Lecture 8 - 10/01/2019

Access Control

The protection of your resources, setting who can and can't read and write. This will enforce your security policy.

- Can be applied at different levels
    - H/W Level
        - What programs can access what resources
    - OS Level
        - What users can access what files/directories
    - Middlewear
        - In Between the OS and the hardware level
        - An example would be a database.
    - Application layer
        - Applications will control access to their own data.

Origins

Originated in 1971 from a person named Butler Lampson. He was thinking about access control at the OS level and decided to create a structure:

Users	OS	Accounting Program	Accounting Data	Audit Trail
Sam - SYSAdmin	RWX	RWX	RW	R
Alice - Manager	X	X	R	-
Bob - External Auditor	RX	R	R	R

This is called an access control matrix. In a simple context this is effective, however in a larger context it gets much more inefficient.

3 Approaches to improve efficiency

1. Use groups or roles (e.g. RBAC)
    - Reduce amount of info
2. Store Columns only (ACL)
    - Stores with resources
3. Store Rows Only (Capability List)
    - Stores with user

} Spread info around

ACL - Access Control List

Pros:

- Simple; easy to use, create, and understand.

Cons:

- Not suited for large dynamic populations.
    - If users are constantly changing updating this list will be too frequent.
    - If users delegate their access to another user, this can't be represented.
- Not efficient
- Not suited for revocation

Capability List

- harder to create
- Delegation is easier
- More suited to revocation
    - There is one data structure for a user's permissions
- Not suited to changing the status of an object
    - You're required to go through each user's capability list

Implementation Considerations

ACL

1. Which subjects can modify an ACL?
    - Who gets to create it and who can modify it?
    - Typically it's the creator of the object (ORCON)

2. Privileged users. Do ACLs apply?
    - Are the above the ACLs or is there a row for them?

3. Does ACL support groups or wildcards?
    - E.g. when UID:GID:permission
        i. Alice:*:'r'
        ii. *:dev:'r'

4. Contradictions in ACL
    - If we allow things like groups it will shrink the ACL. However it can lead to contradictions.

5. Defaults & ACls
    - You might have default settings in the system and you may also have some ACls.
    - E.g.
        i. Default policy is that access is granted.
        ii. We also have an ACL that lists who may access what.
        iii. if alice's name is not on that list she will be granted access by the default policy.

6. Revocation
    - A gives permission x to B
    - B gives permission x to C
    - If A revokes permission x to B, should C have permission x revoked?
    - What if C also got permission x from D?
    - We have to keep track of all delegations and revocations

Capability Lists

- Stored with user and not the OS
- You're storing the permissions with the person that is most likely to be an attacker
- They need to be protected from the actual user

Ways to protect CL from users:

1. Specified hardware
    a. Called Tagged architecture
    b. Set of bits association with each h/w word
2. Store in page that processes read but not modify
3. Use cryptography

Revocation:

- Revoke access to a specific object
    - Could check all Cls and delete relevant entries. However this is hard if not impossible
    - Could use indirection
        - Within a CL you don't point to an object, you can point to a name or an identifier.
        - Repoint this name to a new object

General Questions

1. Given a subject, what objects can it access?
    a. Check one CL - Easy
    b. Check every ACl is system - Hard/Time Consuming
2. Given an object, what subjects have access to it?
    a. Check one ACL - Easy
    b. Check every CL - Impossible?

ACL-based System: Easy or Hard
CL-based System: Easy or Impossible.

Because of this: ACLs are typically used, not CLs

Alternative to ACLs and CLs

Not a row of the matrix, not a column of the matrix. A single cell of the matrix.

Attribute certificate

- Here is a permission Alice has, this is then signed by an authority.
- Might hold a single permission for Alice.
- Easy to delegate without revealing much information.

Revoking permissions:

- In the certificate is { Name/id, Permission }
- Instead of putting name, just put a pointer to her public certificate
- Whoever can authenticate using the public key is the owner of the permission
- Dynamic population is controlled through AC and PKI creation

Inefficient.

Other AC technologies

Lock and key

- Here is an object, here are the permissions required
- Take an ACL and an AC, find the intersection
- It can also be dynamic, can add time of day or location requested from

Ring Based AC

Propagated ACL

Sandbagging

Proof-carrying code

Object Request Broker

- Middleware that all requests go through (CORBA)