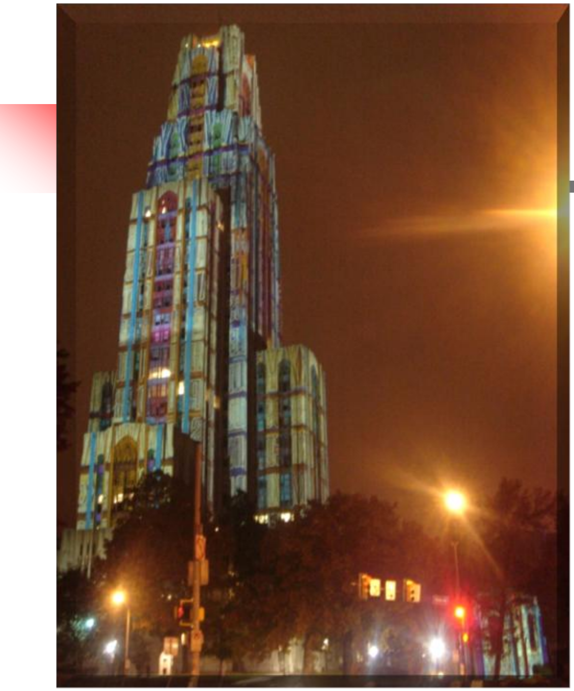
IS 2150 / TEL 2810
Information Security and Privacy



James Joshi

Associate Professor, SIS

Secure Design Principles

OS Security Overview

Lecture 2

Jan 23, 2013

Objectives

- Understand the basic principles of secure system design
- Learn about the basics of access control
- Understand access control in Unix and Windows environment

Some questions

- Should a system be secure by design or can system be made secure after it is built?
- In Unix can you control permissions associated with files when they are created?
- Can you specify that “user A, B and C can read, write and execute, respectively,” your file - in Unix?, in Windows?

Design Principles

Design Principles for Security

- Principles
  - Least Privilege
  - Fail-Safe Defaults
  - Economy of Mechanism
  - Complete Mediation
  - Open Design
  - Separation of Privilege
  - Least Common Mechanism
  - Psychological Acceptability

Overview

- Based on the idea of *simplicity* and *restriction*
    - *Why* Simplicity?
    - *Why* Restriction?

Least Privilege

- A subject should be given only those privileges necessary to complete its task
  - Assignment of privileges based on
    - Function OR Identity-based, … ?
  - Based on "Need to know"; "Relevance to situation" …
    - Examples?
  - Confine processes to "minimal protection domain"

  - How can it be enforced?
    - In Unix? Windows?
    - Challenge? [Complexity?]

Fail-Safe Defaults

- What should be the default action?
- If action fails, how can we keep the system safe/secure?
  - Transactions based systems?
  - When a file is created, what privileges are assigned to it?
    - In Unix? In Windows?

Economy of Mechanism

- Design and implementation of security mechanism
  - KISS Principle (Keep It Simple, Silly!)

- Simpler means?

- Careful design of Interfaces and Interactions

Complete Mediation

- No caching of information
- Mediate all accesses
  - Why?
  - How does Unix read operation work?
  - Any disadvantage of this principle?

Open Design

- Security should not depend on secrecy of design or implementation
  - Source code should be public?
  - “Security through obscurity” ?

  - Does not apply to certain “information”
    - Secrecy of : keys vs encryption algorithm”?
  - What about the “Proprietary software”?

Separation of Privilege

- Restrictive access
  - Use multiple conditions to grant privilege
  - Equivalent to Separation of duty
    - Example?
  - Changing to root account in Berkley-based Unix … need two conditions!

Least Common Mechanism

- Mechanisms should not be shared
  - What is the problem with shared resource?
    - Covert channels?
- Isolation techniques
  - Virtual machine
  - Sandbox

Psychological Acceptability

- Security mechanisms should not add to difficulty of accessing resource
  - Hide complexity introduced by security mechanisms
  - Ease of installation, configuration, use
  - Human factors critical here
    - Proper messages

Access Control - Introduction

ACM Background

- Access Control Matrix
  - Captures the current protection state of a system
- Butler Lampson proposed the first Access Control Matrix model
- Refinements
  - By Graham and Denning
  - By Harrison, Russo and Ulman – with some theoretical results

Protection System

- Subject (S: set of all subjects)
  - Active entities that carry out an action/operation on other entities;
  - Examples?
- Object (O: set of all objects)
  - Examples?
- Right (R: set of all rights)
  - An action/operation that a subject is allowed/disallowed on objects
  - Access Matrix $A: a[s, o] \subseteq R$
- Set of Protection States: (S, O, A)

Access Control Matrix Model

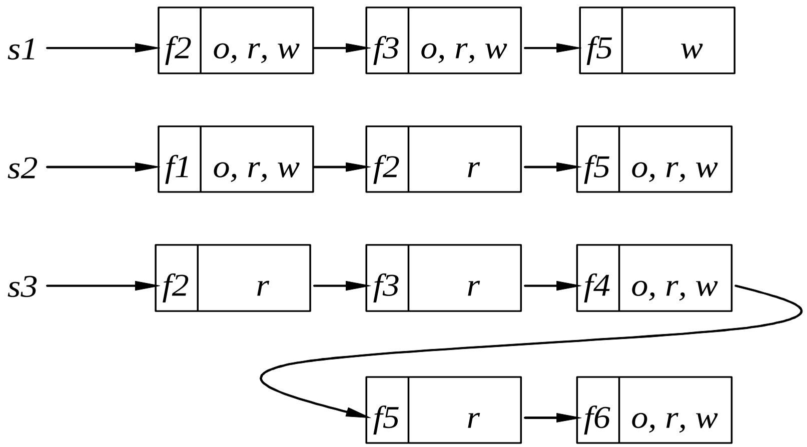
- Access control matrix model
  - Describes the protection state of a system.
  - Elements indicate the access rights that subjects have on objects
  - Is an abstract model - what does it mean?
- ACM implementation
  - What is the disadvantage of maintaining a matrix?
  - Two ways implement:
    - Capability based
    - Access control list

o: own
r: read
w:write

	f1	*f2*	*f3*	*f4*	*f5*	*f6*
s1		*o, r, w*	*o, r, w*		*w*	
s2	*o, r, w*	*r*			*o, r, w*	
s3		*r*	*r*	*o, r, w*	*r*	*o, r, w*

Access Matrix

Capabilities

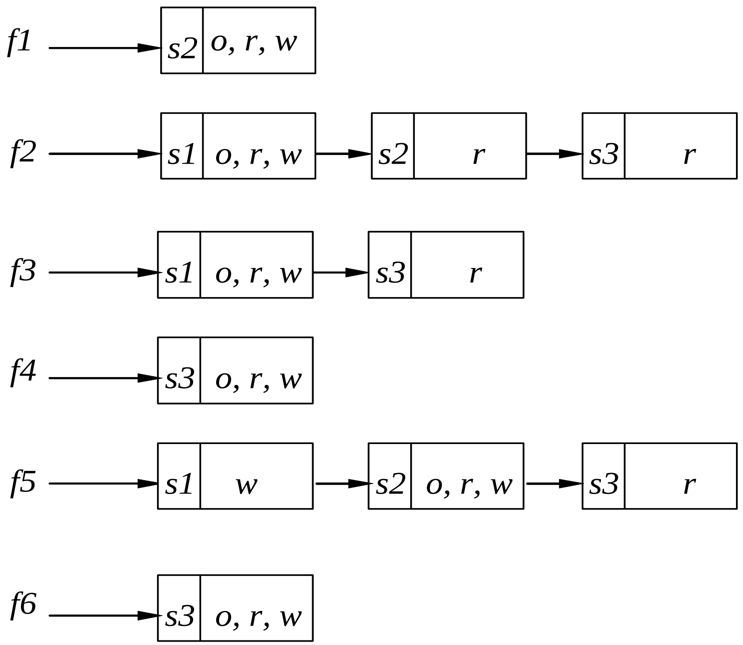


s1 → *f2* *o, r, w* → *f3* *o, r, w* → *f5* *w*

s2 → *f1* *o, r, w* → *f2* *r* → *f5* *o, r, w*

s3 → *f2* *r* → *f3* *r* → *f4* *o, r, w* → *f5* *r* → *f6* *o, r, w*

Access Control List



f1 → *s2* *o, r, w*

f2 → *s1* *o, r, w* → *s2* *r* → *s3* *r*

f3 → *s1* *o, r, w* → *s3* *r*

f4 → *s3* *o, r, w*

f5 → *s1* *w* → *s2* *o, r, w* → *s3* *r*

f6 → *s3* *o, r, w*

Access Control Matrix

Hostnames	*Telegraph*	*Nob*	*Toadflax*
Telegraph	*own*	*ftp*	*ftp*
Nob		*ftp, nsf, mail, own*	*ftp, nfs, mail*
Toadflax		*ftp, mail*	*ftp, nsf, mail, own*

- *telegraph* is a PC with ftp client but no server
- *nob* provides NFS but not to Toadfax
- *nob* and *toadfax* can exchange mail

	Counter	*Inc_ctr*	*Dcr_ctr*	*Manager*
Inc_ctr	+			
Dcr_ctr	-			
manager		*Call*	*Call*	*Call*

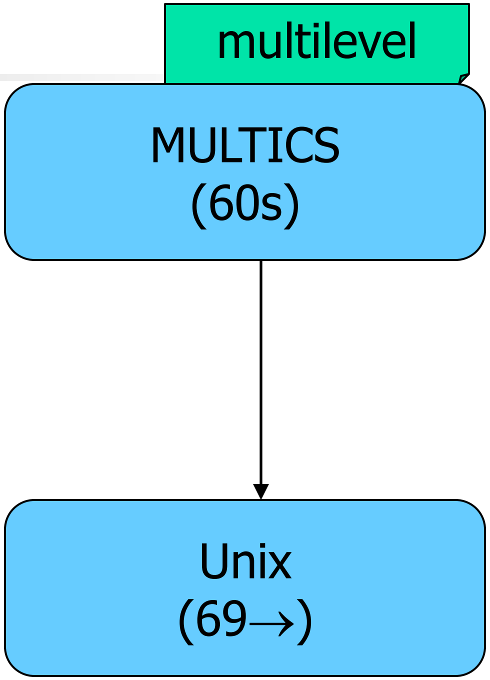
Unix Security Overview

Unix

- Kernel
  - I/O, Load/Run Programs, Filesystem; Device Drivers …
- Standard Utility Programs
  - /bin/ls, /bin/cp, /bin/sh
- System database files
  - E.g, /etc/passwd; /etc/group

(interacts with)

Security Policy



multilevel

MULTICS
(60s)

Unix
(69→)

Multi-user
Multi-tasking

Developed at
AT&T Bell Labs

Users and password

- Each user has a
  - unique *account* identified by a *username*
  - Each *account* has a *secret password*
    - Standard: 1-8 characters; but varies
    - Passwords could be same – bad choice!
- /etc/passwd contains
  - Username, Identification information
  - Real name, Basic account information

```
root:x:0:1:System Operator:/:/bin/ksh
daemon:x:1:1::/tmp:
uucp:x:4:4::/var/spool/uucppublic:/usr/lib/uucp/uucico
rachel:x:181:100:Rachel Cohen:/u/rachel:/bin/ksh
arlin:x.:182:100:Arlin Steinberg:/u/arlin:/bin/csh
```

Account info

Field	Contents
rachel	Username.
x	Holding place for the user's "encrypted password." Newer Unix systems store encrypted passwords in a separate file (the *shadow password file*) that can be accessed only by privileged users.
181	User's user identification number (UID).
100	User's group identification number (GID).
Rachel Cohen	User's full name
/u/rachel	User's home directory.
/bin/ksh	User's shell (empty field means default shell)

```
rachel:x:181:100:Rachel Cohen:/u/rachel:/bin/ksh
```

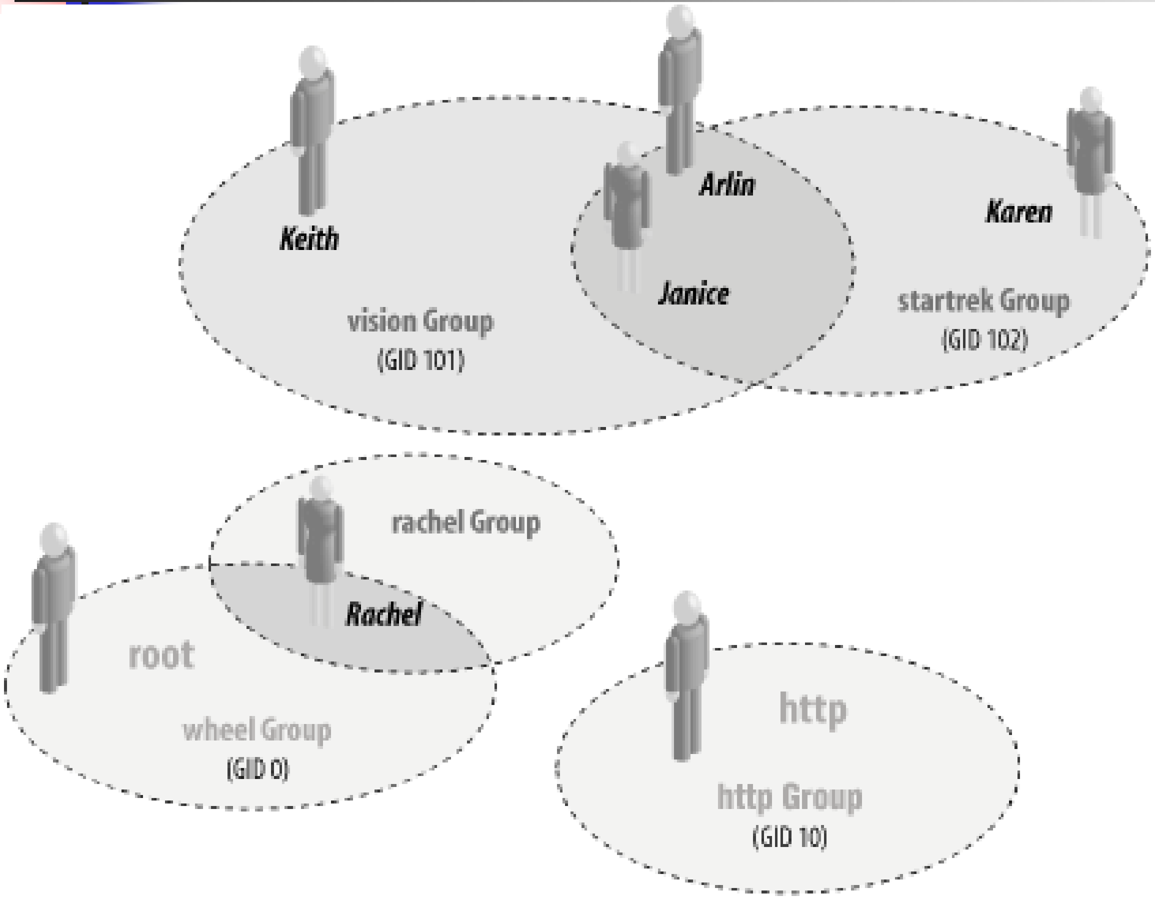
Users and Groups

- Each user is uniquely identified by a UID
  - Special user names
    - Root; Bin; Daemon; Mail; Guest; ftp
- Every user belongs to one or more groups
  - A *primary group*
  - */etc/group*
    - Gname, Gpassword, GID, Users

16 bits: How many IDs?
UID 0: superuser
(More bits too)

```
wheel:*:0:root,rachel
http:*:10:http
users:*:100:
vision:*:101:keith,arlin,janice
startrek:*:102:janice,karen,arlin
rachel:*:181:
```

Users and Groups



Some useful commands

- groups
- id
- newgrp
- su

```
wheel:*:0:root,rachel
http:*:10:http
users:*:100:
vision:*:101:keith,arlin,janice
startrek:*:102:janice,karen,arlin
rachel:*:181:
```

Superuser

- root; UID = 0 …….. Complete Control
  - Used by OS itself for basic functions
    - Logging in/out users
    - Recording accounting info
    - Managing input/output devices
  - Security controls are bypassed
  - There are few things not allowed
    - Decrypt passwords shadow-file, …

Key Security Weakness in Unix

Processes can run with Effective UID = 0

User ids

- Each process has three Ids
  - Real user ID (RUID)
    - a user’s “real identity”
    - same as the user ID of parent (unless changed)
  - Effective user ID (EUID)
    - from set user ID (SUID) bit on the file being executed
    - Can use su command to assume another’s RUID
  - Saved user ID (SUID)
    - Allows restoring previous EUID

- Similar for Group

- While accessing files
  - Process EUID compared against the file UID
  - GIDs are compared; then Others are tested

A quick question …

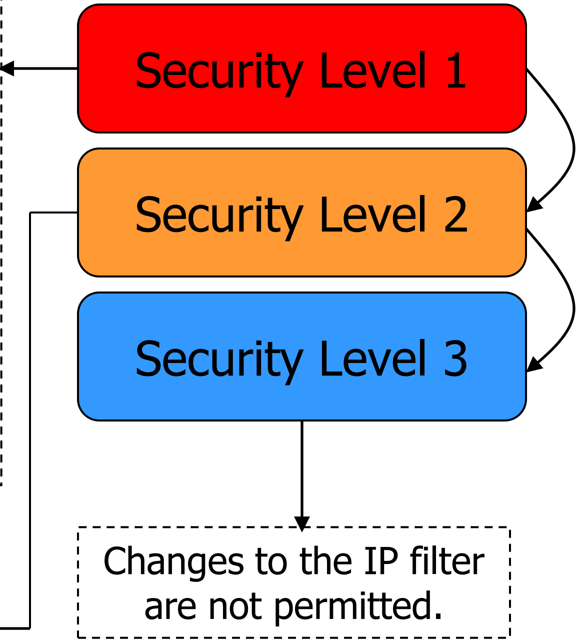
One should always use the full path /ls/su if changing to root
… WHY?

Kernel security Levels (BSD, Mac OS ..)

Restricts power of superuser

```
sysctl kern.securelevel=1
```

- Write access to the raw disk partitions is prohibited.
- Raw access to the SCSI bus controller is prohibited.
- Files that have the immutable flag set cannot be changed. Files that have the append-only bit set can only be appended to, and not otherwise modified or deleted.
- The contents of IP packets cannot be logged.
- Raw I/O to the system console is prohibited.
- Raw writes to system memory or I/O device controllers from user programs are prohibited.
- Additional kernel modules cannot be loaded.
- The system clock cannot be set backwards.



Security Level 1

Security Level 2

Security Level 3

Changes to the IP filter are not permitted.

Reads from raw disk partitions are not permitted.

Not a comprehensive list

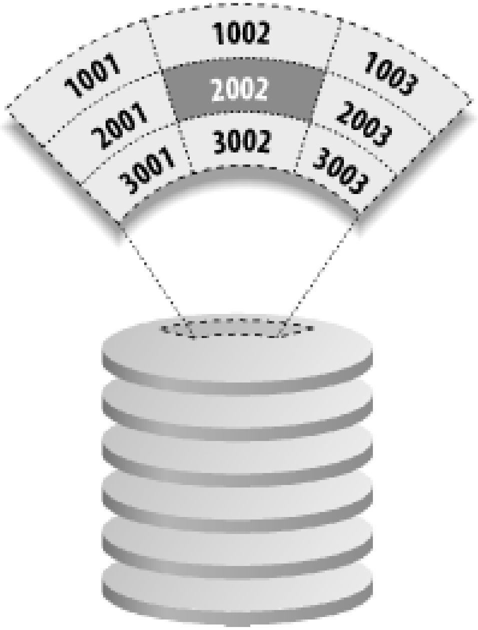
Unix file system

- File systems store
  - information in files and metadata about files.
  - tree-structured

A file is a block of information that is given a single name and can be acted upon with a single operation.

"everything is a file"

Finenames stored in directory and Have pointers to *inodes*



inode 2002

Item Location	Item Type	Item Size *(bytes)*
Time Inode Modified *(ctime)*	Time Contents Modified *(mtime)*	Time File Accessed *(atime)*
File's Owner *(UID)*	File's Group *(GID)*	Per-missions *(mode bits)*
Reference Count	Location of Data on Disk	

Directory

- A Unix directory is
  - a list of names
    - files, directories,.
  - associated inode numbers.
  - Special entries

    "." and its inode # (self)

    ".." and its inode # (parent)

r	Read	Listing files in the directory.
w	Write	?
x	Execute	?

Unix file security

- Each file/directory has owner and group
- How are the permissions set by a owner for
  - Read, write, execute
  - Owner, group, other ???
- Only owner, root can change permissions
  - This privilege cannot be delegated or shared

Unix File Permissions

- File type, owner, group, others

```
drwx------   2 jjoshi  isfac  512  Aug 20  2003 risk management
lrwxrwxrwx   1 jjoshi  isfac   15  Apr  7 09:11 risk_m->risk management
-rw-r--r--   1 jjoshi  isfac 1754  Mar  8 18:11 words05.ps
-r-sr-xr-x   1 root    bin   9176  Apr  6  2002 /usr/bin/rs
-r-sr-sr-x   1 root    sys   2196  Apr  6  2002 /usr/bin/passwd
```

- File type: regular -, directory d, symlink l, device b/c, socket s, fifo f/p
- Permissions: r, w, x
- Any other permissions?

Umask

- Specifies the permission you do not want given by default to new files
  - Bitwise AND with the bitwise complement of the umask value

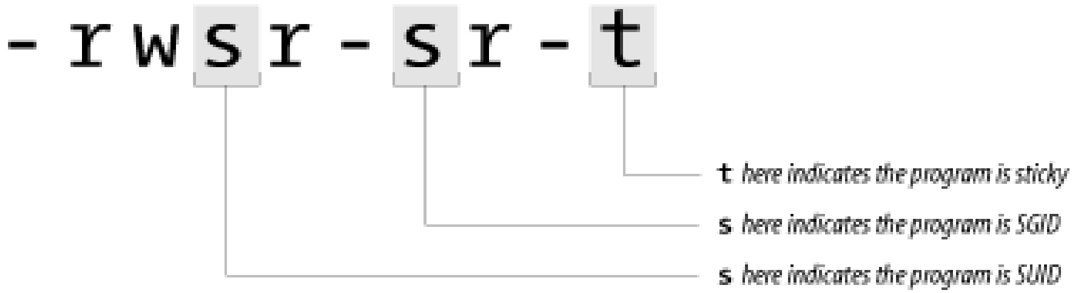
Umask	User Access	Group Access	Other Access
0000	All	All	All
0002	All	All	Read, Execute
0007	All	All	None
0022	All	Read, Execute	Read, Execute
0027	All	Read, Execute	None
0077	All	None	None

IDs/Operations

- Root can access any file
- Fork and Exec
  - Inherit three IDs,
    - except exec of file with setuid bit
- Setuid system calls
  - seteuid(newid) can set EUID to
    - Real ID or saved ID, regardless of current EUID
    - Any ID, if EUID=0
  - Related calls: setuid, seteuid, setgid, setegid

Setid bits

- Three setid bits
  - suid
    - set EUID of process to ID of file owner
  - sgid
    - set EGID of process to GID of file
  - suid/sgid used when a process executes a file
    - If suid(sgid) bit is on – the EUID (EGID) of the process changed to UID (GUID) of the file
  - Sticky
    - Off: if user has write permission on directory, can rename or remove files, even if not owner
    - On: only file owner, directory owner, and root can rename or remove file in the directory

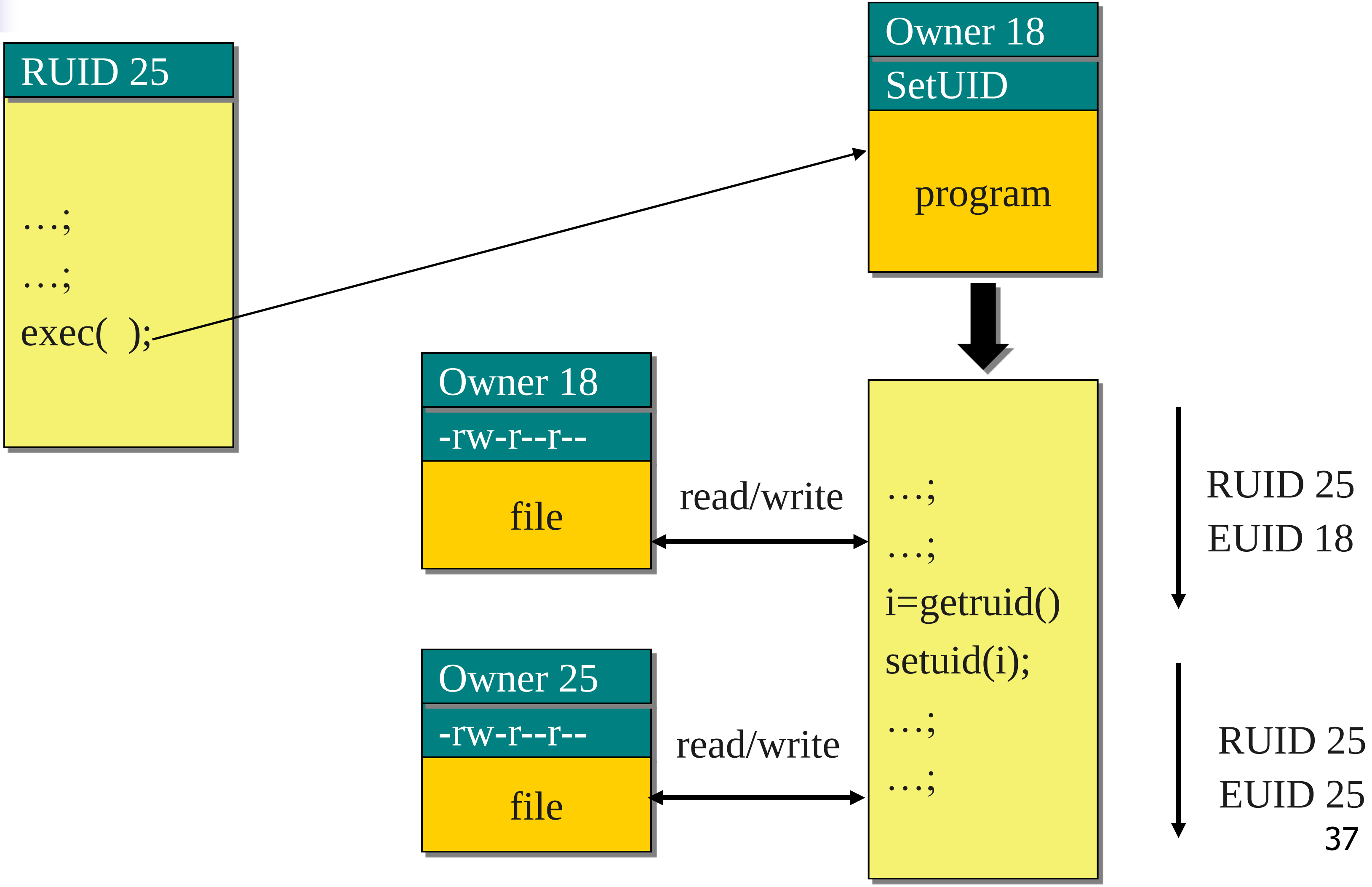


-rwsr-sr-t
t here indicates the program is sticky
s here indicates the program is SGID
s here indicates the program is SUID

What does this mean?

```
-r--r-Sr-T 1 root user 12324 Mar 26 1995 /tmp/example
```

SUID – dangerous!



Careful with Setuid !

- Can do what owner of file is allowed to do
- Be sure not to
  - Take action for untrusted user
  - Return secret data to untrusted user

- Principle of least privilege
  - change EUID when root privileges no longer needed

  - Do not leave unattended sh terminals !!

Windows NT

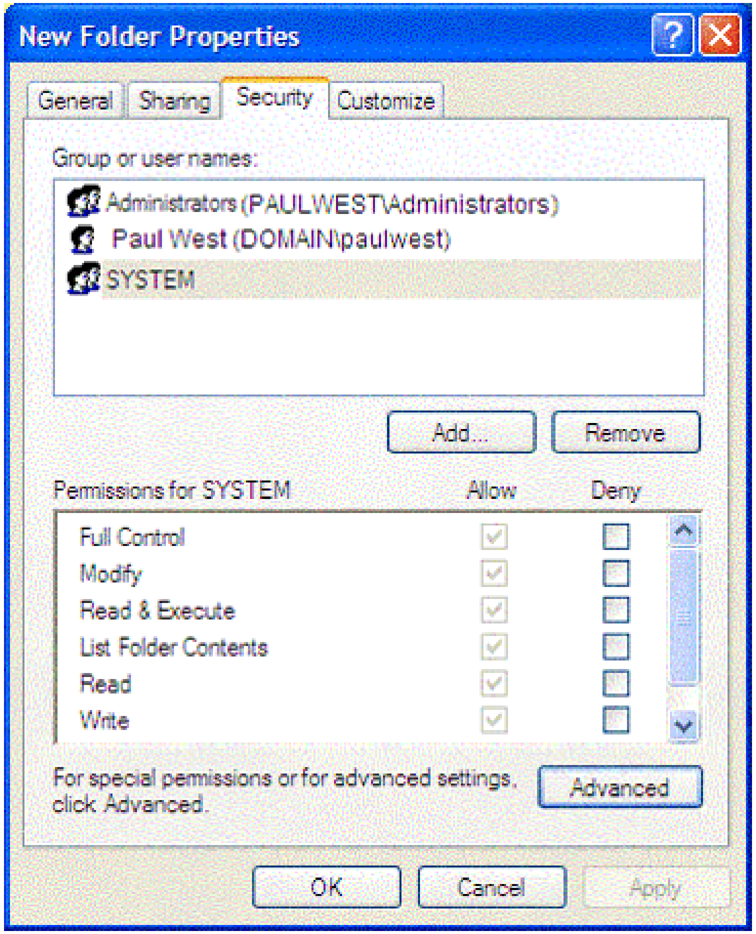
- Windows 9x, Me
  - Never meant for security
  - FAT file system – no file level security
  - PWL password scheme – not secure
    - Can be simply deleted
- Windows NT
  - Username mapped to Security ID (SID)
  - SID is unique within a domain
    - SID + password stored in a database handled by the Security Accounts Manager (SAM) subsystem

Windows NT

- 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 give some but not all administrator privileges

Sample permission options

- 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



Group or user names:

Administrators (PAULWEST\Administrators)
Paul West (DOMAIN\paulwest)
SYSTEM

Permissions for SYSTEM	Allow	Deny
Full Control	☑	□
Modify	☑	□
Read & Execute	☑	□
List Folder Contents	☑	□
Read	☑	□
Write	☑	□

For special permissions or for advanced settings, click Advanced.

Permission Inheritance

- Static permission inheritance (Win NT)
  - Initially, subfolders inherit permissions of folder
  - Folder, subfolder changed independently
  - *Replace Permissions on Subdirectories* command
    - Eliminates any differences in permissions

Permission Inheritance

- Dynamic permission inheritance (Win 2000)
  - Child inherits parent permission, remains linked
  - Parent changes are inherited, except explicit settings
  - Inherited and explicitly-set permissions may conflict
    - Resolution rules
      - Positive permissions are additive
      - Negative permission (deny access) takes priority

Tokens

- Security context
  - privileges, accounts, and groups associated with the process or thread
- Security Reference Monitor
  - uses tokens to identify the security context of a process or thread
- Impersonation token
  - Each thread can have two tokens – primary & impersonation
  - thread uses temporarily to adopt a different security context, usually of another user

Security Descriptor

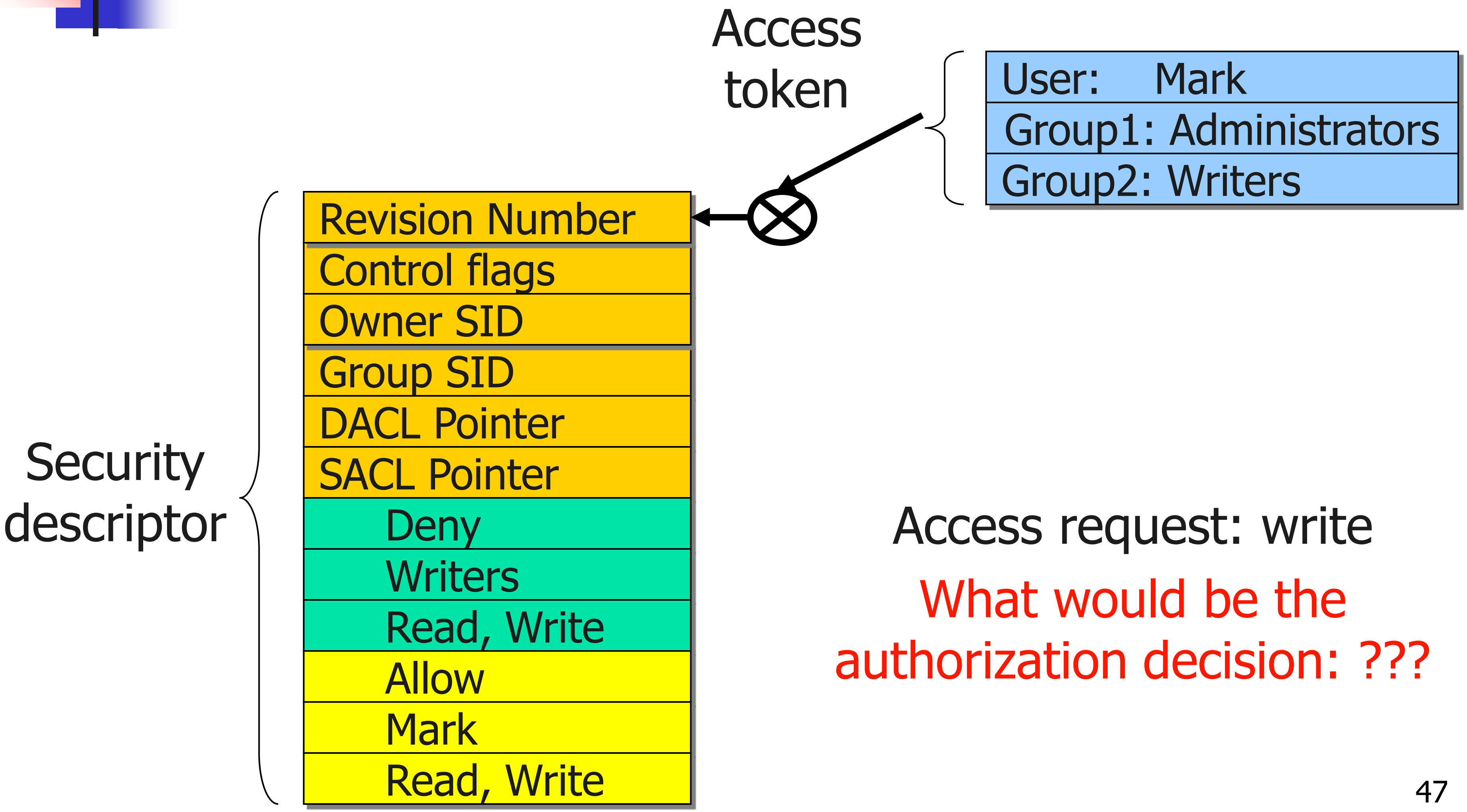
- Information associated with an object
  - who can perform what actions on the object
- Several fields
  - Header
    - Descriptor revision number
    - Control flags, attributes of the descriptor
      - E.g., memory layout of the descriptor
  - SID of the object's owner
  - SID of the primary group of the object
  - Two attached optional lists:
    - Discretionary Access Control List (DACL) – users, groups, …
    - System Access Control List (SACL) – system logs, ..

Using ACEs in DACL

One of the following need to occur:

1. If access-denied for any requested permission – DENY
2. If access-allowed through one or more ACEs for trustees listed – GRANT
3. All ACEs have been checked – but there is still one permission that has not been allowed - DENY

Example access request



Access token

User: Mark

Group1: Administrators

Group2: Writers

Security descriptor

Revision Number

Control flags

Owner SID

Group SID

DACL Pointer

SACL Pointer

Deny

Writers

Read, Write

Allow

Mark

Read, Write

Access request: write

What would be the authorization decision: ???

Impersonation Tokens (setuid?)

- Process uses security attributes of another
  - Client passes impersonation token to server
- Client specifies impersonation level of server
  - Anonymous
    - Token has no information about the client
  - Identification
    - server obtains the SIDs of client and client's privileges, but server cannot impersonate the client
  - Impersonation
    - server identifies and impersonates the client
  - Delegation
    - lets server impersonate client on local, remote systems

Mandatory Access Control

- Integrity controls
  - Limit operations that might change the state of an object
  - Objects and subjects – integrity levels
    - Low, Medium, High, System
    - SIDs in token would include the level info

  - Process with Medium integrity should be able to write to Objects with what integrity level?

Encrypted File Systems (EFS)

- Store files in encrypted form
  - Key management: user’s key decrypts file
  - Useful protection if someone steals disk
- Windows – EFS
  - User marks a file for encryption
  - Unique file encryption key is created
  - Key is encrypted, can be stored on smart card

SELinux Security Policy Abstractions

- Type enforcement
  - Each process has an associated domain
  - Each object has an associated type
  - Configuration files specify
    - How domains are allowed to access types
    - Allowable interactions and transitions between domains
- Role-based access control
  - Each process has an associated role
    - Separate system and user processes
  - configuration files specify
    - Set of domains that may be entered by each role

Sample Features of Trusted OS

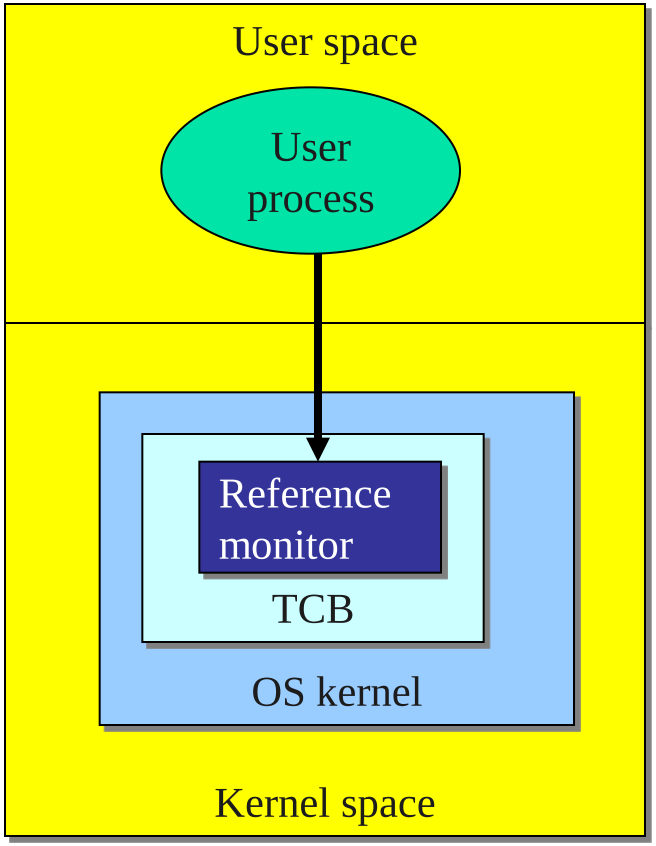
- Identification and authentication
- Mandatory access control
  - MAC not under user control, precedence over DAC
- Object reuse protection
  - Write over old data when file space is allocated
- Complete mediation
  - Prevent any access that circumvents monitor
- Audit
  - Log security-related events
- Intrusion detection
  - Anomaly detection
    - Learn normal activity, Report abnormal actions
  - Attack detection
    - Recognize patterns associated with known attacks

Kernelized Design

- Trusted Computing Base
  - Hardware and software for enforcing security rules
- Reference monitor
  - Part of TCB
  - All system calls go through reference monitor for security checking
- Reference validation mechanism –
  1. Tamperproof
  2. Never be bypassed
  3. Small enough to be subject to analysis and testing – the completeness can be assured

↑

Which principle(s)?



Is Windows “Secure”?

- Good things
  - Design goals include security goals
  - Independent review, configuration guidelines
- But …
  - “Secure” is a complex concept
    - What properties protected against what attacks?
  - Typical installation includes more than just OS
    - Many problems arise from applications, device drivers
    - Windows driver certification program

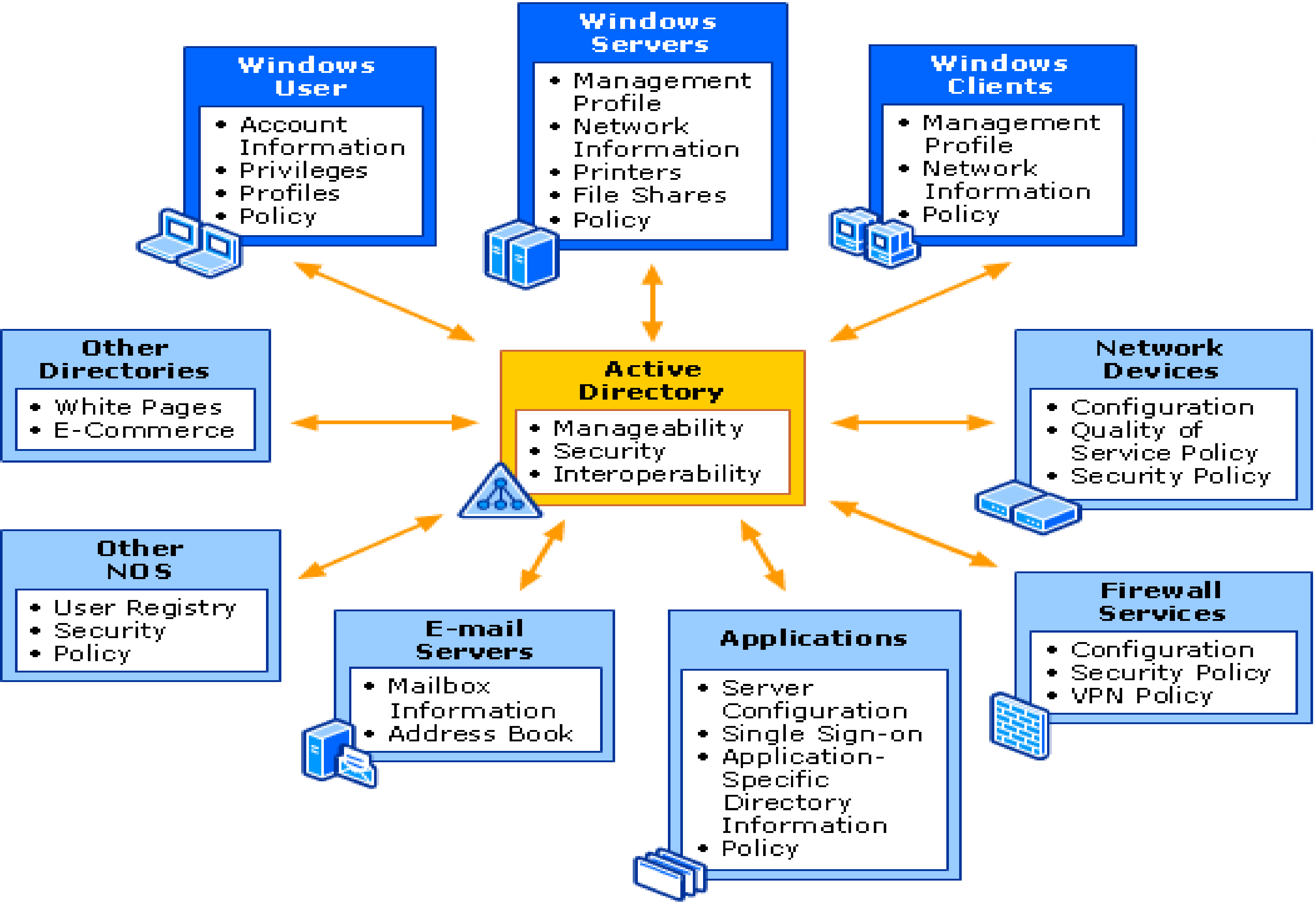
Window 2000

- Newer features than NT
- NTFS file system redesigned for performance
- Active directory
  - Kerberos for authentication
  - IPSec/L2TP

Active Directory

- Core for the flexibility of Win2000
  - Centralized management for clients, servers and user accounts
- Information about all objects
- Group policy and remote OS operations
- Replaces SAM database
  - AD is trusted component of the LSA
- Stores
  - Access control information – authorization
  - User credentials – authentication
- Supports
  - PKI, Kerberos and LDAP

Win 2003



Windows User

- Account Information
- Privileges
- Profiles
- Policy

Windows Servers

- Management Profile
- Network Information
- Printers
- File Shares
- Policy

Windows Clients

- Management Profile
- Network Information
- Policy

Other Directories

- White Pages
- E-Commerce

Active Directory

- Manageability
- Security
- Interoperability

Network Devices

- Configuration
- Quality of Service Policy
- Security Policy

Other NOS

- User Registry
- Security
- Policy

E-mail Servers

- Mailbox Information
- Address Book

Applications

- Server Configuration
- Single Sign-on
- Application-Specific Directory Information
- Policy

Firewall Services

- Configuration
- Security Policy
- VPN Policy