Access Control

Windows
Access control in Windows (since NTFS)

• Some basic functionality similar to Unix
  – Specify access for groups and users
    • Read, modify, change owner, delete

• Some additional concepts
  – Tokens
  – Security attributes

• Generally
  – More flexibility than Unix
    • Can define new permissions
    • Can give some but not all administrator privileges
Identify subject using SID

- Security ID (SID)
  - Identity (replaces UID)
    - SID revision number
    - 48-bit authority value
    - variable number of Relative Identifiers (RIDs), for uniqueness
  - Users, groups, computers, domains, domain members all have SIDs
Each process has set of tokens

• Security context
  – Privileges, accounts, and groups associated with the process or thread
  – Presented as set of tokens

• Reference Monitor
  – Uses tokens to identify the security context of a process or thread

• Impersonation token
  – Used temporarily to adopt a different security context, usually of another user
Each object has security descriptor

- Security descriptor, associated with an object, specifies who can perform what actions on the object
- Fields:
  - Header
    - Descriptor revision number
    - Control flags, attributes of the descriptor
      - E.g., memory layout of the descriptor
  - SID of the object's owner
  - SID of the primary group of the object
  - Two attached optional lists:
    - Discretionary Access Control List (DACL) controls access by users, groups, ...
    - System Access Control List (SACL) controls logging of access to system logs
**Example access request**

<table>
<thead>
<tr>
<th>Access token</th>
</tr>
</thead>
<tbody>
<tr>
<td>User: Mark</td>
</tr>
<tr>
<td>Group1: Administrators</td>
</tr>
<tr>
<td>Group2: Redhaired</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Security descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revision Number</td>
</tr>
<tr>
<td>Control flags</td>
</tr>
<tr>
<td>Owner SID</td>
</tr>
<tr>
<td>Group SID</td>
</tr>
<tr>
<td>DACL Pointer</td>
</tr>
<tr>
<td>SACL Pointer</td>
</tr>
<tr>
<td>Deny</td>
</tr>
<tr>
<td>Redhaired</td>
</tr>
<tr>
<td>Read, Write</td>
</tr>
<tr>
<td>Allow</td>
</tr>
<tr>
<td>Mark</td>
</tr>
<tr>
<td>Read, Write</td>
</tr>
</tbody>
</table>

**Access request: write**
- Action: denied

- User Mark requests write permission
- Descriptor denies permission to group
- Reference Monitor denies request

Order of ACEs in ACL matters. Windows reads ACL until permission is Granted or Denied, then stops.
Impersonation Tokens  (compare to setuid)

- Process adopts security attributes of another
  - Client passes impersonation token to service
- Client specifies impersonation level of service
  - Anonymous
    - The service can impersonate the client but the impersonation token does not contain any information about the client.
  - Identification
    - The service can get the identity of the client and use this information in its own security mechanism, but it cannot impersonate the client.
  - Impersonation
    - The service can impersonate the client (on local computers).
  - Delegation
    - The service can impersonate the client on local and remote computers.
Access control policies
Multi-Level Security (MLS) Concepts

- **Military security policy**
  - Classification involves sensitivity levels, compartments
  - Do not let classified information leak to unclassified files

- **Group individuals and resources**
  - Use some form of hierarchy to organize policy

- **Other policy concepts**
  - Separation of duty
  - “Chinese Wall” Policy (managing conflicts of interests)
Military security policy

- Sensitivity levels:
  - Top Secret
  - Secret
  - Confidential
  - Restricted
  - Unclassified

- Compartments:
  - Satellite data
  - Micronesia
  - Middle East
  - Israel
Example: military security policy

- Classification of personnel and data
  - Class = \( \langle \text{rank}, \text{compartment} \rangle \)

- Dominance relation
  - \( C_1 \leq C_2 \) iff \( \text{rank}_1 \leq \text{rank}_2 \)
    and \( \text{compartment}_1 \subseteq \text{compartment}_2 \)

  - Example: \( \langle \text{Restricted, Israel} \rangle \leq \langle \text{Secret, Middle East} \rangle \)

- Applies to
  - Subjects – users or processes
  - Objects – documents or resources
Example: commercial policy

<table>
<thead>
<tr>
<th>Internal</th>
<th>Product specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proprietary</td>
<td>Discontinued</td>
</tr>
<tr>
<td>Public</td>
<td>In production</td>
</tr>
<tr>
<td></td>
<td>OEM</td>
</tr>
</tbody>
</table>
Generalizing: lattices

- A dominance relationship that is transitive and antisymmetric defines a **lattice**
- For two levels \(a\) and \(b\), maybe neither \(a \geq b\) nor \(b \geq a\)
- However, for every \(a\) and \(b\), there is a **lowest upper bound** \(u\) for which \(u \geq a\) and \(u \geq b\), and a **greatest lower bound** \(l\) for which \(a \geq l\) and \(b \geq l\)
- There are also two elements \(U\) and \(L\) that dominate/are dominated by all levels
  - In next slide’s example,
    \[U = (\text{“Top Secret”}, \{\text{“Middle East”}, \text{“Micronesia”}\})\]
    \[L = (\text{“Unclassified”}, \emptyset)\]
Example Lattice

(“Top Secret”, {“Middle East”, “Micronesia”}),

(“Top Secret”, {“Middle East”})

(“Secret”, {“Middle East”, “Micronesia”})

(“Secret”, {“Middle East”})

(“Secret”, {“Micronesia”})

(“Unclassified”, ∅)
Bell-LaPadula Confidentiality Model

When is it OK to release information?

Two properties

- No read up
  - A subject $S$ may read object $O$ only if $C(O) \leq C(S)$

- No write down
  - A subject $S$ with read access to $O$ may write to object $P$ only if $C(O) \leq C(P)$

You may only read below your classification and write above your classification

Mandatory Access Control: protect even against malicious software / users / trojan horses who try to violate policy. (by contrast: Discretionary Access Control)
Biba Integrity Model

- Rules that preserve integrity of information (dual to confidentiality)
- Two properties
  - No write up
    - A subject $S$ may write object $O$ only if $C(S) \geq C(O)$
      (Only trust $S$ to modify $O$ if $S$ has higher rank …)
  - No read down
    - A subject $S$ with read access to $O$ may write object $P$
      only if $C(O) \geq C(P)$
      (Only move info from $O$ to $P$ if $O$ is more trusted than $P$)
- You may only write below your classification and read above your classification
Picture: Integrity

Read above, write below

S
Proprietary
Public

Read below, write above

S
Proprietary
Public
Problems

- Covert channels
- Declassification (when OK to violate strict policy?)
- Composability (e.g., one-time-pad)
  - Aggregation (many “low” facts may enable deduction of “high” information)
- Overclassification (label creep)
- Incompatibilities (e.g., upgraded files “disappear”)
- Polyinstantiation (maintaining consistency when different users have different view of the data, and perhaps even see “cover stories”)
Other policy concepts

- **Separation of duty**
  - If amount is over $10,000, check is only valid if signed by two authorized people
  - Two people must be *different*
  - Policy involves role membership and $\neq$

- **Chinese Wall Policy**
  - Lawyers L1, L2 in same firm
  - If company C1 sues C2,
    - L1 and L2 can each work for either C1 or C2
    - No lawyer can work for opposite sides in any case
  - Permission depends on use of other permissions

These policies cannot be represented using access matrix
Access control in web browsers

**Operating system**
- **Objects:**
  - System calls
  - Processes
  - Disk
  - File
- **Principals:**
  - Users
- **Discretionary access control**
- **Vulnerabilities**
  - Buffer overflow
  - Root exploit

**Web browser**
- **Objects:**
  - Document object model
  - Frames
  - Cookies / localStorage
- **Principals:** “Origins”
- **Mandatory access control**
- **Vulnerabilities**
  - Cross-site scripting
  - Universal scripting
Components of browser security policy

Frame-Frame relationships
- canScript(A,B)
  - Can Frame A execute a script that manipulates arbitrary/nontrivial DOM elements of Frame B?
- canNavigate(A,B)
  - Can Frame A change the origin of content for Frame B?

Frame-principal relationships
- readCookie(A,S), writeCookie(A,S)
  - Can Frame A read/write cookies from site S?
Policies: further reading

- Anderson, Stajano, Lee, *Security Policies*
  [https://www.cl.cam.ac.uk/~rja14/Papers/security-policies.pdf](https://www.cl.cam.ac.uk/~rja14/Papers/security-policies.pdf)

- Levy, *Capability-Based Computer Systems*
Information Flow Control
Information Flow Control

- An information flow policy describes authorized paths along which information can flow
- For example, Bell-La Padula describes a lattice-based information flow policy
Program-level IFC

- Input and output variables of program each have a (lattice-based) security classification $S()$ associated with them.
- For each program statement, compiler verifies whether information flow is secure.
- For example, $x = y + z$ is secure only if $S(x) \geq \text{lub}(S(y), S(z))$, where lub() is lowest upper bound.
- Program is secure if each of its statements is secure.
Implicit information flow

Conditional branches carry information about the condition:

```c
secret bool a;
public bool b;
...
// b = a;
if (a) {
    b = 1;
} else {
    b = 0;
}
```

Possible solution: assign label to program counter and include it in every operation. (Too conservative!)
IFC security notion: non-interference

Many variants and formalizations. At very high level: [Goguen Meseguer 1982]

Experiment #1:

\[ S(p) = \{a\} \]

\[ S(q) = \{\} \]

“HIGH”

Experiment #2:

\[ S(p') = \{a\} \]

\[ S(q) = \{\} \]

“LOW”
IFC: Implementation approaches

- **Language-based IFC (JIF & others)**
  - Compile-time tracking for Java, Haskell
  - Static analysis
  - Provable security
  - Label creep (false positives)

- **Dynamic analysis**
  - Language / bytecode / machine code
  - Tainting
  - False positives, false negatives
  - Performance
  - Hybrid solutions

- **OS-based DIFC (Asbestos, HiStar, Flume)**
  - Run-time tracking enforced by a trusted kernel
    - Works with any language, compiled or scripting, closed or open
Difficulties with Information Flow Control

- Label creep
  - Examples:
    - Branches
    - Logically-false flows
    - “Doesn’t really matter”
- Declassification
  - Example: encryption
- Catching all flows
  - Exceptions (math division by 0, null pointer, out of bounds access…)
  - Signals
  - Covert channels