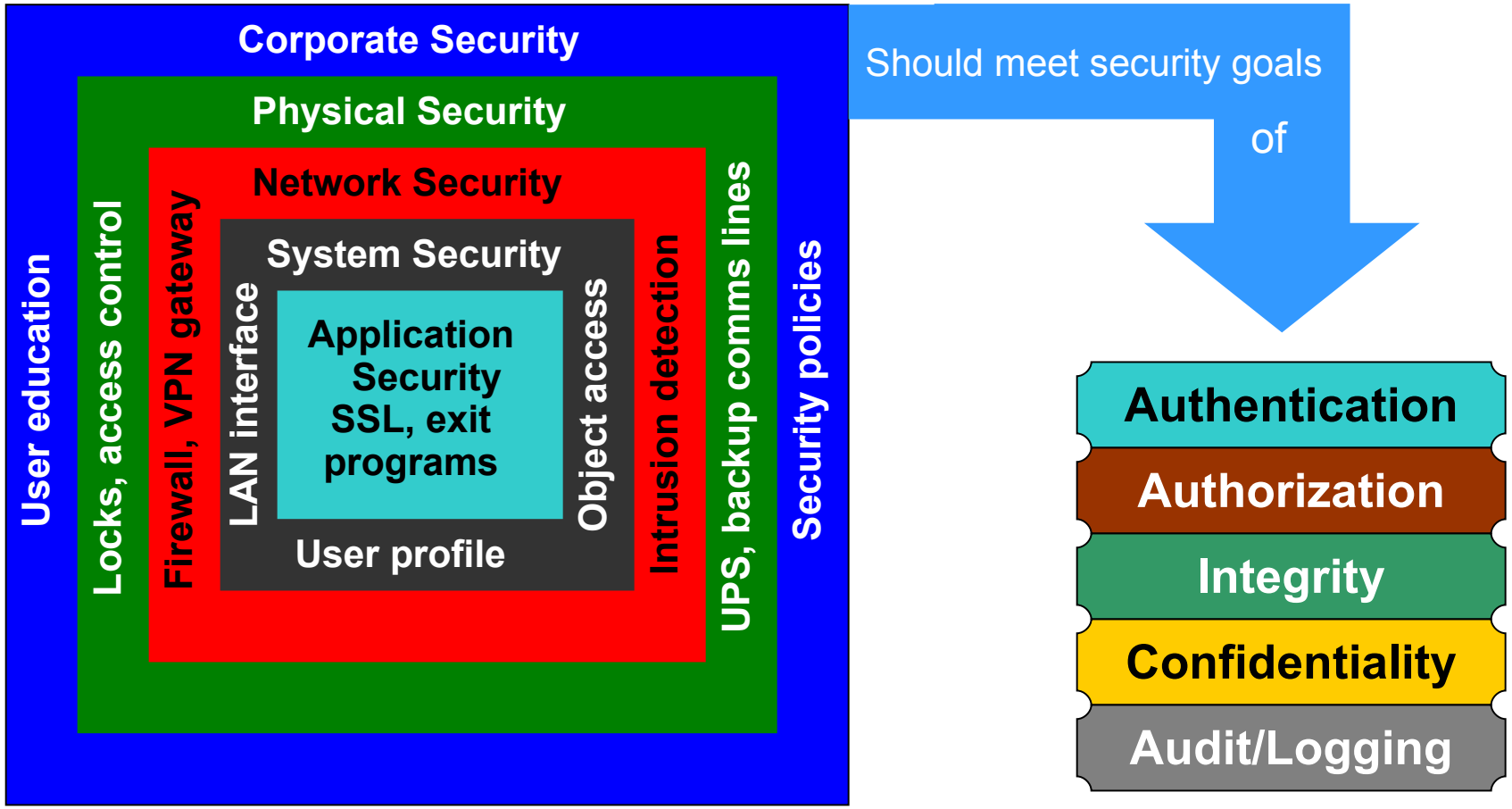
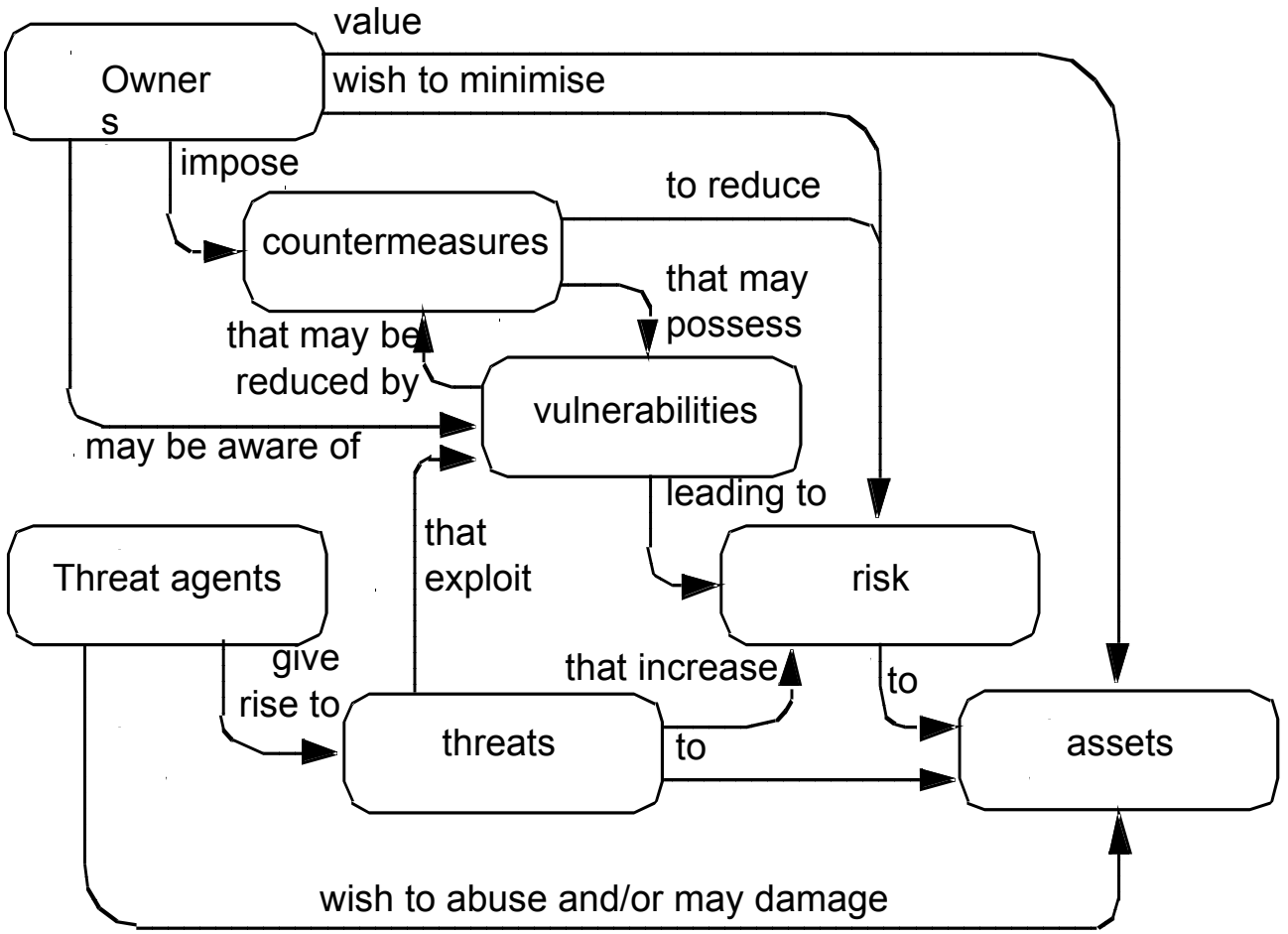
Layered implementation of security



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



Source: Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model

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.

Source: Federal Office for Information Security (BSI), IT Security Guidelines.

Components of Security

- Authentication
- Access control
- Confidentiality
- Data integrity
- Non-repudiation
- Management
- Availability
- Audit

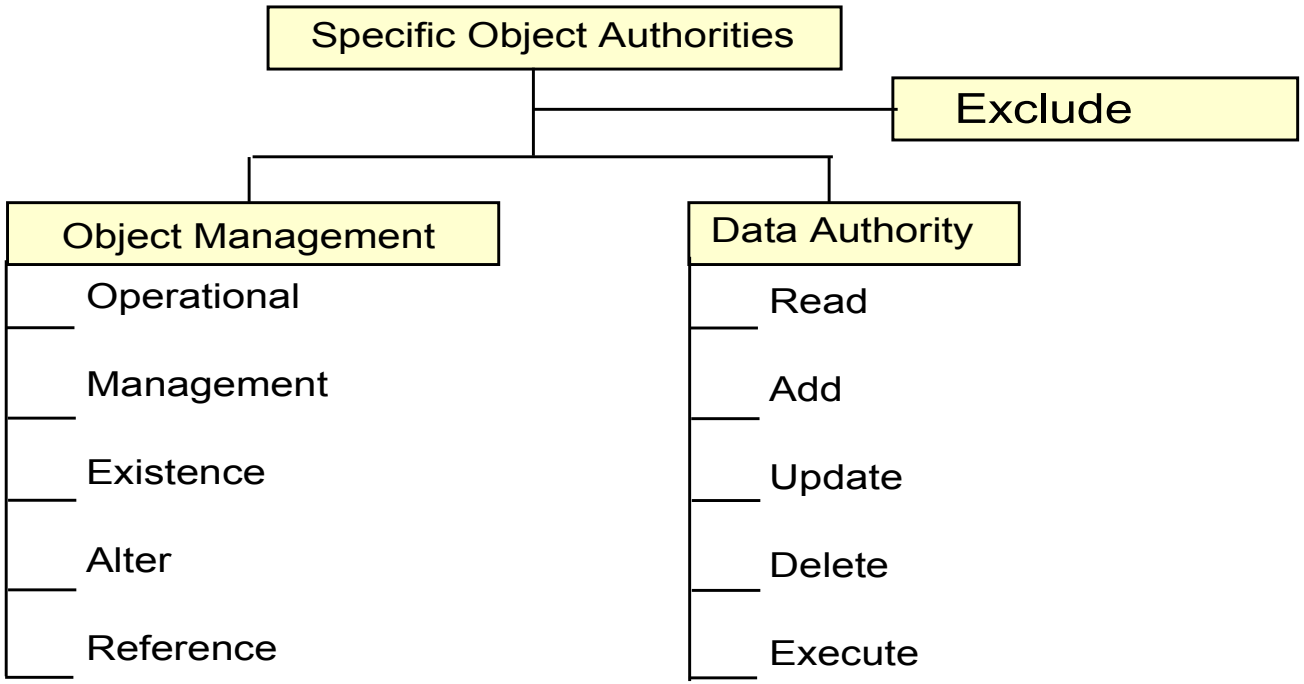
IBM i Resource Security

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:



Object Management Permissions

Permission	Definition
Operational (*OBJOPR)	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.
Management (*OBJMGT)	Authorize users to the object, move or rename the object and add members to database files. All functions defined for *OBJALTER and *OBJREF.
Existence (*OBJEXIST)	Change ownership and delete the object, free storage for the object, perform save and restore operations for the object
Alter (*OBJALTER)	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.
Reference (*OBJREF)	Specify database file as the parent in a referential constraint.
Authorization List (*AUTLMGT)	Add and remove users and their authorities from an authorization list.

Object Data Permissions

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

Commonly Used Permissions

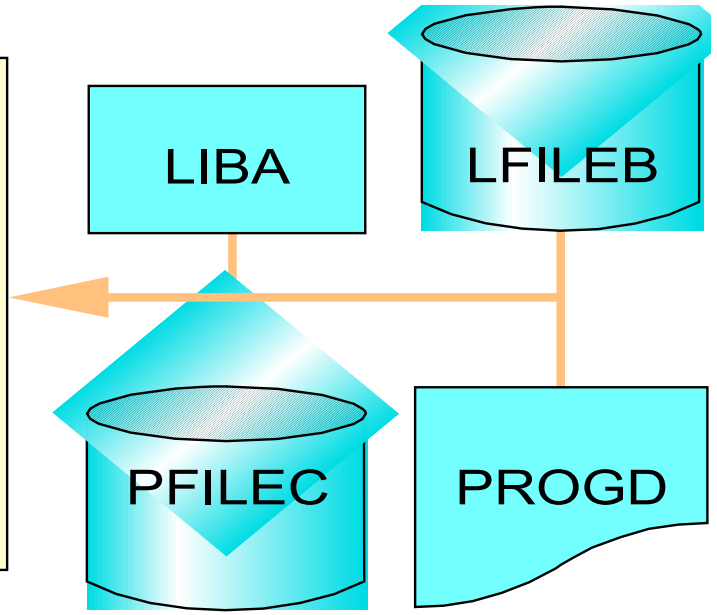
	Object Control					Data Authority				
	Operation	Management	Existence	Alter	Reference	Read	Add	Update	Delete	Execute
*All	X	X	X	X	X	X	X	X	X	X
*Change	X					X	X	X	X	X
*Use	X					X				X
*Exclude										

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

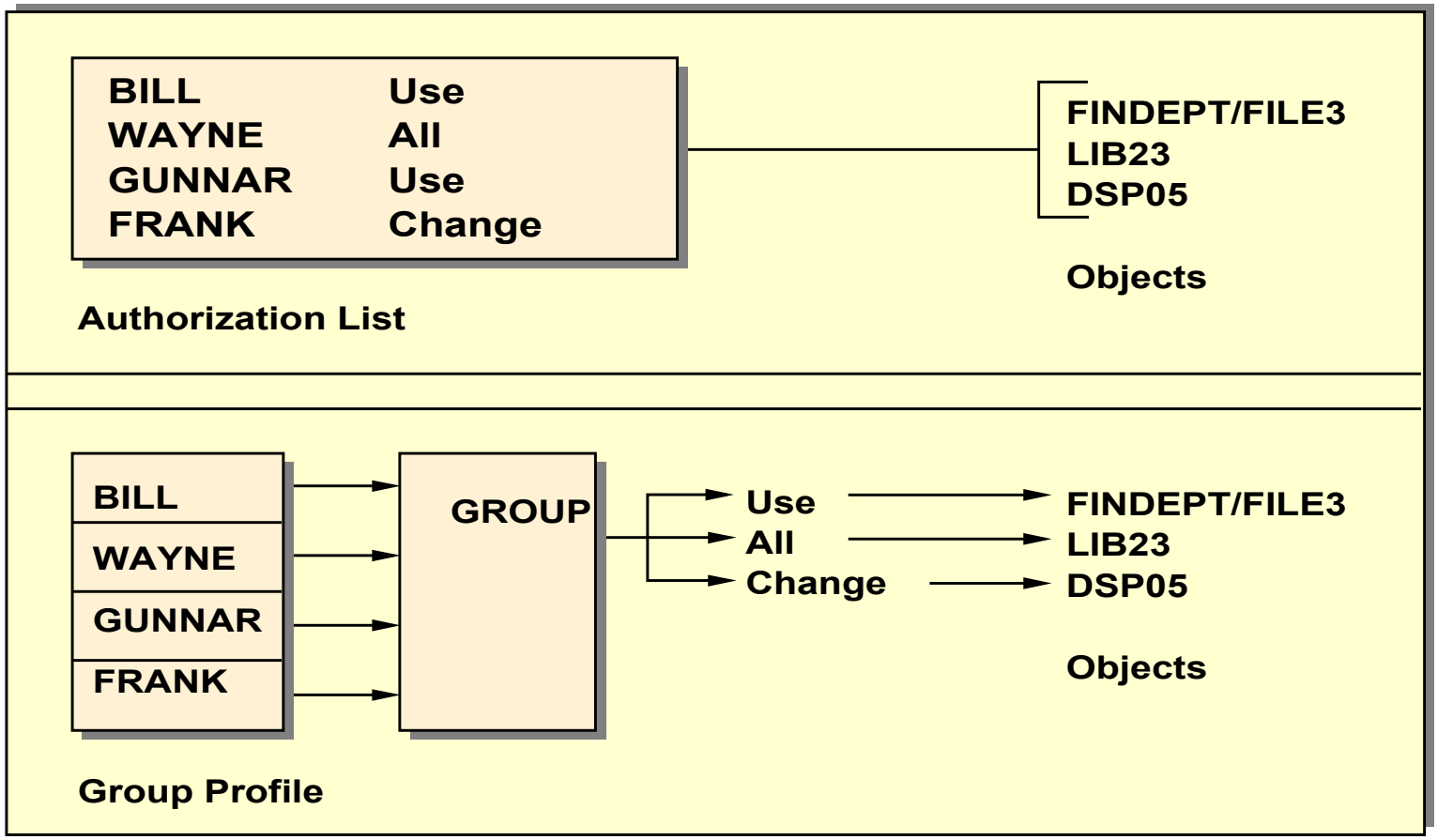
Authorization List

Name:	AUTL1
Owner:	USER3
User Authority:	
***PUBLIC**	**Exclude**
USER1	Use
USER2	Change
USER4	All
USER5	Change



***Public is on all authorization lists**

Authorization List vs. Group Profile



Groups and AUTLs Compared

AUTHORIZATION LISTS	GROUPS
Can secure multiple objects	Can secure multiple objects
A user can be on multiple lists	A user can be a member of 16 groups
Users can have different authority	All users in a group have the same authority
Same authority for different objects using same list	Different authority for different objects
An object can be secured by only one authorization list	An object can be authorized to many groups

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

domain \ object	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

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

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

domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch control
D_3		read	execute					
D_4	write		write		switch			

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.

Adopted Authority (IBM i)

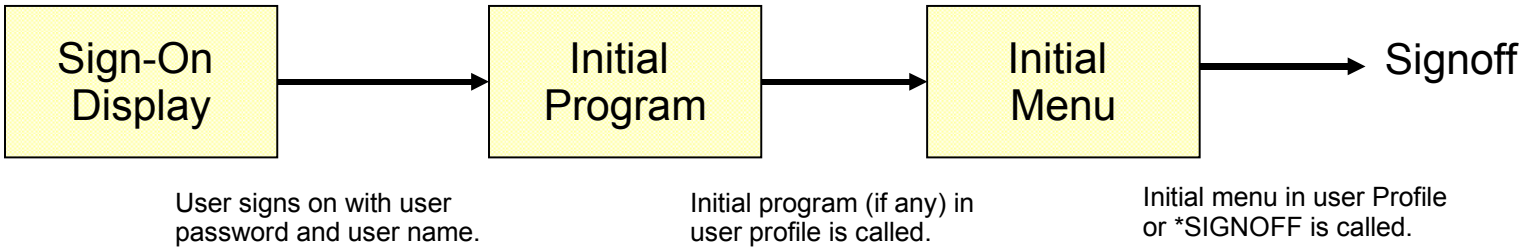
Power Systems

```
                          Display Program Information
_

Program  . . . . . :   STRSBSC       Library  . . . . . :   UNIMASTER
Owner  . . . . . . :   UNIOWNER
Program attribute  . . :   CLP

Program creation information:
  Program creation date/time . . . . . :   05/30/07  08:19:24
  Type of program  . . . . . :   OPM
  Source file  . . . . . :   SOURCES
    Library  . . . . . :     UNIMASTER
  Source member  . . . . . :   STRSBSC
  Source file change date/time . . . . . :   05/30/07  08:19:03
  Observable information . . . . . :   *ALL
  User profile . . . . . :   *OWNER
```

- 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

Power Systems



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

Limit initial program/menu capabilities	Initial Program	Initial Menu	Current Library	Attention Program	Execute Commands
Do not limit	Yes	Yes	Yes	Yes	Yes
Limit some capabilities	No	Yes	No	No	Yes
Limit capabilities	No	No	No	No	No

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)

- Fore 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 Authorithies

Power Systems

- **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

	Security officer	Security administrator	System operator	Programmer	User
All object access	X				
Auditing control	X				
Job control	X		X		
Save/restore	X		X		
Security administration	X	X			
Spool control	X				
System configuration	X				
System service access	X				

IBM i Service Tools

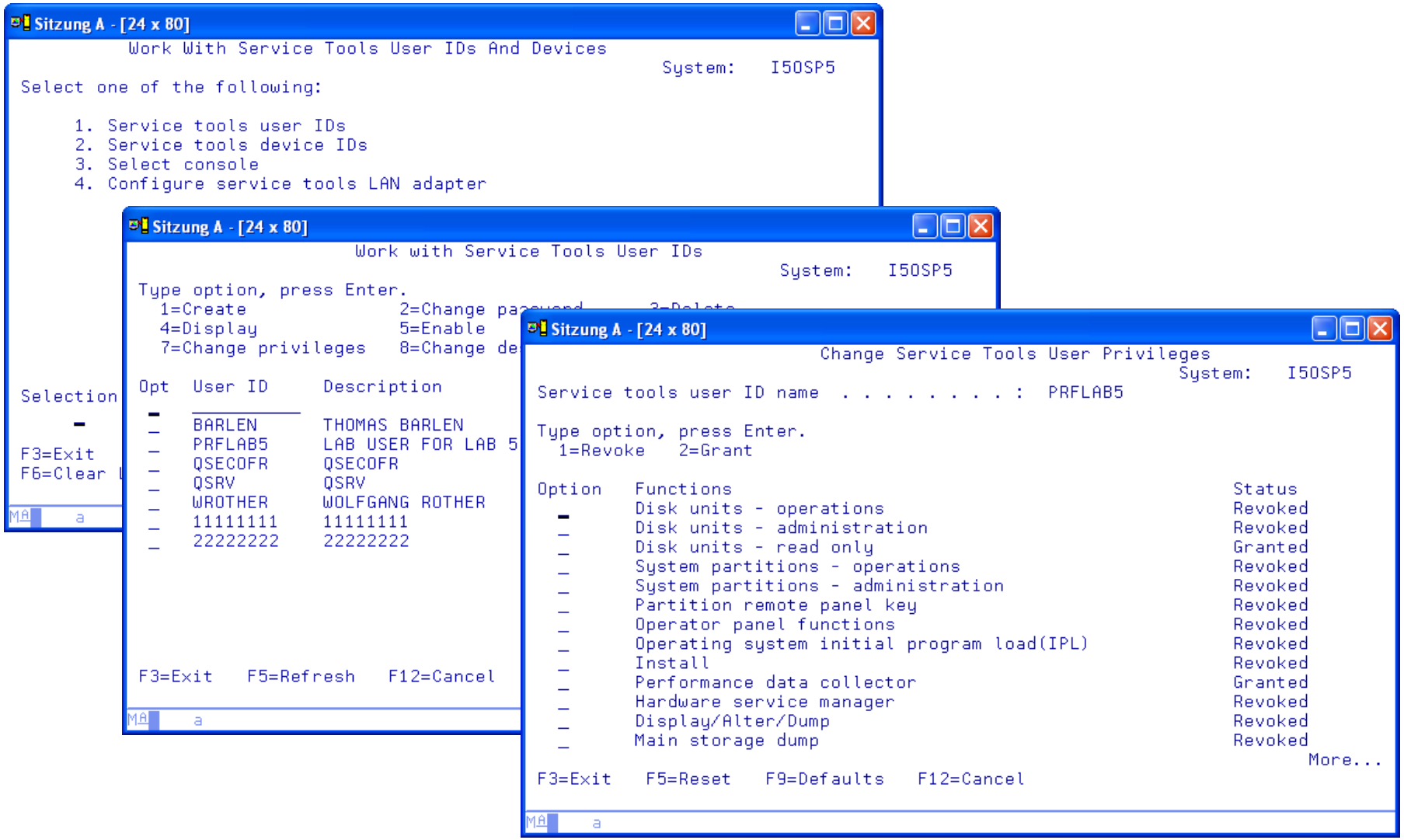
Power Systems

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
  - …

IBM i SST/DST User Profiles

Power Systems



SST Example: Locking down security settings

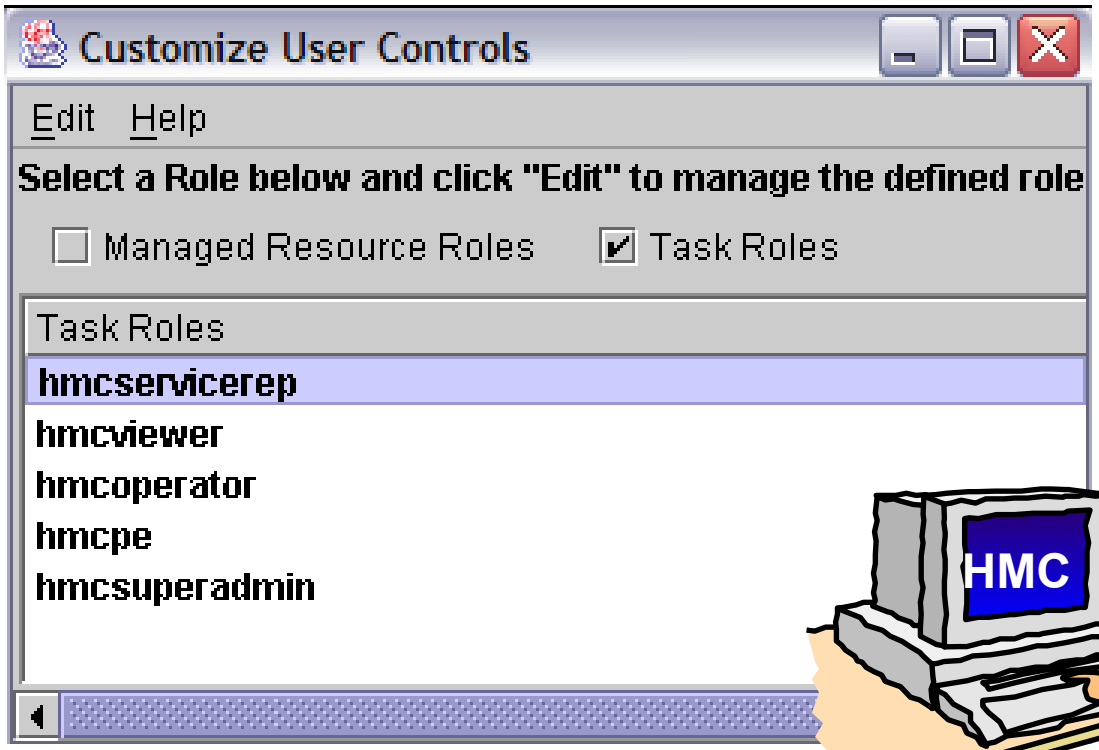
Power Systems

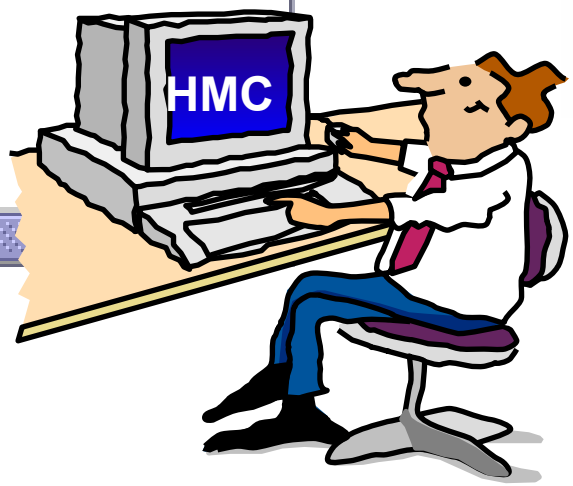
```
                        Work with System Security
                                                              System:   I5OSP3
   Type choices, press Enter.

Allow system value security changes . . . . .            2  1=Yes, 2=No
Allow new digital certificates  . . . . . . .            1  1=Yes, 2=No
Allow a service tools user ID with a
 default and expired password to change
            its own password . . . . . . . . . . . . .   2  1=Yes, 2=No
```

- 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

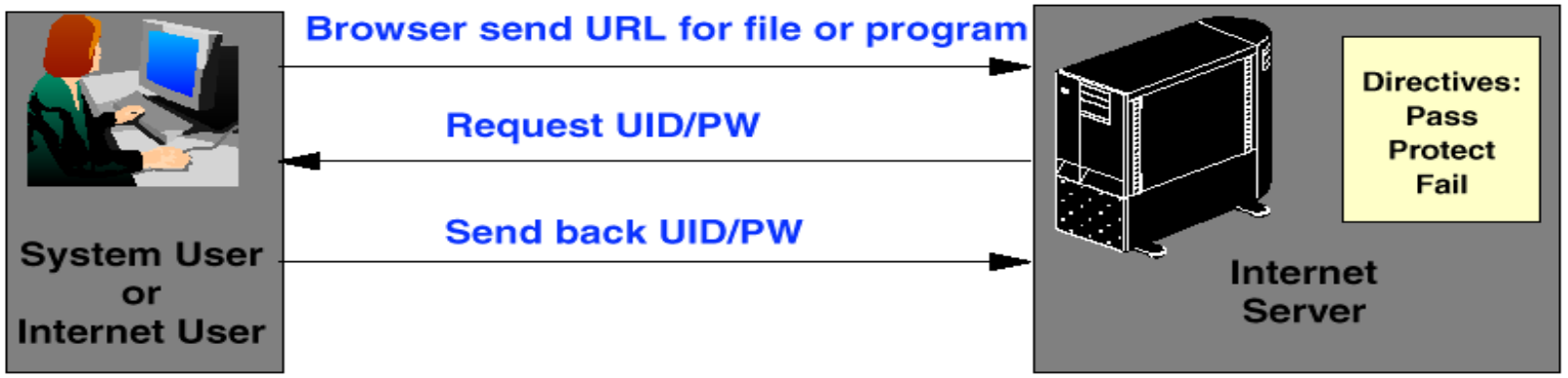
Power Systems





- 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

Internet User



- **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?

Power Systems

„**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 unwillig to throw them out. If Microsoft were to announce that in addition to Windows it had a new product, SecureOS, that was guaranted 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 immediatly.

The second is more subtle. The only way to build a secure is to keep it simple. Features are the emeny 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]