

COP 4610

Operating System Principles

Protection

1

Overview

- Goals of Protection
- Principles of Protection
- Domain of Protection
- Access Matrix
- Implementation of Access Matrix
- Access Control
- Revocation of Access Rights
- Capability-Based Systems
- Language-Based Protection

2

Objectives

- Discuss the **goals and principles** of **protection** in a modern computer system
- Explain how protection domains combined with an **access matrix** are used to specify the resources a process may access
- Examine **capability-based** protection systems

COP 4610 – Operating System Principles

3

3

Goals of Protection

- In one protection model, a computer consists of a **collection** of objects, hardware or software
- Each object has a **unique name** and can be accessed through a **well-defined set of operations**
- **Protection problem** - ensure that each object is accessed **correctly** and only by those processes that are **allowed** to do so

COP 4610 – Operating System Principles

4

4

Principles of Protection

- Guiding principle – **principle of least privilege**
 - Static
 - Dynamic - **domain switching, privilege escalation**
 - “Need to know” a similar concept regarding access to data
 - “Containment of failure”
- Must consider “grain” aspect
 - Rough-grained
 - Fine-grained
- Domain can be user, process, procedure

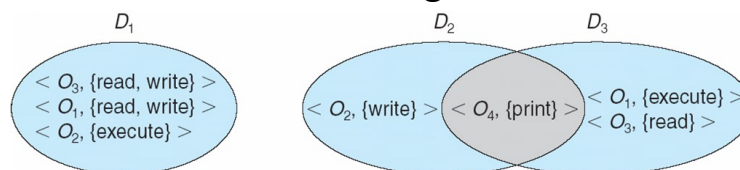
COP 4610 – Operating System Principles

5

5

Domain Structure

- Access-right = $\langle \text{object-name}, \text{rights-set} \rangle$
where *rights-set* is a subset of all valid operations that can be performed on the object
- Domain = set of access-rights



COP 4610 – Operating System Principles

6

6

Domain Implementation (UNIX)

- Domain = user-id
- 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 (in a similar fashion "setgid")
 - When execution completes user-id is reset
- Domain switch accomplished via passwords
 - su command temporarily switches to another user's domain when other domain's password provided
- Domain switching via commands
 - sudo command prefix executes specified command in another domain (if original domain has privilege or password given)

COP 4610 – Operating System Principles

7

7

Sudo (Debian/Ubuntu)

- `usermod -aG sudo [username]`
- `[sudo] apt-get update`

```
dejan@dejan-phoenixnap:~$ apt-get update
Reading package lists... Done
E: Could not open lock file /var/lib/apt/lists/lock - open (13: Permission denied)
E: Unable to lock directory /var/lib/apt/lists/
W: Problem unlinking the file /var/cache/apt/pkgcache.bin - RemoveCaches (13: Permission denied)
W: Problem unlinking the file /var/cache/apt/srcpkgcache.bin - RemoveCaches (13: Permission denied)
dejan@dejan-phoenixnap:~$ sudo apt-get update
Get:1 http://repo.mysql.com/apt/ubuntu focal InRelease [12,2 kB]
Get:2 https://artifacts.elastic.co/packages/7.x/apt stable InRelease [10,4 kB]
Err:1 http://repo.mysql.com/apt/ubuntu focal InRelease
  The following signatures were invalid: EXPKEYSIG 8C7180385072E1F5 MySQL Release Engineering <mysql-build@oss.oracle.com>
Hit:3 http://rs.archive.ubuntu.com/ubuntu focal InRelease
Get:4 http://security.ubuntu.com/ubuntu focal-security InRelease [107 kB]
Get:5 http://rs.archive.ubuntu.com/ubuntu focal-updates InRelease [111 kB]
Get:6 https://artifacts.elastic.co/packages/7.x/apt stable/main amd64 Packages [40,2 kB]
Get:7 http://rs.archive.ubuntu.com/ubuntu focal-backports InRelease [98,3 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [157 kB]
Get:9 https://artifacts.elastic.co/packages/7.x/apt stable/main i386 Packages [32,6 kB]
Get:10 http://rs.archive.ubuntu.com/ubuntu focal-updates/main i386 Packages [191 kB]
Get:11 http://rs.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [320 kB]
Get:12 http://rs.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [121 kB]
Get:13 http://rs.archive.ubuntu.com/ubuntu focal-updates/main amd64 DEP-11 Metadata [196 kB]
Get:14 http://security.ubuntu.com/ubuntu focal-security/main i386 Packages [59,5 kB]
Get:15 http://security.ubuntu.com/ubuntu focal-security/main Translation-en [56,4 kB]
```

8

8

Sudo

```
dejan@dejan-phoenixnap:~$ whoami
dejan
dejan@dejan-phoenixnap:~$ sudo -u root whoami
root
dejan@dejan-phoenixnap:~$
```

9

Password Example

```
cooluser@LAPTOP-5V55HON5:~$ ls -l /etc/shadow
-rw-r-- 1 root shadow 1824 Oct 18 19:49 /etc/shadow
cooluser@LAPTOP-5V55HON5:~$
```

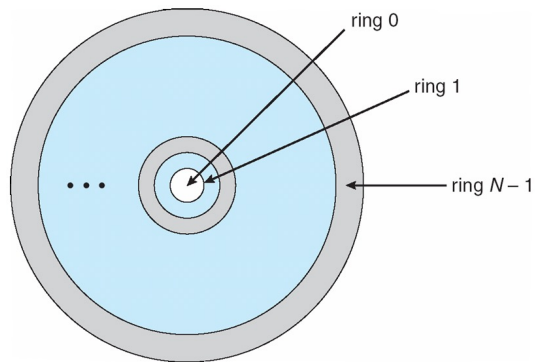
```
cooluser@LAPTOP-5V55HON5:~$ passwd
Changing password for cooluser.
Current password:
New password:
Retype new password:
passwd: password updated successfully
cooluser@LAPTOP-5V55HON5:~$
```

```
cooluser@LAPTOP-5V55HON5:~$ ls -l /bin/ls
-rwxr-xr-x 1 root root 142144 Sep  5 2019 /bin/ls
cooluser@LAPTOP-5V55HON5:~$ ls -l /bin/passwd
-rwsr-xr-x 1 root root 68208 May 28 01:37 /bin/passwd
cooluser@LAPTOP-5V55HON5:~$
```

10

Domain Implementation (MULTICS)

- Let D_i and D_j be any two domain rings
- If $j < i \Rightarrow D_i \subseteq D_j$



COP 4610 – Operating System Principles

11

11

Multics Benefits and Limits

- Ring / hierarchical structure provided more than the basic kernel / user or root / normal user design
- Fairly complex -> more overhead
- But does not allow strict need-to-know
 - Object accessible in D_j but not in D_i , then j must be $< i$
 - But then every segment accessible in D_i also accessible in D_j

COP 4610 – Operating System Principles

12

12

Access Matrix

- 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_{*j*}

COP 4610 – Operating System Principles

13

13

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	

COP 4610 – Operating System Principles

14

14

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
- User who **creates object** can define access column for that object
- Can be expanded to dynamic protection
 - Operations to **add, delete** access rights
 - Special access rights:
 - *owner of O_i*
 - *copy op from D_i to D_j (denoted by “*”)*
 - *control – D_i can modify D_j access rights*
 - *transfer – switch from domain D_i to D_j*
 - *Copy and Owner* applicable to an object
 - *Control* applicable to domain object

COP 4610 – Operating System Principles

15

15

Access Matrix Example

object \ domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

(a)

object \ domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	

(b)

COP 4610 – Operating System Principles

16

16

Access Matrix Example

object domain \	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
D_3		read	execute					
D_4	read write		read write		switch			

COP 4610 – Operating System Principles

17

17

Access Matrix Example

object domain \	F_1	F_2	F_3
D_1	owner execute		write
D_2		read* owner	read* owner write
D_3	execute		

(a)

object domain \	F_1	F_2	F_3
D_1	owner execute		write
D_2		owner read* write*	read* owner write
D_3		write	write

(b)

COP 4610 – Operating System Principles

18

18

Access Matrix Example

object domain	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			

COP 4610 – Operating System Principles

19

19

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

COP 4610 – Operating System Principles

20

20

Implementation of Access Matrix

- Generally, a **sparse matrix**
- **Option 1** – Global table
 - Store ordered triples $\langle domain, object, rights-set \rangle$ in table
 - A requested operation M on object O_j within domain D_i \rightarrow search table for $\langle D_i, O_j, R_k \rangle$
 - with $M \in R_k$
 - But table could be very large
 - Difficult to group objects (e.g., an object that all domains can read)
- **Option 2** – Access lists for objects
 - Each column implemented as an access list for a specific object (and **stored** with said object)
 - Resulting **per-object list** consists of ordered pairs $\langle domain, rights-set \rangle$ defining all domains with non-empty set of access rights for the object
 - Easily extended to contain default set \rightarrow If $M \in$ default set, also allow access

COP 4610 – Operating System Principles

21

21

Implementation of Access Matrix

- **Option 3** - Capability list for domains
 - Instead of object-based, list is domain-based
 - **Capability list** for domain is list of objects together with operations allowed on them
 - Object represented by its name or address, called a **capability**
 - Execute operation M on object O_j , process requests operation and specifies capability as parameter
 - Possession of capability means access is allowed

COP 4610 – Operating System Principles

22

22

Comparison of Implementations

- Many trade-offs to consider
 - Global table is simple, but can be large
 - Access lists correspond to needs of users
 - Determining set of access rights for domain non-localized difficult
 - Every access to an object must be checked
 - Many objects and access rights -> slow
 - Capability lists useful for localizing information for a given process
 - But revocation capabilities can be inefficient
- Most systems use **combination of access lists and capabilities**
 - First access to an object -> access list searched
 - If allowed, capability created and attached to process
 - Additional accesses need not be checked
 - After last access, capability destroyed

23

Revocation of Access Rights

- Various options to remove the access right of a domain to an object
 - Immediate vs. delayed
 - Selective vs. general
 - Partial vs. total
 - Temporary vs. permanent

24