
  - Buffer overflow attack
  - Tenex password problem
- Today
  - Cryptography as a security tool
  - Viruses