CSE543 - Introduction to Computer and Network Security

Module: Mandatory Access Control

Professor Trent Jaeger
Access Control and Security

- **Claim**: Traditional access control approaches (UNIX and Windows) do not enforce security against a determined adversary
  - (1) Access control policies do not guarantee secrecy or integrity
  - (2) Protection systems allow untrusted processes to change protection state
- **Mandatory Access Control** (MAC) solves these limitations
  - What is “mandatory”?
  - How do MAC models guarantee security?
Security Goals

- **Secrecy**
  - Don’t allow reading by unauthorized subjects
  - Control where data can be written by authorized subjects
    - Why is this important?

- **Integrity**
  - Don’t allow modification by unauthorized subjects
  - Don’t allow dependence on lower integrity data/code
    - Why is this important?
    - What is “dependence”?

- **Availability**
  - The necessary function must run
  - Doesn’t this conflict with above?
Trusted Processes

• Do you trust every process you run?
Trusted Processes

• Do you trust every process you run?
  ‣ To not be malicious?
Trusted Processes

• Do you trust every process you run?
  ‣ To not be malicious?
  ‣ To not be compromised?
Secrecy

- Does the following protection state ensure the secrecy of J’s private key in $O_1$ (i.e., $S_2$ and $S_3$ cannot read)?

<table>
<thead>
<tr>
<th></th>
<th>$O_1$</th>
<th>$O_2$</th>
<th>$O_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>R</td>
<td>RW</td>
<td>RW</td>
</tr>
<tr>
<td>$S_2$</td>
<td>-</td>
<td>R</td>
<td>RW</td>
</tr>
<tr>
<td>$S_3$</td>
<td>-</td>
<td>R</td>
<td>RW</td>
</tr>
</tbody>
</table>
Secrecy Threat

- Trojan Horse
  - Some process of yours is going to give away your secret data
- Write your photos to the network
Integrity

- Does the following access matrix protect the integrity of J’s public key file $O_2$?

<table>
<thead>
<tr>
<th></th>
<th>$O_1$</th>
<th>$O_2$</th>
<th>$O_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>R</td>
<td>RW</td>
<td>RW</td>
</tr>
<tr>
<td>$S_2$</td>
<td>-</td>
<td>R</td>
<td>RW</td>
</tr>
<tr>
<td>$S_3$</td>
<td>-</td>
<td>R</td>
<td>RW</td>
</tr>
</tbody>
</table>
Integrity Threat

• Unexpected Attack Surface
  ‣ *Process reads untrusted input when expects input protected from adversaries*
  • Read a user-defined config file
  • Execute a log file
  • Admin executes untrusted programs
Protection vs Security

• Protection
  ‣ Secrecy and integrity met under trusted processes
  ‣ Protects against an error by a non-malicious entity

• Security
  ‣ Security goals met under potentially malicious processes
  ‣ Protects against any malicious entity

• Hence, For J:
  ‣ Non-malicious process shouldn’t leak the private key by writing it to $O_3$
  ‣ A potentially malicious process may contain a Trojan horse that can write the private key to $O_3$
Types of Security Goals

• *In practice, goals focus on security or availability (function)*
  ‣ *but not both*

• **Security Goals** (Secrecy and Integrity)
  ‣ Advantage: Focus is security
  ‣ Disadvantage: May prevent required functionality

• **Functional Goals** (Availability)
  ‣ Advantage: Enables required functionality
  ‣ Disadvantage: May not block all attack paths

• Let’s look at some common goals
  ‣ Least Privilege and Information Flow
Principle of Least Privilege

A system should only provide those privileges needed to perform the processes’ functions and no more.

- **Implication 1:** you want to reduce the protection domain to the smallest possible set of objects
- **Implication 2:** you want to assign the minimal set of operations to each object
- **Caveat:** of course, you need to provide enough permissions to get the job done.
Least Privilege

• Limit permissions to those required and no more
• Suppose $J_1$-$J_3$ must use the permissions below
  ‣ What is the impact of the secrecy of $O_1$?

<table>
<thead>
<tr>
<th></th>
<th>$O_1$</th>
<th>$O_2$</th>
<th>$O_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_1$</td>
<td>R</td>
<td>RW</td>
<td>-</td>
</tr>
<tr>
<td>$J_2$</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>$J_3$</td>
<td>-</td>
<td>R</td>
<td>RW</td>
</tr>
</tbody>
</table>
Least Privilege

• Can least privilege prevent attacks?
  ‣ Trojan horse
  ‣ Unexpected attack surface
Least Privilege

• Can least privilege prevent attacks?
  ‣ Trojan horse
  ‣ Unexpected attack surface
  ‣ Some. No guarantee such attacks are not possible
Information Flow

• Access control that focuses on information flow restricts the flow of information between subjects and objects
  ‣ Regardless of functional requirements

• Confidentiality
  ‣ Processes cannot read unauthorized secrets
  ‣ Processes cannot leak their own secrets to authorized processes
    • How does this prevent Trojan horse attacks?

• Integrity
  ‣ Processes cannot write objects that are “higher integrity”
  ‣ In addition, processes cannot read objects that are “lower integrity” than they are
    • How does this prevent Unexpected Attack Surfaces?
Denning Security Model

- Information flow model $FM = (N, P, SC, x, y)$
  - $N$: Objects
  - $P$: Subjects
  - $SC$: Security Classes
  - $x$: Combination
  - $y$: Can-flow relation

- $N$ and $P$ are assigned security classes ("levels" or "labels").
- $SC_1 + SC_2$ determines the resultant security class when data of security classes $SC_1$ and $SC_2$ are combined.
- $SC_1 _ SC_2$ determines whether an information flow is authorized between two security classes $SC_1$ and $SC_2$.
- $SC$, $+$, and $-$ define a lattice among security classes.
Denning Security Model

• Preventing Trojan horse attacks
  ‣ Process and secret data are labeled $SC_1$ (secret)
  ‣ Public objects are labeled $SC_2$ (public)
  ‣ Only flows from $SC_2$ to $SC_1$ are authorized (public to secret)
  ‣ When data of $SC_1$ and $SC_2$ are combined, the resultant security class of the object is $SC_1$ (public and secret data make secret data)

• How does this prevent a Trojan horse from leaking data?
Information Flow

• Does information flow security impact functionality?
Information Flow

• Does information flow security impact functionality?
  ‣ Yes, so need special processes to reclassify objects
  • Called guards, but are assumed to be part of TCB
    ‣ Back to formal assurance :-P
Information Flow Models

- **Secrecy**: Multilevel Security, Bell-La Padula
- **Integrity**: Biba, LOMAC
Multilevel Security

- A multi-level security system tags all object and subject with security tags classifying them in terms of sensitivity/access level.
  - We formulate an access control policy based on these levels
  - We can also add other dimensions, called categories which horizontally partition the rights space (in a way similar to that as was done by roles)
US DoD Policy

• Used by the US military (and many others), uses MLS to define policy

• Levels:
  
  UNCLASSIFIED < CONFIDENTIAL < SECRET < TOP SECRET

• Categories (actually unbounded set)
  
  NUC(lear), INTEL(igence), CRYPTO(graphy)

• Note that these levels are used for physical documents in the governments as well.
Assigning Security Levels

• All subjects are assigned clearance levels and compartments
  ‣ Alice: (SECRET, {CRYPTO, NUC})
  ‣ Bob: (CONFIDENTIAL, {INTEL})
  ‣ Charlie: (TOP SECRET, {CRYPTO, NUC, INTEL})

• All objects are assigned an access class
  ‣ DocA: (CONFIDENTIAL, {INTEL})
  ‣ DocB: (SECRET, {CRYPTO})
  ‣ DocC: (UNCLASSIFIED, {NUC})
Multilevel Security

- Access is allowed if
  subject clearance level $\geq$ object sensitivity level \textit{and}
  subject categories $\supseteq$ object categories \textit{(read down)}

- Q: What would write-up be?

Bob: CONF., \{INTEL\})
Charlie: TS, \{CRYPTO, NUC, INTEL\})
Alice: (SEC., \{CRYPTO, NUC\})
DocA: (CONFIDENTIAL, \{INTEL\})
DocB: (SECRET, \{CRYPTO\})
DocC: (UNCLASSIFIED, \{NUC\})
Bell-La Padula Model

• A Confidentiality MLS policy that enforces:
  ‣ **Simple Security Policy**: a subject at specific classification level cannot read data with a higher classification level. This is short hand for “no read up”.
  ‣ *(star) Property*: also known as the confinement property, states that subject at a specific classification cannot write data to a lower classification level. This is shorthand for “no write down”.

```
(Top Secret, {nuclear, crypto})
(Top Secret, {nuclear})   (Secret, {nuclear, crypto})   (Top Secret, {crypto})
(Top Secret, {nuclear})   (Secret, {nuclear, crypto})   (Top Secret, {crypto})
(Secret, {nuclear})   (Top Secret, {})     (Secret, {crypto})
(Secret, {})     (Top Secret, {})     (Secret, {crypto})
```
How about integrity?

- MLS as presented before talks about who can “read” a document (confidentiality)
- Integrity considers who can “write” to a document
  - Thus, who can affect the integrity (content) of a document
  - Example: You may not care who can read DNS records, but you better care who writes to them!
- **Biba** defined a dual of secrecy for integrity
  - Lattice policy with, “no read down, no write up”
    - Users can only *create* content at or *below* their own integrity level (a monk may write a prayer book that can be read by commoners, but not one to be read by a high priest).
    - Users can only *view* content at or *above* their own integrity level (a monk may read a book written by the high priest, but may not read a pamphlet written by a lowly commoner).
Biba (example)

• Which users can modify what documents?
  ‣ Remember “no read down, no write up”

Bob: (CONF., {INTEL})
Charlie: (TS, {CRYPTO, NUC, INTEL})
Alice: (SEC., {CRYPTO, NUC})

DocA: (CONFIDENTIAL, {INTEL})
DocB: (SECRET, {CRYPTO})
DocC: (UNCLASSIFIED, {NUC})
Window Vista Integrity

- Integrity protection for **writing**
- Defines a series of protection level of increasing protection
  - installer (highest)
  - system
  - high (admin)
  - medium (user)
  - low (Internet)
  - untrusted (lowest)
- **Semantics**: If subject’s (process’s) integrity level dominates the object’s integrity level, then the write is allowed
Vista Integrity

S1 (installer)  O1 (admin)

S2 (user)  02 (untrusted)

S3 (untrusted)  03 (user)
Integrity, Sewage, and Wine

- Mix a gallon of sewage and one drop of wine gives you?
- Mix a gallon of wine and one drop of sewage gives you?

*Integrity is really a contaminant problem:* you want to make sure your data is not contaminated with data of lower integrity.
- **Low-Water Mark integrity**
  - Change integrity level based on actual dependencies

- Subject is initially at the **highest integrity**
  - But integrity level can change based on objects accessed

- Ultimately, subject has integrity of **lowest object read**
Integrity and Upgrading

• In practice, **all important programs are “high integrity”**
  ‣ And they receive input from lower integrity programs
    • Potential adversaries
  ‣ But, we **expect them to maintain their integrity**

• How do information flow integrity models model this?
  ‣ Biba - guards (no one implements)
  ‣ LOMAC - reduce integrity level (violates practice)

• So, what do we do?
  ‣ **Clark-Wilson**: Programs that receive untrusted input must either (immediately) upgrade or discard that input
    • How do we trust a complex program to ensure that property?
    • Limit entrypoints that upgrade (**CW-Lite**)
Are Goals Actually Enforced?

• Once a policy is defined that enforces expected security goals, we would like to know that it is really enforced
  ▷ But traditional access control allows processes to change the access control policy
Suppose J owns $O_1$ and $O_2$ - Is $O_1$ secret in a DAC system?

<table>
<thead>
<tr>
<th></th>
<th>$O_1$</th>
<th>$O_2$</th>
<th>$O_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>J</strong></td>
<td>R</td>
<td>R</td>
<td>RW</td>
</tr>
<tr>
<td><strong>$S_2$</strong></td>
<td>-</td>
<td>R</td>
<td>RW</td>
</tr>
<tr>
<td><strong>$S_3$</strong></td>
<td>-</td>
<td>R</td>
<td>RW</td>
</tr>
</tbody>
</table>
Safety Problem

• For a protection system
  ‣ (protection state and administrative operations)
• Prove that all future states will not result in the leakage of an access right to an unauthorized user
  ‣ Q: Why is this important?

• For most discretionary access control models,
  ‣ Safety is *undecideable*
• Means that we need another way to prove safety
  ‣ Restrict the model (no one uses)
  ‣ Test incrementally (constraints)
Access Control Administration

There are two central ways to manage a policy

1. **Discretionary** - Object “owners” define policy
   - Users have discretion over who has access to what objects and when (trusted users)
   - Canonical example, the UNIX filesystem
     - RWX assigned by file owners

2. **Mandatory** - Environment defines policy
   - OS distributor and/or administrators define a system policy that cannot be modified by normal users (or their processes)
   - Typically, information flow policies are mandatory
     - More later…
DAC vs. MAC in Access Matrix

- **Subjects:**
  - DAC: users
  - MAC: labels (via labeling state)

- **Objects:**
  - DAC: files, sockets, etc.
  - MAC: labels (via labeling state)

- **Operations:**
  - Same

- **MAC:** transitions between labels are explicit; DAC: setuid
## Mandatory Protection System

### Labeling State

<table>
<thead>
<tr>
<th></th>
<th>secret</th>
<th>unclassified</th>
<th>trusted</th>
<th>untrusted</th>
</tr>
</thead>
<tbody>
<tr>
<td>secret</td>
<td>read, write</td>
<td>read</td>
<td>read</td>
<td></td>
</tr>
<tr>
<td>unclassified</td>
<td>read, write</td>
<td>read</td>
<td>read</td>
<td></td>
</tr>
<tr>
<td>trusted</td>
<td>write</td>
<td>read, write</td>
<td>write</td>
<td></td>
</tr>
<tr>
<td>untrusted</td>
<td>read, write</td>
<td>read, write</td>
<td>read, write</td>
<td></td>
</tr>
</tbody>
</table>

### File: newfile

### File: acct

### Transition State

### Protection State

- Process: newproc
- Process: other
Mandatory Protection System

• Protection State
  ‣ Fixed set of labels for subjects and objects
  ‣ Fixed set of operations
  ‣ What happens when a new file is created?

• Labeling State
  ‣ Associates subjects and objects with labels
  ‣ All subjects and objects are labeled at all times
  ‣ What happens when you want to change permissions?

• Transition State
  ‣ Associate condition with label change
  ‣ What happens when you want to invoke a privileged program?
MAC and MLS

• A multi-level security system tags objects with security labels classifying them in terms of sensitivity/access level.
  ‣ Users login at a sensitivity/access level (transition)
  ‣ **Labeling state:** New file/process gets label of creator
  ‣ All the users’ processes run at their level (no transitions)
Take Away

- **Claim**: Traditional access control approaches (UNIX and Windows) **do not enforce security** against a determined adversary
  - (1) Trojan horses and confused deputies violate security goals
  - (2) DAC models prevent goals from being enforced
- **Mandatory Access Control** (MAC) is the way these can be achieved
  - Mandatory protection system
  - Enforce policies
    - Information flow models (MLS, Biba)
    - Least privilege MAC is often used (see next)