Security I

CS 4410
Operating Systems

[E. Birrell, A. Bracy, F. B. Schneider, E. Sirer, R. Van Renesse]

References: Security Introduction and Access Control by Fred B. Schneider
Secure Systems?

A **secure** system will
  - do what is expected
  - not do the unexpected
despite attacks and offers assurance about this claim.

Example:
  - Do more: reveal secrets
  - Do less: fail to store or retrieve information
Security Properties: CIA

Confidentiality: *keeping secrets*
- who is allowed to learn what information

Integrity: *permitting changes*
- what changes to the system and its environment are allowed

Availability: *guarantee of service*
- service should be “timely”
Protect against what?

Some terminology:

**vulnerability**: Weakness that can be exploited to cause damage.

**attack**: Method of exploiting a vulnerability.

**threat**: Motivated capable adversary who will mount attacks.

All systems have vulnerabilities.

*Understand the threats and defend against attacks they can mount.*

All assumptions are vulnerabilities.
Cyber threats

• Operator/user blunders.
• Hackers driven by intellectual challenge (or boredom).
• Insiders: employees or customers seeking revenge.
• Criminals seeking financial gain.
• Organized crime seeking gain or hiding criminal activities.
• Organized terrorist groups or nation states trying to influence national policy.
• Foreign agents seeking information for economic, political, or military purposes.
• Tactical countermeasures intended to disrupt military capability.
• Large organized terrorist groups or nation-states intent on overthrowing the US government.
Cyber threats: Classification

Class I: Execute existing attacks against known vulnerabilities.

Class II: Analyze system, find new vulnerabilities, develop new attacks.

Class III: Create new vulnerabilities (e.g., compromise the supply chain).
Cyber threats: Classification

Access based:
- Physical access
- Software access
- User access

Capability based:
- Computational (probabilistic polynomial time TM)
Security in the “real world”

Use locks to block attacks: “Prevention”
• Locks must not be annoying, or they won’t be used.
• All locks aren’t the same. They are:
  • Scaled for what they are protecting.
  • Scaled for their environment.

• Police and courts are central---not the locks!
  “Deterrence through accountability”
  • Tracking down the “bad guys” is what’s central.
  • Locks reduce temptation and reduce workload on police and courts.

“Recovery based on audit”
“Real world” (con’t)

• People only pay for security that they think they need.
  … need is based on personal experience & others experiences.

• People avoid annoying locks by buying insurance.
  • Risk avoidance versus risk management.
  • Externalities is a wrinkle!

• Security is holistic…
  • Security is only as strong as the weakest link.
  • Making any link stronger than the weakest link doesn’t much improve security.
Security in Computer Systems

Gold (Au) Standard for Security [Lampson]

**Authorization**: mechanisms that govern whether actions are permitted

**Authentication**: mechanisms that bind *principals* to actions

**Audit**: mechanisms that record and review actions
Security Mechanism Design

What should the mechanism do?

Best to distinguish policy from mechanism.

Desire mechanisms that implement many policies.
Operations: learn and/or update information
Principals: instigate operations
  • users, processes, threads, procedures
Objects of operations: memory, files, modules, services

Access Control Policy:
  • which principals may perform which operations on which objects
  … Enforces confidentiality & integrity
Access Control Mechanisms

Reference Monitor:
• Consulted on each operation invocation
• Allows operation if invoker has required privileges
  - … Can enforce confidentiality and/or integrity

Assumptions:
• Predefined operations are the sole means by which principals learn or update information
• All predefined operations can be monitored (“complete mediation”)
A secure system should have a well defined TCB in order to discharge assurance requirements.

TCB typically contains:

- HW & SW necessary for enforcing security rules
  - most hardware, firmware
  - portion of OS kernel
  - most or all programs with superuser power

- Desirable features of TCB to facilitate analysis:
  - Small size
  - Isolated, so separable
Reference Monitor is in TCB

- All operations intercepted by reference monitor
- Monitor decides: should operation proceed?

… Reference monitor is not a separable module in most OSes…

![Diagram showing the Trusted Computing Base (TCB) with user space and kernel space, with the reference monitor and trusted computing base highlighted.]
Who defines authorizations?

Discretionary Access Control (DAC):
• Object **owner** defines authorizations for operations on that object.
• Supported in all OS (Linux/MacOSX/Windows File Systems)
• Vulnerable to “Trojan Horse” attacks.

Mandatory Access Control (MAC):
• System imposes access control policy that object owners cannot change
• Centralized authority defines authorizations
• Commonly required in institutional settings.
Principle of Least Privilege

“Every program and every privileged user of the system should operate using the least amount of privilege necessary to complete the job.”

- Jerome Saltzer

Implies that we should want to minimize:

- code running inside kernel
- code running as sysadmin
Access Control Matrix

- Abstract model of authorization
- Rows: **principals** = users
- Columns: **objects** = files, I/O, etc.

<table>
<thead>
<tr>
<th>Principals</th>
<th>OBJECTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>prelim.pdf</td>
</tr>
<tr>
<td>rvr (prof)</td>
<td>r, w</td>
</tr>
<tr>
<td>jan (student)</td>
<td></td>
</tr>
</tbody>
</table>

*Unordered set of triples <Principal, Object, Operation>*

What does Principal of Least Privilege say about this?
Fine-Grained Principals

**Protection Domain** = set of principals
- each process belongs to some protection domain at all times
- process transitions from domain to domain

Example domain: user $\triangleright$ task
- task = program, procedure, block of statements
- task = started by user or in response to user’s request
- user $\triangleright$ task: holds minimum privilege to get task done for user

$\rightarrow$ task-specific privileges ($PoLP$ is 😊)
## When to transition protection-domains?

- invoking a program
- changing from user to kernel mode
- ...

Need to explicitly authorize them in the matrix

### Access Matrix with Protection Domains

<table>
<thead>
<tr>
<th>Principals</th>
<th>OBJECTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>prelim.pdf</td>
</tr>
<tr>
<td>rvr▷sh</td>
<td></td>
</tr>
<tr>
<td>rvr▷latex</td>
<td>r, w</td>
</tr>
<tr>
<td>rvr▷excel</td>
<td></td>
</tr>
<tr>
<td>jan▷sh</td>
<td></td>
</tr>
<tr>
<td>jan▷latex</td>
<td>r, w</td>
</tr>
<tr>
<td>jan▷excel</td>
<td></td>
</tr>
</tbody>
</table>
## Access Matrix with Domain Transitions

<table>
<thead>
<tr>
<th>Principals</th>
<th>prelim.pdf</th>
<th>jan-hw.tex</th>
<th>scores.xls</th>
<th>rvr▷sh</th>
<th>rvr▷latex</th>
<th>rvr▷excel</th>
<th>jan▷sh</th>
<th>jan▷latex</th>
<th>jan▷excel</th>
</tr>
</thead>
<tbody>
<tr>
<td>rvr▷sh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>e</td>
<td>e</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rvr▷latex</td>
<td></td>
<td>r, w</td>
<td>r</td>
<td></td>
<td></td>
<td>e</td>
<td>e</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rvr▷excel</td>
<td>r, w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>jan▷sh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>e</td>
<td>e</td>
<td></td>
<td></td>
</tr>
<tr>
<td>jan▷latex</td>
<td></td>
<td>r, w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>jan▷excel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*e = enter*
Protection Domain Transitions

Two possibilities:

1. Certain control flow transitions cause protection-domain transitions. Obvious candidates:
   - invoking a program
   - changing from user mode to supervisor mode

2. Single explicit domain-change syscall
   - application programmer or a compiler then required to decide when to invoke this domain-change system call
DAC Implementation Needs

Must support:

• Determining whether \( <\text{Principal}, \text{Object}, \text{Operation}> \) is in authorization relation.

• Changing the matrix

• Assigning each process a protection domain

• Transitioning between domains, as needed

• Listing each principal’s privileges (for each object)

• Listing each object’s privileges (held by principals)

2D array?

+ looks good in powerpoint!

- sparse \( \rightarrow \) store only the non-empty cells
Implementing the “Matrix”?

Access Control List (ACL): column for each object stored as a list for the object

<table>
<thead>
<tr>
<th>Principals</th>
<th>OBJECTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>prelim.pdf</td>
</tr>
<tr>
<td></td>
<td>jan-hw.tex</td>
</tr>
<tr>
<td></td>
<td>scores.xls</td>
</tr>
<tr>
<td>rvr▷sh</td>
<td>r, w</td>
</tr>
<tr>
<td>rvr▷latex</td>
<td>r</td>
</tr>
<tr>
<td>rvr▷excel</td>
<td>r, w</td>
</tr>
<tr>
<td>jan▷sh</td>
<td>r, w</td>
</tr>
<tr>
<td>jan▷latex</td>
<td>r, w</td>
</tr>
<tr>
<td>jan▷excel</td>
<td></td>
</tr>
</tbody>
</table>
## Implementing the “Matrix”? 

**Access Control List (ACL):** column for each object stored as a list for the object  
**Capabilities:** row for each subject stored as list for the principal

<table>
<thead>
<tr>
<th>Principals</th>
<th>OBJECTS</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>prelim.pdf</td>
<td>jan-hw.tex</td>
<td>scores.xls</td>
<td></td>
</tr>
<tr>
<td>rvr▷sh</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rvr▷latex</td>
<td>r, w</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rvr▷excel</td>
<td></td>
<td></td>
<td></td>
<td>r, w</td>
</tr>
<tr>
<td>jan▷sh</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>jan▷latex</td>
<td></td>
<td></td>
<td>r, w</td>
<td></td>
</tr>
<tr>
<td>jan▷excel</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Same in theory; different in practice!