Design Motivation

- Security was designed to meet requirements for C2 rating in Orange Book
  - Secure logon – users must be uniquely identified
  - Discretionary access control – Owner determines access
  - Auditing – Record security related events in a logfile
  - Object reuse protection – Initialize all objects before giving access to users
  - Trusted path – Functionality to detect trojan horses at authentication time (called SAS in Windows)
  - Trusted facility management – Separate accounts for users and administrators
- Windows NT 3.5 SP3 was the first Windows version to earn C2 rating (1995)
  - Windows NT 4 SP6a earned C2 rating in 1999

Introduction

- Windows XP evolved from Windows 2000
- Windows 10, 8, 7 and Vista evolved from XP
- Similar security solution
- Rich support for managing security
- Standalone computers administered locally
- Domains used for centralized administration
- Domain controller (DC) has information about users
- Acts as a trusted third party

Windows Logon (somewhat simplified)

- Winlogon.exe handles the logon and responds to the Secure Attention Sequence (SAS)
  - CTRL+ALT+DEL
- Winlogon uses libraries that authenticates the user
  - Can be libraries for passwords, smartcards, biometric data etc
- Local Security Authority (LSA) creates an access token
  - LSA is responsible for the local security policy (who can log in, password policies, privileges, what should be audited etc)
- Password hashes are stored in SAM
  - Security Accounts Manager
**SAM File**
- Stores user account information
  - Username
  - Full name
  - Expiration date
  - Password dates (date of last change, expiry, when it can be changed next time, if it can be changed)
  - Logon hours and workstations (thrown out a certain time or continue)
  - Profile path and logon script name
  - Home directory
  - Groups
- Locked while machine is running

**Local Accounts VS Domain accounts**
- Local accounts
  - NTLM used as authentication protocol
- Domain accounts
  - Kerberos V5 used as authentication protocol
    - Mutual authentication
    - This will be covered in detail later in the course
  - NTLM used in some cases
    - Unilateral authentication

**NTLM Hash and Protocol**
- Challenge response
- Server sends 8 byte random challenge
- Response calculated as:
  - MD4(password) gives 16 byte result (NTLM hash stored in SAM database)
  - Pad with 5 zero bytes → 21 bytes
  - Split into 3 DES keys and encrypt challenge with each key
  - 24 byte response

**LM Hash**
- Can you find problems here?
  - If wanted, both NTLM and LM response are used
    - This was default before Windows Vista
  - LM hash calculated as
    - Convert password to uppercase and pad to 14 bytes
    - Split into two parts of 7 byte each → two DES keys
    - Encrypt "KGS!@#$%" with the two keys to get 16 bytes LM hash which is stored in the SAM database
  - LM response calculated same way as NTLM response
Password Hashes, Problems

- NTLM hash is stored in the SAM file (local accounts)
  - Problem 1: MD4 is a very fast hash function
  - Problem 2: No salt is used so time-memory tradeoff attacks (rainbow tables) can be used
- Possibly, also the LM hash is stored in the SAM file
  - Problem 3: DES is a fast block cipher
  - Problem 4: No salt here either…
  - Problem 5: Passwords up to 14 characters are never better than passwords of 7 characters
  - Problem 6: There are no lowercase characters in the effective character set

Access Control

- Security Reference Monitor (SRM) is responsible for determining access control
- Three parameters are considered
  - Identity of subject (SID)
  - Type of access
  - Object security settings (Security Descriptor)

Access Token

- After successful authentication LSA builds an access token
- Processes which run as the user has a copy of the token
- When a process interacts with a securable object, token determines authorization level

<table>
<thead>
<tr>
<th>User SID</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIDs of groups the user is member of</td>
</tr>
<tr>
<td>List of privileges</td>
</tr>
<tr>
<td>Default DACL, Owner, Group</td>
</tr>
<tr>
<td>Miscellaneous</td>
</tr>
</tbody>
</table>

Two Kinds of Access Tokens

- Token is either a primary access token or an impersonation access token
- Primary access token – access token of the user account associated with the process.
  - Every process has this
- Impersonation access token – allows a thread to execute in a different security context than the process owner.
  - A thread may additionally have an impersonation access token

- Example: Database server runs with high privileges
  - Threads handle concurrent user requests
  - Thread get token of user → restricted access
**SID**

- Security Identifier
- Unique for each user or group
- Format: \[S-R-I-SA-SA-SA-N\]
  - S: The letter S (just means that the string is a SID)
  - R: revision number (1)
  - I: Identifier authority (5 for user accounts)
  - SA: subauthority (specifies domain or computer)
    - Can be up to 14 groups, but 3 is typical
  - N: relative identifier, incremented for each new principal

**Known SIDs**

- Generic groups and users
  - S-1-1-0 Everyone, a group that includes all users
  - S-1-5-20 Network Service
  - S-1-5-18 SYSTEM, local operating system
  - S-1-5-SA-SA-SA-500 Administrator
  - S-1-5-SA-SA-SA-501 Guest account (no password required)
  - S-1-5-SA-SA-SA-512 Domain Admins (global group)

**Privileges**

- The right to perform system related operations
  - Shutting down
  - Change system time
  - Backup files
  - Generate audit
- Applies only to local computer. A user can have different privileges on different machines in a domain.
- Privileges can be assigned to both users and groups
- Access token is checked when user tries to perform privileged operation
- Differs from access rights
  - Access to resources and tasks, not objects
  - Stored with subject
  - Admin assigns privileges
- Stored in access token produced at logon

**Objects**

- All resources are objects
  - Files, folders, printers, registry keys, processes, threads, access tokens, etc..
- Containers can hold other objects, e.g., folders
- Noncontainers cannot hold other objects, e.g., files
- **Securable object** – Any object that can be shared
- All securable objects can have a security descriptor
  - But it is not necessary
Security Descriptor

- Contains security information associated with an object
- SID for the owner
- SID for the primary group
- DACL (Discretionary Access Control List) specifying access rights
- SACL (System Access Control List) specifying types of events that should generate audit records

DACL

- Identifies who is allowed or denied access to an object
- If an object has no DACL, everyone has full control
- An empty DACL results in everyone is denied access
- A SID can be allowed or denied access.
- All “deny” entries are stored in the beginning of the DACL
- Contains a list of access control entries (ACEs)

Example: Accessing Object

- Two processes (subjects) wants read access to an object

Searching the DACL

- Go through list of ACEs until all access requests are allowed or any access request is denied
- Otherwise deny access
- Consequences
  - Deny has higher precedence than allow
  - If user SID has read only access and user is member of group which SID has read + write, then user has read + write access (Different from Unix/Linux)
- Example, Alice is member of group “Students”

ACE 1
Principal SID: Alice
Deny: read
Result
- Read access allowed
- Write access denied

ACE 2
Principal SID: Students
Allow: read, write

ACE 1
Principal SID: Alice
Deny: write

ACE 2
Principal SID: Alice
Allow: read, write

ACE 3
Principal SID: Group 3
Allow: read

ACE 4
Principal SID: Group 3
Allow: write

ACE 5
Principal SID: Group 3
Allow: read + write

Example, Alice is member of group “Students”
Access Rights (in the ACE)

- Since there are so many different types of objects, access rights look different for different types.
- Standard access rights apply to (almost) all objects:
  - `DELETE` – delete the object
  - `READ_CONTROL` – read info in security descriptor (owner, group and DACL)
  - `WRITE_DAC` – write access to the DACL
  - `WRITE_OWNER` – write access to the field “owner” in the security descriptor
  - `SYNCHRONIZE` – The right to synchronize with the object

Generic Access Rights

- Since there are many different types of objects, there are very many different types of access rights.
- Generic access rights gives a mapping to specific access rights for a type of objects

<table>
<thead>
<tr>
<th>Access Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENERIC_EXECUTE</td>
</tr>
<tr>
<td>GENERIC_READ</td>
</tr>
<tr>
<td>GENERIC_WRITE</td>
</tr>
</tbody>
</table>

Example – Files and directories

- `FILE_EXECUTE`
- `FILE_READ_ATTRIBUTES`
- `STANDARD_RIGHTS_EXECUTE`
- `SYNCHRONIZE`
- `FILE_READ_DATA`
- `FILE_READ_ATTRIBUTES`
- `STANDARD_RIGHTS_READ`
- `SYNCHRONIZE`
- `FILE_WRITE_ATTRIBUTES`
- `FILE_WRITE_DATA`
- `FILE_WRITE_ATTRIBUTES`
- `STANDARD_RIGHTS_WRITE`
- `SYNCHRONIZE`

Access Control, Network Shares

- Users must go through two ACL’s to access a file via a share:
  - ACL on the share
  - ACL on the file itself
  - User’s effective permission through a file share is determined by masking both sets of ACL’s together.

Example 1:
- Client sets share permission to read only for everyone and file permission to read+write for everyone.
  - Result: Users on client machine get read+write, network users get read.

Example 2:
- Client sets share permission to full control for everyone and file permission to read for everyone.
  - Result: Users on client machine get read access, network users get read access.

Access Mask

- The access rights are given by a 32-bit integer

<table>
<thead>
<tr>
<th>Bits</th>
<th>Access Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-15</td>
<td>Specific rights for the current object type</td>
</tr>
<tr>
<td>16-22</td>
<td>Standard rights</td>
</tr>
<tr>
<td>23</td>
<td>Access system security (e.g., SACL)</td>
</tr>
<tr>
<td>24-27</td>
<td>reserved</td>
</tr>
<tr>
<td>28</td>
<td>generic all</td>
</tr>
<tr>
<td>29</td>
<td>generic execute</td>
</tr>
<tr>
<td>30</td>
<td>generic write</td>
</tr>
<tr>
<td>31</td>
<td>generic read</td>
</tr>
</tbody>
</table>
Access Control Matrix Implementation

Recall the two variants
- In Windows a combination is used

### Two alternatives

- **Capabilities**
  - Store access rights with *subjects*

- **Access Control List**
  - Store access rights with *objects*

- **Privileges**
  - Access rights to objects

The Registry

- Central database for Windows configuration data
- Just files on the harddisk
- Entries are called *keys* and *values*
- A registry *Hive* is a group of keys, subkeys, and values in the registry stored in a file
  - “Registreringsdatafil” in swedish
- Protecting the integrity of registry data is important
  - Example: The search path is set in registry, if an attacker can modify it, malicious software can be inserted/executed.
- Proprietary format: registry editor (Regedit.exe)

The Registry

5 root keys (none is a hive)

- **HKEY_CLASSES_ROOT**
  - Merge of HKEY_LOCAL_MACHINE\SOFTWARE\CLASSES and HKEY_USERS\SIDs\Classes
  - Contains file extension associations.

- **HKEY_CURRENT_USER**
  - Symbolic link to key under HKEY_USERS that represents the user that is logged in

- **HKEY_LOCAL_MACHINE**
  - Contains several hives that store information about the local computer

- **HKEY_USERS**
  - Contains all active user profiles on the system.

- **HKEY_CURRENT_CONFIG**
  - Symbolic link to HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Hardware Profiles\Current.
  - Information about the hardware profiles. Used when system starts up.

Some hives

<table>
<thead>
<tr>
<th>Path to registry hive</th>
<th>Path to file hive</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKEY_LOCAL_MACHINE\SYSTEM</td>
<td>WINDOWS\system32\config\system</td>
</tr>
<tr>
<td>HKEY_LOCAL_MACHINE\SAM</td>
<td>WINDOWS\system32\config\sam</td>
</tr>
<tr>
<td>HKEY_LOCAL_MACHINE\SECURITY</td>
<td>WINDOWS\system32\config\security</td>
</tr>
<tr>
<td>HKEY_LOCAL_MACHINE\SOFTWARE</td>
<td>WINDOWS\system32\config\software</td>
</tr>
</tbody>
</table>

Temporary Hives

- **HKEY_LOCAL_MACHINE\hardware**
  - Hardware is detected when system starts
- **HKEY_LOCAL_MACHINE\system\clone**
  - Built during startup, saved as HKEY_LOCAL_MACHINE\SYSTEM\Select\LastKnownGood Control Set if startup is successful
  - If there is a problem to start (e.g., if an installed driver has damaged the system), then LastKnownGood configuration can be used by copying this to CurrentControlSet.
**Restricted Context**

- Application can start process with **restricted token**
- Process can start process or thread with restricted token
  - Can be either primary token or impersonation token
- **Example 1**: Untrusted webpages can be displayed with restrictions
- **Example 2**: Email attachments can be opened with restrictions
- Restrict by (one or more of):
  1. Remove privileges
  2. Set deny-only attribute to SIDs
  3. Specify restricting SID

**How To Restrict a Token**

1. Let group SIDs be used for deny only
2. Add restricted SID
   → Two access checks are done

<table>
<thead>
<tr>
<th>Restricted Token</th>
<th>User SID</th>
<th>Group SIDs</th>
<th>Privileges</th>
</tr>
</thead>
<tbody>
<tr>
<td>User SID</td>
<td>Alice</td>
<td>Admin (deny only)</td>
<td></td>
</tr>
<tr>
<td>Group SIDs</td>
<td>Users</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restricted SIDs</td>
<td>SID_Restr</td>
<td></td>
<td>none</td>
</tr>
<tr>
<td>Privilages</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example: Process with restricted token require read access

<table>
<thead>
<tr>
<th>Principal SID</th>
<th>Allow</th>
<th>Access</th>
<th>Principal SID</th>
<th>Allow</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>read, write</td>
<td>b) Access granted</td>
<td>Admin</td>
<td>read</td>
<td>c) Access denied</td>
</tr>
<tr>
<td>SID_Restr</td>
<td>read</td>
<td>a) Access granted</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**User Account Control (UAC)**

- Introduced in Windows Vista
- Administrators get two access tokens when logging in
  - One administrator token
  - One standard user token
- Standard user token used unless administrator privileges are needed
  - User has to actively acknowledge use of administrator token
- Windows 7+ uses UAC, but not all programs ask for explicit permission

**Mandatory Access Control**

- Windows Vista and later include mandatory access control
  - Called Integrity Control
- Access tokens have an integrity level
  - Untrusted (Processes started by group Anonymous)
  - Low integrity (e.g., IE in protected mode)
  - Medium integrity (Used by normal applications when UAC is enabled)
  - High integrity (Admin applications started through UAC, normal applications if UAC is disabled)
  - System integrity (Used by some system processes)
**Mandatory Access Control**

- Each object can also have an integrity level stored in the SACL.
- Default for newly created objects:
  - If access token is lower than medium, integrity level of object is same as in access token.
  - If access token is medium or higher, integrity level of object is medium.
- **Partial ordering:** Subject has label S, object has label O.
  - Write access granted if O ≤ S.
- Subjects' integrity level must dominate object's integrity level in write operations.
  - Checked before DACL.

**Example**

- Internet Explorer 7 can run in Protected Mode.
  - Will run with "low integrity" access token.
- Can not be forced to make changes to operating system files, registry, etc.
  - However, it can read all this data.
- Can write to history, cookies etc.
  - This can be compared to the Biba security model.

**Secure Boot in Windows 8**

- **UEFI** (Unified Extensible Firmware Interface) provides support for Secure Boot.
  - OEMs providing Windows 8 must support it.
- Only trusted boot loader can be loaded.
- **db** is a database with known good CAs, hashed:
  - Includes Microsoft Windows CA.
- **dbx** is a database with known bad CAs and hashes.
- Databases are signed with a Microsoft key.

Idea: It will not be possible to install other Boot loaders than those trusted. Protects against certain rootkits.

**Windows security on the Man-Machine Scale**

- Complex solution with many options.
- Users can easily get the exact functionality they want.
- Relatively difficult to get high assurance.