Authentication and Authorization

Moshe Kravchik

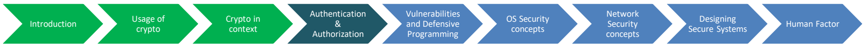
Credits: Mark Stamp

https://flic.kr/p/8xzAnc

Agenda

- Authentication vs Authorization
- Access Control
- Mandatory and discretional AC
- Multilevel security

https://flic.kr/p/6keSjR



Introduction

Usage of crypto

Crypto in context

Authentication & Authorization

Vulnerabilities and Defensive Programming

OS Security concepts

Network Security concepts

Designing Secure Systems

Human Factor

Authentication

- Who are you? (Prove it!)
- Hard for others to fake
- Basis for deciding what you're allowed

3

Authorization

- What are you allowed to do?
- Allow access to Alice
- Prohibit access to everyone else
- Allow access to Alice only to what she needs
- AKA Access control

4

Security policy

- Rules and guidelines for information security in organization
  - Least Privilege principle
  - Separation of Duties
  - Auditing
  - Accountability

https://flic.kr/p/6zzNyg

5

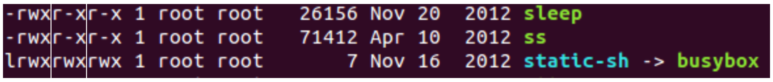
Access Control basics

- Is access control good or bad?
  - Stallman breaking into professor's office
- **Subject** – a user, a process
- **Object** – a file, database record, a process
- Types of
  - Read, write, execute
  - All are independent

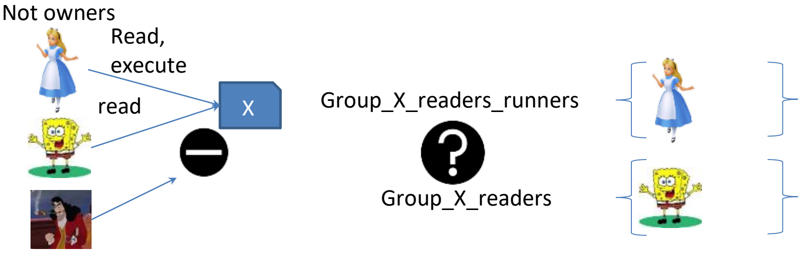
6

Access control example

- Unix :
  - Read, write, execute
  - All, Group, Owner

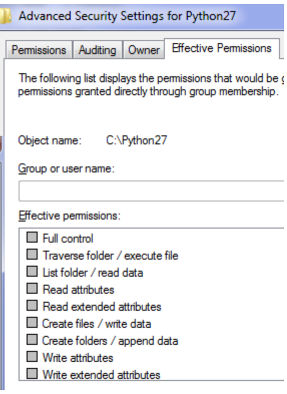


```
-rwxr-xr-x 1 root root   26156 Nov 20  2012 sleep
-rwxr-xr-x 1 root root   71412 Apr 10  2012 ss
lrwxrwxrwx 1 root root       7 Nov 16  2012 static-sh -> busybox
```



Access rights on Windows

- Windows :
  - Standard and Specific
  - Multiple users and groups



Advanced Security Settings for Python27

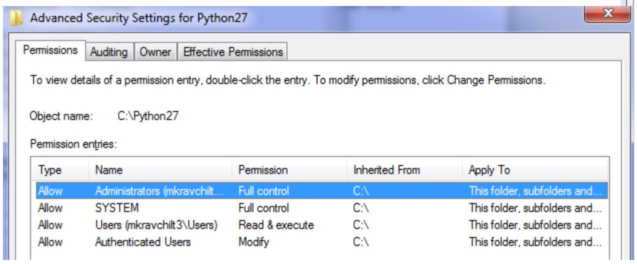
The following list displays the permissions that would be g
permissions granted directly through group membership.

Object name: C:\Python27

Group or user name:

Effective permissions:

- □ Full control
- □ Traverse folder / execute file
- □ List folder / read data
- □ Read attributes
- □ Read extended attributes
- □ Create files / write data
- □ Create folders / append data
- □ Write attributes
- □ Write extended attributes



Advanced Security Settings for Python27

To view details of a permission entry, double-click the entry. To modify permissions, click Change Permissions.

Object name: C:\Python27

Permission entries:

Type	Name	Permission	Inherited From	Apply To
Allow	Administrators (mkravchilt...	Full control	C:\	This folder, subfolders and...
Allow	SYSTEM	Full control	C:\	This folder, subfolders and...
Allow	Users (mkravchilt3\Users)	Read & execute	C:\	This folder, subfolders and...
Allow	Authenticated Users	Modify	C:\	This folder, subfolders and...

Access matrix (Lampson)

- **Subjects** (users) index the rows
- **Objects** (resources) index the columns

	OS	Accounting program	Accounting data	Insurance data	Payroll data
Bob	rx	rx	r	---	---
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	rw

9

ACL – Access Control List

- ACL: store access control matrix by **column**
- Example: ACL for **insurance data** is in **blue**
- Can contains defaults for other users, groups

	OS	Accounting program	Accounting data	**Insurance data**	Payroll data
Bob	rx	rx	r	---	---
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	rw

10

Capabilities

- Store access control matrix by row
- Example: Capability for Alice is in red
- Can be passed in runtime between subjects

	OS	Accounting program	Accounting data	Insurance data	Payroll data
Bob	rx	rx	r	---	---
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	rw

ACLs vs Capabilities

- ACLs
  – Stored with the object, simple
  – Good when users manage their own files
  – Easy to determine & change rights to a resource
- Capabilities
  – Easy to delegate
  – Easy to add/delete users (opposite of ACL)
  – More difficult to implement (security of tickets)

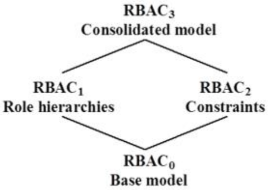
Role Based Access Control

- Access is given to Roles, not users
- A user can have multiple roles
  - A subset is active during a **session**
- Simplifies access management
  - Easy to change user’s role
  - Easy to add new rights to the role
- Follows known security principles
  - Least privilege
  - Separation of duties

https://flic.kr/p/6jwq7

RBAC models

- $RBAC_0$ – user can act in different Roles using Sessions
- $RBAC_1$ – hierarchy or Roles.
  - An accounting manager has all permissions of an accountant
- $RBAC_2$ Constraints on Roles
  - A person who gives the loan can’t approve it
- $RBAC_3$ – Constraints on hierarchies



$RBAC_3$
Consolidated model

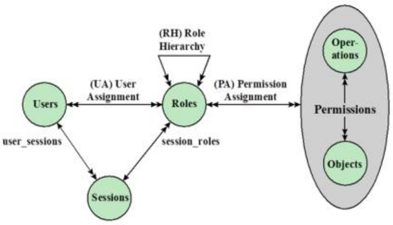
$RBAC_1$
Role hierarchies

$RBAC_2$
Constraints

$RBAC_0$
Base model

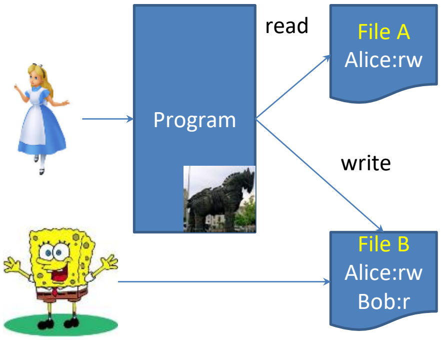
RBAC advantages

- Reflects organizational structure
  - Assigns permissions to Roles, not users
- Allows defining domain specific permissions
  - Debit/Credit for banking system
- User can use only some of permissions
- Hierarchy of roles – efficient administration



Information flow problem

- Bob can't read A
  - But can read B
- Alice can read A and write to B
- Alice or Trojan infecting Alice's program leaks A to Bob!
- Hard to solve with ACLs and capabilities



Bob can read File A now!

Security Models

- Theoretical models formalizing access control
- Many of them rooted in military systems
- US DoD levels of classifications:
  - Top Secret, Secret, Confidential, Unclassified
- Practical classification problems
  - Proper classification not always clear
  - Level of granularity to apply classifications

https://flic.kr/p/8xzAnc

17

Mandatory and discretionary policy

- Multilevel security models (MLS) are an example of **<u>mandatory</u>** protection (MAC)
- Mandatory protection – enforced in a way that users can't change or violate it
- In order to get access to Top Secret document the user must have Top Secret clearance (and gets access to all Top Secret docs)

https://flic.kr/p/e7ULKC

18

Mandatory and Discretionary Access Control

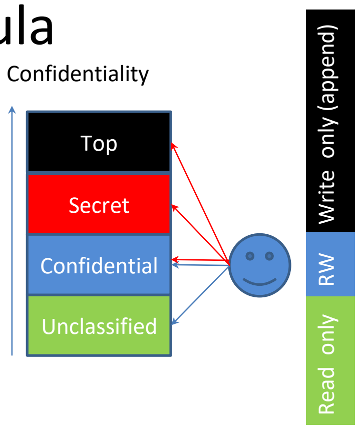
- **Discretionary** – the users decide which protection to apply to objects (DAC)
- Modern OSes are discretionary, with some examples of mandatory policy
- In discretionary systems access is defined based on **Need-To-Know** principle
- SELinux and Integrity in Windows

https://flic.kr/p/nrwgpV

19

Bell LaPadula

- The most famous security model
- Defines minimal requirements that MLS must satisfy
- Every object and subject has security level
  - Unclassified, Secret, Top Secret
- Simple Security – No read up
- *-Property – No Write Down
- ds-property – DAC constrained by MAC



20

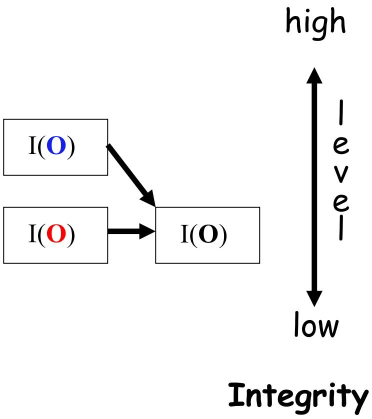
Bell LaPadula

- Very simple, solves information flow
- Not practical, administrator will try to declassify documents
- Provable, but not realistic
- Inspired other models
- High Water Mark property – the actual process level is equal to the highest document it read

https://flic.kr/p/4fWUk

Biba’s Model

- BLP for confidentiality, Biba for **integrity**
  - Biba is to prevent unauthorized writing
  - Not reading low integrity data
- Integrity model
  - Suppose you trust the integrity of **O** but not **O**
  - If object **O** includes **O** and **O** then you cannot trust the integrity of **O**
  - Integrity level of O is minimum of the integrity of any object in O



Biba's Model

- I(O) - the integrity of object O, I(S) - the integrity of subject S

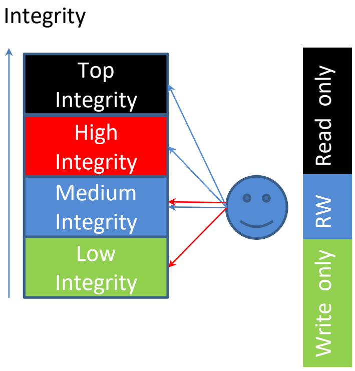
  Simple Integrity: S can write O if and only if

  I(O) ≤ I(S)

  Integrity confinement: S can read O if and only if

  I(S) ≤ I(O)

  Low Water Mark Policy: If S reads O, then I(S) = min(I(S), I(O))
- BLP ignores integrity, Biba - confidentiality



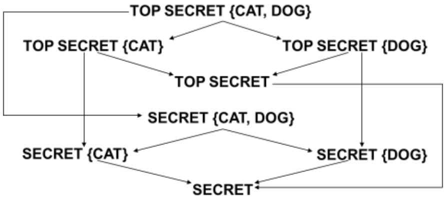
Compartments

- Multilevel Security (MLS) enforces access control **up and down**
- Simple hierarchy of security labels is generally *not* flexible enough
- Compartments enforces restrictions **across**
- Suppose **TOP SECRET** divided into **TOP SECRET {Administration}** and **TOP SECRET {Intelligence}**
- Both are **TOP SECRET** but information flow restricted across the **TOP SECRET** level

https://flic.kr/p/8E7pP

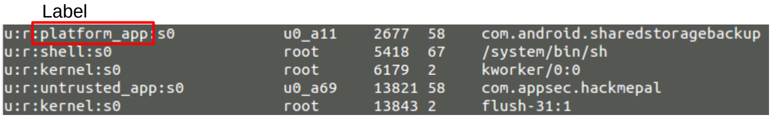
Compartments

- Why compartments?
  - Why not create a new classification level?
- May not want either of
  - TOP SECRET {Admin} ≥ TOP SECRET {Intelligence}
  - TOP SECRET {Intelligence} ≥ TOP SECRET {Intelligence}
- Compartments designed to enforce the **need to know** principle
  - Regardless of clearance, you only have access to info that you need to know to do your job



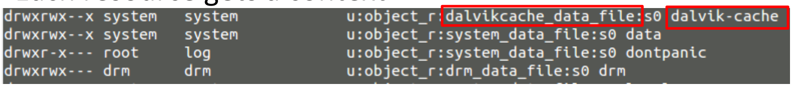
SEAndroid/SELinux

- Based on SELinux by NSA, enforced in 5.0
- Mandatory Access Control policy enforced by kernel
- Each processes gets a label



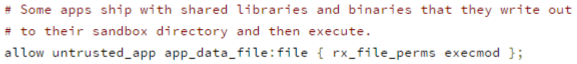
```
u:r:platform_app:s0          u0_a11    2677  58    com.android.sharedstoragebackup
u:r:shell:s0                 root      5418  67    /system/bin/sh
u:r:kernel:s0                root      6179  2     kworker/0:0
u:r:untrusted_app:s0         u0_a69    13821 58    com.appsec.hackmepal
u:r:kernel:s0                root      13843 2     flush-31:1
```

- Each resource gets a context



```
drwxrwx--x system   system              u:object_r:dalvikcache_data_file:s0 dalvik-cache
drwxrwx--x system   system              u:object_r:system_data_file:s0 data
drwxr-x--- root     log                 u:object_r:system_data_file:s0 dontpanic
drwxrwx--- drm      drm                 u:object_r:drm_data_file:s0 drm
```

- Policy matches between them



```
# Some apps ship with shared libraries and binaries that they write out
# to their sandbox directory and then execute.
allow untrusted_app app_data_file:file { rx_file_perms execmod };
```

Questions?

https://flic.kr/p/pqiJNt

27

Terms learnt

- Authentication vs Authorization
- Least Privilege, Separation of Duties
- Subject & Object
- ACL & Capabilities
- DAC, MAC and RBAC
- Multi Level Security
- Bell LaPadula
- Biba
- Compartments
- Need to know

28

28