New submission boxes for ex 3 and 4, but also split box 1 and 2

From now on we have our own submission boxes for ex.

Resubmit ex 1 and 2

Ex 4 we got…

Ex3:

2 main problems, import pycryptodome

Depends on python version, get thing that matches version

Decrypt fn is similar to encrypt function – symmetric encryption

16 bit – hint on type of car

Ex4:

No code for the 2nd question, explain how protocol looks like

- Describe without code

1st question we need code…

DB or txt file?

Last time reminder:

- One time pad encryption
    - Best way to encrypt things
    - Key is as long as the data
    - Pad is as much of secret as code itself
    - If re-used, we can do an XOR attack
    - Hard to generate random keys
- Entropy = how random it is
    - Randomness
    - Patterns in 128 bit key which makes it easier to crack
    - 40 bits random = 40 entropy
- MITM
    - Adapt protocols
    - Attacking protocols
- Randomness
    - True randomness via lava lamp or external tools

Authentication vs authorization

- Multi level security
- Access control
- Authentication: Bank identify a person to make an account
    - hard to break or fake
    - decide who is responsible for that account
        - Alice?

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

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

Authorization: what can Alice do with her account

- Shared accounts - separation
- Deposit / withdraw
- Can she borrow money from bank?
    - Loans
    - Bank approves it

Info security rules and guidelines via principles

- Least privilege principle
    - Just what they need
    - User vs Admin
- Separation of duties
    - Get access to resource via other people
    - Person who gives the loans shouldn't be the one to approve them
        - Israeli scammer recently stole hundreds of millions of shekels

Audit

- Detection method

Accountability

- Non repudiation
- Who is responsible for operations

Stallman

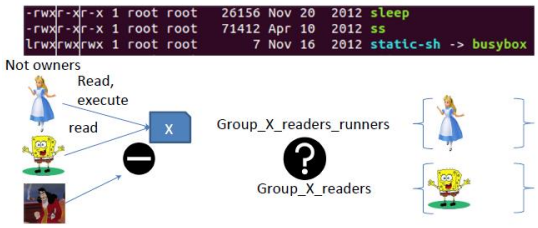
- Free computing for all
- Broke into staff's office to access computers
    - Handle problem and protect systems
        - Split things up
            - Subject vs objects
            - Users or processors
            - Objects: files DBs, etc
            - Read, write, access are independent of each other

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

Access control example

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



unix versions, divided system to types of groups

Owner, group, and all (everything else)

rwx with

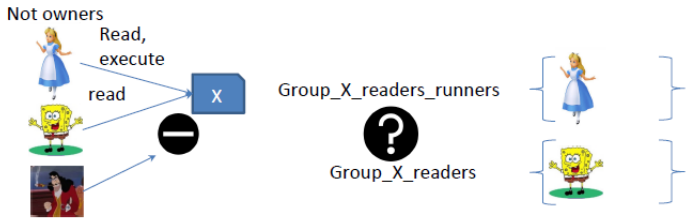
replace – with the one we don't want to access

L = link file – symbolic link

rwx – owner ->

group can read and execute

all can just link



Give users permission to read and execute but ban others rom the same thing

Alice = in a group

Eve = all

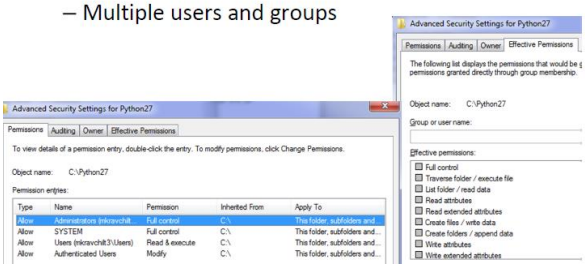
Now we want to give bob an option to read but not execute, so bob needs a different group

- But we have just 3 groups in unix

Simple to implement and important

Access rights on Windows

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



Windows file system

- More options for groups and define management

Linux/unix took it too

Windows needs computing resources to handle the file system

- OS is heavier

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	$\boxed{\text{rw}}$	rw	rw

New way to handle users, resources

Put everything in a matrix

Users in rows

And columns we have data types

Accounting program is only one that can write to account data to protect properties

More users = bigger matrix to handle

Most of table will have blank and empty spots – we don't need that

- Search table to find access

Handle via sibjects vs users separate

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

Column = by object

Relate to each file, users and access we defined

Don't have permission for a file -> no empty spot for you there

ACL

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

Other method is via subject and each user gets and keeps the file he can access and what level

Alice read execute OS files

If user doesn't have access to file, no place in table for him

Relational DB

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)

Analysis of tables and comparison of the permissions

ACL = files and processes

- Good for users managing their own files
- If we fire someone we need to remove them from 5000 files

Capabilities

- Delegate easy
- Switch users is easy too
- More difficult and security risks with users
    - If we make mistakes, some user gets permission
    - Or inherits privileges from other users

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

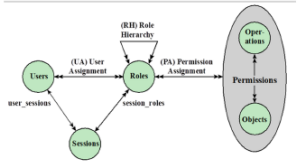
https://flic.kr/p/6jwq7

Is alice a student? – gets access to those student related info and nothing else

Bob = professor – gets more privileges

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



Role based access control = RBAC

Role = job title

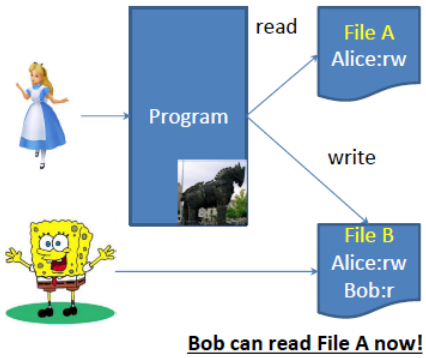
Hierarchy of roles

But has its issues

- Alice can leak info from a file she has access to that bob doesn't have access to and send over information
- Issue with military classifications – shared folders with organization – take classified info and put them in public spaces

Information flow problem

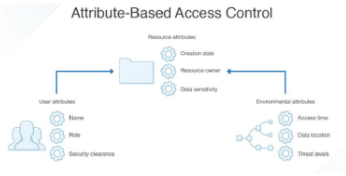
- Bob can't read A
  - But can read B
- Alice can read A and write to B
- Alice or Trojan infecting Alice's program leaks A to Bob!
- Hard to solve with ACLs and capabilities



How to protect things? Use security models

Attribute Based Access Control (ABAC)

- More fine-grained control based on attributes:
  - User
  - Environment
  - Resource
- RBAC is simpler
- ABAC is more fine-grained and powerful



https://www.dnsstuff.com/rbac-vs-abac-access-control

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

Tools we can use to help us – split classifications up

Mandatory and discretionary policy

- Multilevel security models (MLS) are an example of **mandatory** 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

Mandatory access control

Mac address

message authentication code (MAC)

get security clearance at each level

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

Only get access on a need to know basis to perform your job, nothing else

Nurse can only know specific info about patient, give what doctor ordered nothing else

Modern OS have DAC and Mandatory policies too (not user dependent), built into the OS and security policy

Discresionary – depends on user but part of system not depends on user

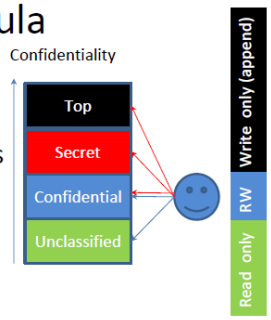
DAC makes life easier

Not accessed to all files – user can destroy the system (hackers)

Selinux – uses DAC policy with MAC too

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



Structure – theoretical

Hard to do in practice

What are the main differences between capability lists and access lists?
Answer:
An access list is a list for each object consisting of the domains with a nonempty set of access rights for that object. A capability list is a list of objects and the operations allowed on those objects for each domain.

Structure -> no read up from lowest to highest security level

No write down either

Top level can read what low levels wrote

No read up

Problem is chief of staff (head of security) cannot write from top to unclassified (top secret to unclassified)

Cannot communicate

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

High water mark is the solution

Chief of staff logs into system -> starts r/w to everyone

At moment he opens document from higher unclassified and in that level of clearance levels -> he stays

➔ And remains there

Solves our communication level issues

Log out and log back in to get to lowest level.

Always will have leaks in the system and find weak links

Solution to Confidentiality (CIA) but

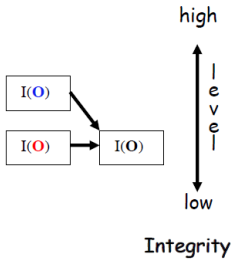
Biba's Model

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



guarantee integrity

Uses subject and objects

Trusted file – high integrity mixed with information from low integrity file

New file will be low integrity (keeps the level)

Lowest common denominator

Low integrity – pollutes the file in a way and rank will drop down

Biba's Model

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

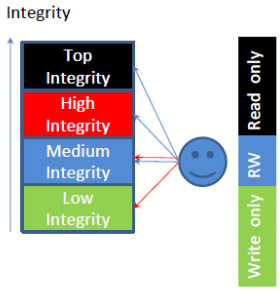
  Simple Integrity: S can write O if and only if

  $I(O) \le I(S)$

  Integrity confinement: S can read O if and only if

  $I(S) \le I(O)$

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



different model than biba confidentiality

Top level cannot read from low levels (untrusted)

Chief of staff making choices based on social media (misinformation)

No read down

And also no write up (don't want to pollute top integrity documents)

Top reads down gets the lowest integrity option

Top level starts from the beginning at top and stays there

with integrity in biba's (ignores confidentiality)

Bela pedula ignores integrity

BLP doesn't allow copying on 1 doc and pasting on lower level of confidentiality

- Impossible
- But later in the course, there is way to leak similar info
- Still sophisticated ways to leak info

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

Next thing

Related to org, diff departments

From admin staff to look at top secret docs, they don't need to look at other depts. Top secret documents

Know only what you need to know

- Also separation of roles

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 {Admin}**

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

Medical user info is top secret

- Personal
  - HIV patient – doesn't want rest of world to know about disease
- Give medicine for patient with specific disease as pharmacist
  - Don't know illness but need to know how much medicine?
    - Pharmacist will know what patient is sick with – know the medicine relates to illness
- So gave pharmacists top secret clearance
  - Now everybody knows about everything…
  - So solution is work with compartments
    - Specific meds -> get access to related files in same level

SEAndroid/SELinux

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

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

File Context File Name

```
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 };
```

Same OS

MAC – mandatory access control implemented

Processes and IDS and users and got platform info and where it stays

Untrusted apps – from play store – gets that classification

- Limited

Platform app

- Google app or something trusted

Split applications we download to different groups

And for each app, which of the 5 can R/W/X

Malicious apps cannot get root access (like in unix), superuser with all privileges

But now is highly restricted based on policy

Cannot harm kernel or OS files

NSA developed this system and probably has a backdoor

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