Layered implementation of security

- Corporate Security
  - Physical Security
    - Network Security
      - System Security
        - Application Security
        - Object access
        - Intrusion detection
        - UPS, backup comms lines
  - User profile
  - Firewall, VPN gateway
  - LAN interface
  - Locks, access control
  - User education

Security policies

Should meet security goals of

- Authentication
- Authorization
- Integrity
- Confidentiality
- Audit/Logging
Security Violations

• **Categories**
  – Breach of confidentiality
  – Breach of integrity
  – Breach of availability
  – Theft of service
  – Denial of service

• **Methods**
  – Masquerading (breach authentication)
  – Replay attack
    • Message modification
  – Man-in-the-middle attack
  – Session hijacking
Security concepts and relationships

Owner\ns

impose\n
value

threats\n
give rise to\n
that increase to\n
vulnerabilities\n
that may be reduced by\n
leading to\n
risk\n
that may possess\n
to reduce\n
countermeasures\n
may be aware of\n
That increase\n
to assets\n
Threat agents\n
wish to minimise\n
that exploit\n
impose\n
that may wish to abuse and/or may damage

Terms Frequently Used

- **Authentication**
  - when a person logs on to a system, authentication means checking and verifying the identity of the person logging on. The term is also used where the identity of IT components or application is checked.

- **Authorization**
  - authorization entails checking whether a person, IT component or application has permission to carry out a particular action.

- **Data protection**
  - data protection refers to the protection of person-related data against misuse by third parties (not to be confused with data security).

- **Data security**
  - data security refers to the protection of data in relation to the pertinent confidentiality, availability and integrity requirements. Another term for this is "IT security".

- **Data backup**
  - during a backup, copies of existing data sets are created to protect against loss of data.

- **Penetration testing**
  - a penetration test is a deliberate, normally simulated, attempted attack on an IT system. It is used to check the effectiveness of existing security measures.

Components of Security

- Authentication
- Access control
- Confidentiality
- Data integrity
- Non-repudiation
- Management
- Availability
- Audit
IBM i Resource Security
User Owned Objects

• **Each** object has one owner
• When an object is created, an owner is assigned. The ownership may be transferred later
• The owner initially has all object and data permission
• The authority may be removed, but the owner may grant any authority back to himself at anytime
• It is not possible to delete a user who owns objects. Two solutions are offered:
  – Transfer ownership
  – Delete owned objects

• **QDFTOWN** is an IBM supplied user profile used when:
  – An object has no owner
  – The object ownership might pose security exposure
Object Permissions

Private and Public permissions consist of one or more of the following:

- **Specific Object Authorities**
  - **Exclude**
  - **Object Management**
    - Operational
    - Management
    - Existence
    - Alter
    - Reference
  - **Data Authority**
    - Read
    - Add
    - Update
    - Delete
    - Execute
# Object Management Permissions

<table>
<thead>
<tr>
<th>Permission</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operational (*OBJOPR)</td>
<td>Look at the description of an object and use the object as determined by the data authorities the user has. To open a file, the user must have *OBJOPR.</td>
</tr>
<tr>
<td>Management (*OBJMGT)</td>
<td>Authorize users to the object, move or rename the object and add members to database files. All functions defined for *OBJALTER and *OBJREF.</td>
</tr>
<tr>
<td>Existence (*OBJEXIST)</td>
<td>Change ownership and delete the object, free storage for the object, perform save and restore operations for the object.</td>
</tr>
<tr>
<td>Alter (*OBJALTER)</td>
<td>Add, clear, initialize and reorganize members of database files, alter and add attributes of database files, add and remove triggers, change attributes of SQL packages.</td>
</tr>
<tr>
<td>Reference (*OBJREF)</td>
<td>Specify database file as the parent in a referential constraint.</td>
</tr>
<tr>
<td>Authorization List (*AUTLMGT)</td>
<td>Add and remove users and their authorities from an authorization list.</td>
</tr>
</tbody>
</table>
## Object Data Permissions

<table>
<thead>
<tr>
<th>Permission</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read (*READ)</td>
<td>Display the contents of an object, such as viewing the records in a file.</td>
</tr>
<tr>
<td>Add (*ADD)</td>
<td>Add entries to an object, such as adding messages to a message queue, or records to a file.</td>
</tr>
<tr>
<td>Update (*UPD)</td>
<td>Change entries in an object, such as changing records in a file.</td>
</tr>
<tr>
<td>Delete (*DLT)</td>
<td>Remove entries from an object, such as removing messages from a message queue or deleting records from a file.</td>
</tr>
<tr>
<td>Execute (*EXECUTE)</td>
<td>Run a program, or search a library or directory.</td>
</tr>
<tr>
<td>Exclude (*EXCLUDE)</td>
<td>Object access prevented</td>
</tr>
</tbody>
</table>
# Commonly Used Permissions

<table>
<thead>
<tr>
<th>Object Control</th>
<th>Data Authority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>Read</td>
</tr>
<tr>
<td>Management</td>
<td></td>
</tr>
<tr>
<td>Existence</td>
<td></td>
</tr>
<tr>
<td>Alter</td>
<td></td>
</tr>
<tr>
<td>Reference</td>
<td></td>
</tr>
<tr>
<td>*All</td>
<td>X</td>
</tr>
<tr>
<td>*Change</td>
<td></td>
</tr>
<tr>
<td>*Use</td>
<td></td>
</tr>
<tr>
<td>*Exclude</td>
<td></td>
</tr>
</tbody>
</table>
Group Profiles

- Allows users with similar jobs to share permissions without having to share the same password
- Should choose a naming convention which makes groups easily recognizable
- Create with No password (sign-on not allowed)
- Users can be a member of more than one group profiles
- Should assign groups in order of use
- Permissions are additive at the group level
Authorization List

Name: AUTL1
Owner: USER3
User Authority:
*PUBLIC
USER1
USER2
USER4
USER5

Exclude
Use
Change
All
Change

LIBA
LFILEB
PFILEC
PROGD

*Public is on all authorization lists
Authorization List vs. Group Profile

Authorization List

BILL
WAYNE
GUNNAR
FRANK

Use
All
Use
Change

FINDEPT/FILE3
LIB23
DSP05

Objects

Group Profile

BILL
WAYNE
GUNNAR
FRANK

GROUP

Use
All
Change

FINDEPT/FILE3
LIB23
DSP05

Objects
Groups and AUTLs Compared

<table>
<thead>
<tr>
<th><strong>AUTHORIZATION LISTS</strong></th>
<th><strong>GROUPS</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Can secure multiple objects</td>
<td>Can secure multiple objects</td>
</tr>
<tr>
<td>A user can be on multiple lists</td>
<td>A user can be a member of 16 groups</td>
</tr>
<tr>
<td>Users can have different authority</td>
<td>All users in a group have the same authority</td>
</tr>
<tr>
<td>Same authority for different objects</td>
<td>Different authority for different objects</td>
</tr>
<tr>
<td>using same list</td>
<td></td>
</tr>
<tr>
<td>An object can be secured by only one</td>
<td>An object can be authorized to many groups</td>
</tr>
<tr>
<td>one authorization list</td>
<td></td>
</tr>
</tbody>
</table>
Specifying Specific Authority for Objects In the Integrated File System

*RWX  object operational authority, and all the data authorities
*RX   object operational authority, read and execute
*RW   object operational authority, read add, update and delete
*WX   object operational authority, add, update, delete and execute
*R    object operational authority and read
*W    object operational, add, update, and delete
*X    object operational and execute
*EXCLUDE  prevents access to object
Access Matrix
**Access Matrix**

<table>
<thead>
<tr>
<th>Domain</th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_3$</th>
<th>printer</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1$</td>
<td>read</td>
<td></td>
<td>read</td>
<td></td>
</tr>
<tr>
<td>$D_2$</td>
<td></td>
<td></td>
<td></td>
<td>print</td>
</tr>
<tr>
<td>$D_3$</td>
<td></td>
<td>read</td>
<td>execute</td>
<td></td>
</tr>
<tr>
<td>$D_4$</td>
<td>read</td>
<td>write</td>
<td>read</td>
<td>write</td>
</tr>
</tbody>
</table>

- View protection as a matrix (access matrix)
- Rows represent domains
- Columns represent objects
- $\text{Access}(i, j)$ is the set of operations that a process executing in Domain $i$ can invoke on Object $j$

Source: [SIL]
Use of Access Matrix

• If a process in Domain D_i tries to do “op” on object O_j, then “op” must be in the access matrix.

• Can be expanded to dynamic protection.
  – Operations to add, delete access rights.
  – Special access rights:
    • owner of O_i
    • copy op from O_i to O_j
    • control – D_i can modify D_j access rights
    • transfer – switch from domain D_i to D_j

Source: [SIL]
Use of Access Matrix (Cont.)

- Access matrix design separates mechanism from policy.
  - Mechanism
    - Operating system provides access-matrix + rules.
    - If ensures that the matrix is only manipulated by authorized agents and that rules are strictly enforced.
  - Policy
    - User dictates policy.
    - Who can access what object and in what mode.

Source: [SIL]
Access Matrix of Figure A With Domains as Objects

<table>
<thead>
<tr>
<th>Domain</th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_3$</th>
<th>Laser Printer</th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1$</td>
<td>read</td>
<td></td>
<td>read</td>
<td>switch</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_2$</td>
<td></td>
<td>print</td>
<td></td>
<td>switch</td>
<td>switch</td>
<td>switch</td>
<td>switch</td>
<td>control</td>
</tr>
<tr>
<td>$D_3$</td>
<td>read</td>
<td></td>
<td>execute</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_4$</td>
<td>write</td>
<td>write</td>
<td></td>
<td>switch</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: [SIL]
Domain Switch (UNIX)

Domain switch accomplished via file system
– Each file has associated with it a domain bit (setuid bit).
– When file is executed and setuid = on, then user-id is set to owner of the file being executed. When execution completes user-id is reset.
Default: Each Program will be created with USRPRF(*USER).
When a program created with USRPRF(*OWNER) is run, objects are accessed with the authority of the user running the program plus the program owner's authority.
Used to temporarily give authority to objects the user normally would not have.
Limited Capabilities

Users' capability to change their initial program, menu, current library and attention program and prohibit them from running most IBM i OS commands.

<table>
<thead>
<tr>
<th>Limit initial program/menu capabilities</th>
<th>Initial Program</th>
<th>Initial Menu</th>
<th>Current Library</th>
<th>Attention Program</th>
<th>Execute Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not limit</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Limit some capabilities</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Limit capabilities</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Users can still run commands created or changed with parameter ALWLMTCPB(*YES)
Implementation of Access Matrix

Each column = **Access-control** list for one object
- Defines who can perform what operation.
  - Domain 1 = Read, Write
  - Domain 2 = Read
  - Domain 3 = Read
- Example: Unix

Each Row = **Capability List** or C-list (like a key)
- For each domain, what operations allowed on what objects.
  - Object 1 – Read
  - Object 4 – Read, Write, Execute
  - Object 5 – Read, Write, Delete, Copy
- Example: IBM i
Tag-bit (IBM i)

• Pointers should not be modified by User programs
  – User programs can use (object name, library, object type) and should not modify, copy or delete resolved pointers containing a (virtual) address of an object

• Usage of a memory protection bit (tag-bit)
  – Tag-bits are used to detect modifications
  – only privileged instructions (running in system state mode)
    • can modify the contents of a pointer and
    • set the tag bit in the pointer
Revocation of Access Rights

• Access List – Delete access rights from access list.
  – Simple
  – Immediate

• Capability List – Scheme required to locate capability in the system before capability can be revoked.
  – Reacquisition
    • Periodically, capabilities are deleted from each domain
  – Back-pointers
    • A list of pointer is maintained with each domain
  – Indirection
    • Each capability points to a global table
  – ...

Comparison of ACL‘s and Capabilities *

(a) Given a subject, what objects can it access, and how?
(b) Given an object, what subjects can access it, and how?

- Either ACL‘s or capabilities can answer both question without scanning all objects or subjects
- With Capabilities it‘s easier to answer (a)
- With ACL‘s it‘s easier to answer (b)
- In practical (b) is asked more often than (a) -> ACL‘s
- Intrusion detection means answering question (b), will become more common -> Capabilities
- Access right revokation is simpler with ACL‘s
- With ACL‘s every access to an object must be checked, requiring a search in the ACL; capabilities have only to verified if they are valid
- Most systems use a combination of both

*see also: Matt Bishop, Introduction to Computer Security, Addison-Wesley, 2004
Defining Roles in IBM i
Special Authorities

- **All object access** (*ALLOBJ*)
  - Access to all system resources

- **Auditing control** (*AUDIT*)
  - Control audit system values

- **Job control** (*JOBCTL*)
  - Manage output queues, job queues and printers; change job attributes; stop subsystems; IPL

- **Save/restore** (*SAVSYS*)
  - Save, restore and free storage for all system objects

- **Security administration** (*SECADM*)
  - Create/change/delete user profiles;

- **Spool control** (*SPLCTL*)
  - Manage all users' spooled files

- **System configuration** (*IOSYSCFG*)
  - Change system configuration

- **System service access** (*SERVICE*)
  - Display and alter service function

### Privilege User Classes

<table>
<thead>
<tr>
<th></th>
<th>Security officer</th>
<th>Security administrator</th>
<th>System operator</th>
<th>Programmer</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>All object access</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auditing control</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Job control</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Save/restore</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security administration</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spool control</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System configuration</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System service access</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© Copyright IBM Corporation 2008
Dedicated service tools (DST) and system service tools (SST) are both used to access service tools and service functions. DST is available when the Licensed Internal Code has been started, even if i5/OS has not been loaded. SST is available from i5/OS.

- Service tools are used to do any of the following:
  - Diagnose server problems
  - Add hardware resources to the server
  - Manage disk units
  - Review the Licensed Internal Code and product activity logs
  - Trace Licensed Internal Code
  - Perform main storage dumps
  - Manage system security
  - Manage other service tools user IDs
  - ...
Module 4

IBM i SST/DST User Profiles

Select one of the following:
1. Service tools user IDs
2. Service tools device IDs
3. Select console
4. Configure service tools LAN adapter

Type option, press Enter.
1=Create 2=Change password 3=Delete
4=Display 5=Enable
7=Change privileges 8=Change description

Opt User ID Description
- -
F3=Exit F6=Clear

Change Service Tools User Privileges

Service tools user ID name ............: PRFLAB5

Type option, press Enter.
1=Revoke 2=Grant

Option Functions Status
- Disk units - operations Revoked
- Disk units - administration Revoked
- Disk units - read only Granted
- System partitions - operations Revoked
- System partitions - administration Revoked
- Partition remote panel key Revoked
- Operator panel functions Revoked
- Operating system initial program load(IPL) Revoked
- Install Revoked
- Performance data collector Revoked
- Hardware service manager Revoked
- Display/Alter/Dump Revoked
- Main storage dump Revoked

F3=Exit F5=Reset F9=Defaults F12=Cancel
SST Example: Locking down security settings

- Security-related system values reflect the implementation of security policies
- Due to the lack of knowledge, many programmers have permissions to change system security settings
- The concept of split responsibilities can prevent high-authority users from changing security system settings
- Security system values can be locked down in System Service Tools (SST)
HMC User Roles

- Task roles define what tasks an HMC account may perform
  - One role is assigned to each user account when the user account is created
- Set of predefined user roles can be customized
Unix password files
- htpasswd is used to create and update the flat-files used to store usernames and password for basic authentication of HTTP users

IBM i Validation lists (*VLDL)
- objects containing lists of user names and passwords
- validation lists are case-sensitive and reside in i5/OS libraries
- cannot be used as user profiles for executing a job on i5/OS
Is it possible to build a secure computer system?

"The answer ... is basically yes. How to build a secure system has been known for decades. MULTICS, designed in the 1960s, for example, had security as one of its main goals and achieved that very well.

Why secure systems are not being built is more complicated, but it comes down to two fundamental reasons. First, current systems are not secure but users are unwilling to throw them out. If Microsoft were to announce that in addition to Windows it had a new product, SecureOS, that was guaranteed to be immune to viruses but did not run Windows applications, it is far from certain that every person and company would drop Windows like a hot potato and buy the new system immediately.

The second is more subtle. The only way to build a secure is to keep it simple. Features are the enemy of security. System designer believe (rightly or wrongly) that what users want is more features. More features mean more complexity, more code, more bugs, and more security errors."

Source: [TAN]