Operating systems and security - Overview

- Protection in Operating systems
- Protected objects
- Protecting memory, files
- User authentication, especially passwords
- Trusted operating systems, security kernels, etc.
- Assurance methods
- Evaluation
Operating systems

• An *operating system* is a program that acts as an intermediary between a user of a computer and the computer hardware.

• The purpose is to provide an environment in which a user can execute programs in a *convenient* and *efficient* manner.
Operating systems
Protection in Operating Systems

• Reasons:
  – To prevent mischievous, intentional violation of an access restriction by a user
  – To prevent erroneous programs from affecting the execution of other programs
  – Improve reliability
Protected Objects

- Hardware, software and data
- Memory
- Sharable I/O devices
- Serially reusable I/O devices
- Sharable programs and sub-procedures
- Sharable data
Separation

• The basis of protection is separation
• Several ways:
  – Physical separation
  – Temporal separation
  – Logical separation
  – Cryptographical separation
Sharing

• We also want to provide sharing of objects
• Several levels of protection is possible
  – No protection
  – Isolation
  – Share all or share nothing
  – Share via access limitation
  – Share by capabilities
  – Limit use of an object
Protecting Memory and Addressing

• Needs hardware support
• At least two different modes needs to be available:
  – Privileged, system, kernel, etc..
  – Unprivileged, user, etc..
• Additional support for different schemes, e.g. paging or segmentation.
• Relatively easy to do: every memory access is guaranteed to go through certain points in the hardware.
Fence

Adresses
0
n
n+1

Operating system

Hardware Address limitation

User Program Space
Base/Bound Registers

Diagram:

- Base register
- Bounds register
- Operating system
- User A
- User B

Adresses:
- 0
- n
- n+1
- p
- p+1
- q
Segmentation

- A program is divided into separate pieces, **segments**
- Each segment has a unique name.
- Code or data items in a segment is addressed as the pair `<name, offset>`.
Segmentation

• Benefits:
  – Each address reference is checked for protection.
  – Many different classes of data items can be assigned different levels of protection.
  – Two or more users can share access to a segment.
  – A user cannot generate an address or access to an unpermitted segment.
Paging

- The most commonly used memory management scheme.
- The physical memory is divided into frames.
- Each address have two parts: a page number and offset.
- Each page in the memory is mapped to a frame.
- This mapping is maintained in the page table.
- Each process (program) has its own page table.
- Access rights to the different pages is also kept in the page tables.
- Pages can be shared between different programs.
Protecting Access to General Objects

- Objects that needs protection
  - Memory
  - Files on storage devices
  - A executing program in memory
  - A directory of files
  - A hardware device
  - A data structure, such as a stack
  - Parts of the operating system
  - Instructions
  - Passwords
  - The protection mechanism
Goals and principles

• Check every access
• Allow least privilege
• Verify acceptable usage
Directory

- Works like a file directory.
- Easy to implement: one list per user, naming all the objects the user is allowed to access.
- Sharing is possible: several users can have a pointer (a name) to the same object in their directory.
- Problem: revocation of access (the system must search through all lists for the object).
- Too simple for most object protection situations.
Access Control List

• One list for each object.
• The list shows all subjects who should have access to the object and what their access is.
• Groups can also be used as subjects.
• Often used, e.g. in Windows NT.
Access Control Matrix

• A table of subject, objects and access rights.
• Inefficient and seldom used.
A **capability** is an unforgeable token giving the possessor certain rights to an object. Used together with e.g. access control lists. The capabilities must be stored in memory inaccessible to the subjects. Capabilities can be revoked. Kerberos (more about Kerberos later) is an example that uses capabilities.
Procedure-Oriented Access Control

- A procedure controls access to objects.
- Implements the principle of information hiding: only the control procedure need to know how an object is implemented.
- Can be inefficient.
File Protection Mechanisms

- Examples of programs or techniques for file protection:
  - All-None protection
  - Group protection
  - Single permissions
All-None Protection

• The original IBM OS operating system, files were public by default.
• Any user could read, modify and delete any file.
• Certain files could be protected with a password.
• A number of problems:
  – Lack of trust
  – All or nothing
  – Rise of time-sharing
  – Complexity
  – File listings
Group Protection

• Example from Unix, VAX VMS.
• Three classes: user, group and world.
• Users that need to share objects are part of a group.
• A file have different access rights for the user, the group and the rest of the world.
• Problems:
  – Group affiliation: solved by declaring that every user belongs to one group, or by storing group id along with the file.
  – Limited sharing
Single Permissions, Password or other tokens

• A user can assign a password to a file.
• User access is limited to those who can supply the correct password.
• Can give the effect of a different "group" for each file.
• Problems:
  – Lost passwords
  – Disclosure
  – Revocation
  – How to spread the password
Single Permissions, Temporary Acquired Permission

- The **set userid** (setuid) feature in Unix enables temporary acquired permissions:
  - When file with the setuid protection bit set is executed with the protection level of the *owner* of the file, not the *executer*.
  - A program owned by root with the setuid bit runs with root-privileges, no matter who the user that executed the program is.
  - A flaw in a root-owned setuid executable is disastrous.
  - Don't use it if not absolutely necessary.
  - Groups can often be used instead.
User Authentication

• The basis for much of the protection in an operating system is based on user authentication.

• The most common authentication mechanism is a password, a word or phrase known only by the computer and the user.
Attacks on passwords

• Try all possible passwords
• Try many probable passwords
• Try passwords likely for the user
• Search for the system list of passwords
• Install a Trojan horse
• Listen for passwords on the network
• Social engineering: ask the user, impersonate the system administrator, etc..
Exhaustive attack

• In an exhaustive or brute force attack, the attacker tries all possible passwords.
• At a rate of 1 password per microsecond, it would take about 2 months to try all possible passwords of 1-8 characters.
Probable passwords

• A password such as beer or summer is more likely to be used than e3.;%Te.
• Try words from dictionary instead.
• Try the words backwards too.
• People of chooses passwords that means something: name of a pet, birthdays, names of people they know, etc.
Finding a plaintext system password list

- Sometimes the passwords are stored in a (read protected) plaintext file.
- A flaw in a privileged program could be exploited to read the password file.
- Dumping memory at a convenient time could also produce a plaintext list.
- Backups could also be used.


Encrypted password file

• A safer way: encrypt the password file.
• Two ways:
• The passwords are encrypted with conventional encryption:
  – When a user password is received, the stored password is decrypted and compared.
  – Plain text passwords are still available in the memory.
• One-way encryption (hash functions):
  – When a user password is received, the password is encrypted and the cipher text is compared with the stored one.
Encrypted password file

• Example from Unix:
  – Often the password file with encrypted passwords are available to all users.
  – To avoid having two users with the same passwords having the same encrypted password, a salt is added
  – Makes many of the attacks against passwords easier, so the encrypted passwords are often read-protected.
Social engineering

• Easiest way to obtain passwords: ask the user.
• Many people write the passwords somewhere.
Trojan horses

Example:

1. Display copy of the login prompt or screen
2. Read the user name and password when a user tries to login
3. Save the password, and display "password incorrect" and make the real login prompt visible.
Password selection

• Some rules:
  – Use characters other than a-z and A-Z
  – Choose long passwords.
  – Avoid actual names or words.
  – Choose an unlikely password.
  – Change the password regularly.
  – Don't write it down.
  – Don't tell anyone else.

• Passwords that are too hard to remember is also a problem: users are then more likely to write the password somewhere.
One-Time Passwords

- A one-time password is one that changes every time it is used.
- Instead of knowing a static password, each user knows a mathematical function.
- The system provides an argument to the function, and the user computes and returns the function value.
- Also called challenge-response systems.
- An intercepted password is useless.
- Limited use without hardware devices (the functions can be hard to remember and compute).
Other means of authentication

• What you
  – Know - passwords etc.
  – Have - smartcards etc.
  – Are - biometrics

• Hardware devices: finger prints, retinal scans, smart card readers etc.

• Often expensive.

• Can be stolen(?)

• Can be combined with passwords.
Trusted operating systems

• Trusted - believed to be secure to some limit
• A policy is a statement of the security we expect the system to enforce.
• A operating system can be trusted only in relation to a security policy, that is, to the security needs the system is expected to satisfy.
Example - separation of duty

• Separation of duty or separation of responsibility
• A commercial security policy
• Example:
  – In a small company, several people might be authorized to issue orders, receive goods and write checks.
  – You would not want the same person issuing the order, receiving the goods and writing the check since there is potential for abuse.
  – Therefore, you might want a policy that specifies that three separate people issue orders, receive goods and write checks, although any of the three might be authorized to do any of these tasks.
Security features of Trusted operating systems

• User identification and authentication
• Mandatory access control
• Discretionary access control
• Object reuse protection
• Audit
• Audit log reduction
• Trusted path
• Intrusion detection
Security kernels

- A kernel is the part an operating system that performs the lowest-level functions.
- A security kernel is responsible for enforcing the security mechanisms of the entire operating system.
- Reasons for a security kernel:
  - Coverage
  - Separation
  - Unity
  - Modifiability
  - Compactness
  - Verifiability
- Might degrade system performance.
- Does not guarantee that it contains all security functions, or that it has been implemented correctly.
- Can be quite large.
Reference monitor

- The reference monitor controls accesses to objects.
- Can be a collection of access controls for devices, files, etc.
- Must be:
  - Tamper-proof
  - Always invoked
  - Small enough to easily trusted
Trusted computing base

• The trusted computing base (TCB) is everything in the trusted operating system necessary to enforce the security policy.
• Nothing in the non-TCB code should be able to impair the correct security policy enforcement.
• A security kernel and a reference monitor are part of the TCB
Assurance Methods

• Again, many methods from Software Engineering:
  – Testing
  – Formal verification
  – Validation
Testing

- Testing can demonstrate the *existence* of a flaw, but not the *absence* of flaws.
- The many possibilities of e.g. input makes it hard (or even impossible) to create tests with a adequate coverage.
- Penetration or tiger-team analysis: a team of experts try to crack the system.
Formal Verification

• The operating system is reduced to a theorem, which is then proven.
• For a operating system this can take years to perform.
• It is also a very complex process.
Validation

• A more general term than verification.
• Less rigorous methods:
  – Requirement checking
  – Design and code reviews
  – Module and system testing
Evaluation

- Most consumers of operating systems can't verify the security themselves.
- Independent third-party evaluation is thus desirable.
- One example: U.S. Orange Book Evaluation
U.S. Orange Book Evaluation

- Four basic divisions: A, B, C, D
- A is most secure
- Within the divisions, there are added distinctions denoted by numbers (higher is better).
- Complete list, from lowest to highest assurance: D, C1, C2, B1, B2, B3 and A1.
- Most Unix systems can be made to get a C2 certification without much effort.
- One installation of Windows NT3.51 (without any network) has been C2-certified (when people say that NT is C2-certified, this is what they are referring to).
What doesn't lead to assurance

- Emphatic assertion
- Security through obscurity
- “I couldn't find flaws”
- Challenges
Next time

• Examples from real world operating systems