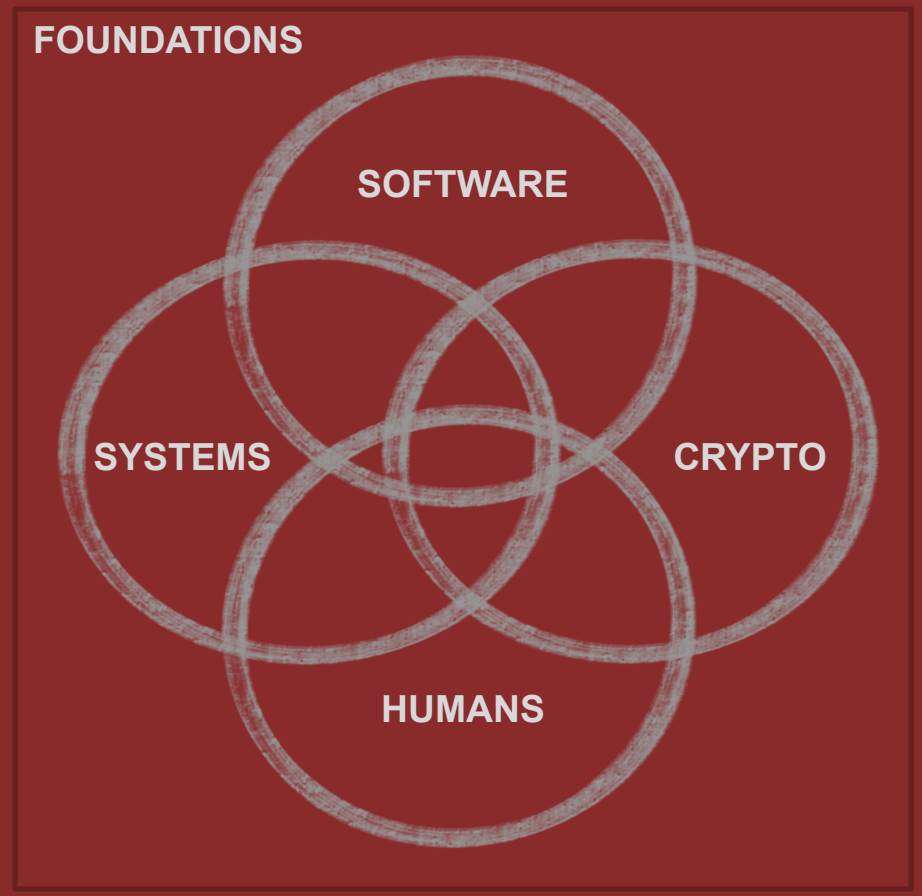
Διάλεξη #10 - Access Control

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Εισαγωγή στην Ασφάλεια

Θανάσης Αυγερινός



*some slides by John Mitchell

Huge thank you to David Brumley from Carnegie Mellon University for the guidance and content input while developing this class

Ανακοινώσεις / Διευκρινίσεις

- Ουδέν νεότερο!

Την Προηγούμενη Φορά

- Security Fundamentals
  - Adversaries
  - Threat Models
  - Security Properties
  - Trusted Computing Base (TCB)
  - Security Principles

Do we remember anything?

Security in the News

Breach of the Week

(CNN) — About 576,000 Roku accounts were compromised in a cyberattack, the company said on Friday, the second security breach for the streaming service this year.

Hackers gained access to user accounts through stolen login credentials, Roku said in a blog post. The security breach was discovered while Roku monitored account activity after a cyberattack affected 15,000 accounts earlier this year.

In each instance, fraudsters used a cyberattack method known as credential stuffing: Hackers try login and password information leaked in one data breach on a variety of users' accounts, exploiting people who use the same credentials across different accounts. (Experts recommend people use different passwords for each of their online accounts.)

Roku, a streaming giant, hosts more than 80 million users. The company announced it is implementing two-factor authentication across all Roku accounts. The two-step security confirmation prompts users on a second device whenever there's an attempted log in.

https://edition.cnn.com/2024/04/12/business/roku-security-breach-user-accounts/index.html

Breach of the Week #2

Sisense's data breach is serious enough that CISA is investigating. Here's what you need to do

Earlier this week, Sisense confirmed that the data dashboard and analytics company had been hit with a breach that allowed hackers to access its customers' data. While Sisense has been somewhat tight-lipped about the exact nature and extent of the attack, security site KrebsonSecurity cited sources who said that the attack appears to have allowed hackers to steal terabytes worth of customer data, "which apparently included millions of access tokens, email account passwords, and even SSL certificates."

This is likely why the US government is taking the attack so seriously. The US Cybersecurity and Infrastructure Security Agency (CISA), which acts to protect critical US infrastructure against cybersecurity threats, issued an advisory on the matter this week, and said that it's working with other companies in the industry to mitigate any potential problems the breach could spur.

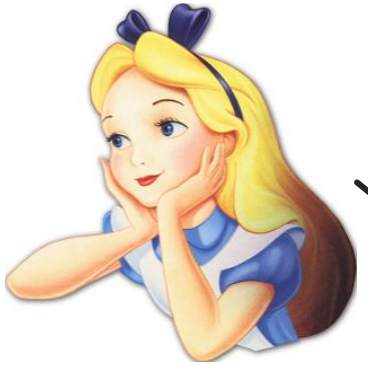
https://www.zdnet.com/article/sisenses-data-breach-is-serious-enough-that-cisa-is-investigating-heres-what-you-need-to-do/

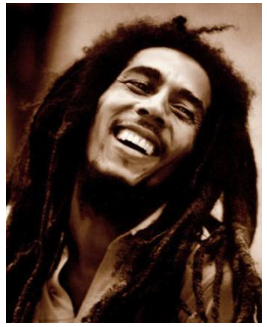
Σήμερα

- Reference Monitors
- "Gold" (Au) Standard
  - Authentication + Authorization + Audit
- Authorization Mechanisms / Access Control
  - Access Control Lists (ACLs) and Capabilities (CAP)
  - Discretionary Access Control (DAC)
  - Role-Based Access Control (RBAC)

Access to Resources and Control

General Purpose Computers





General Purpose Computer

Resource1 (Alice)

Resource2 (Bob)

Resource3 (Shared)

Obvious Questions

How do I know that it is in fact Alice?

- Authentication

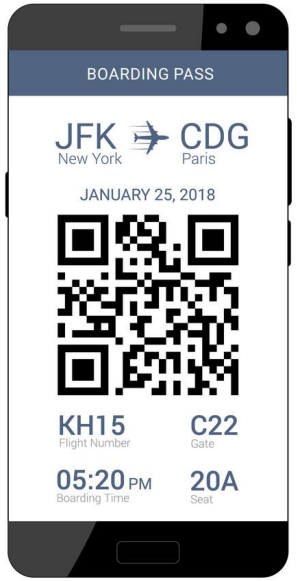
Can Alice access Bob's file? Can she access the shared file?

- Authorization

Did Alice try to delete Bob's file?

- Audit

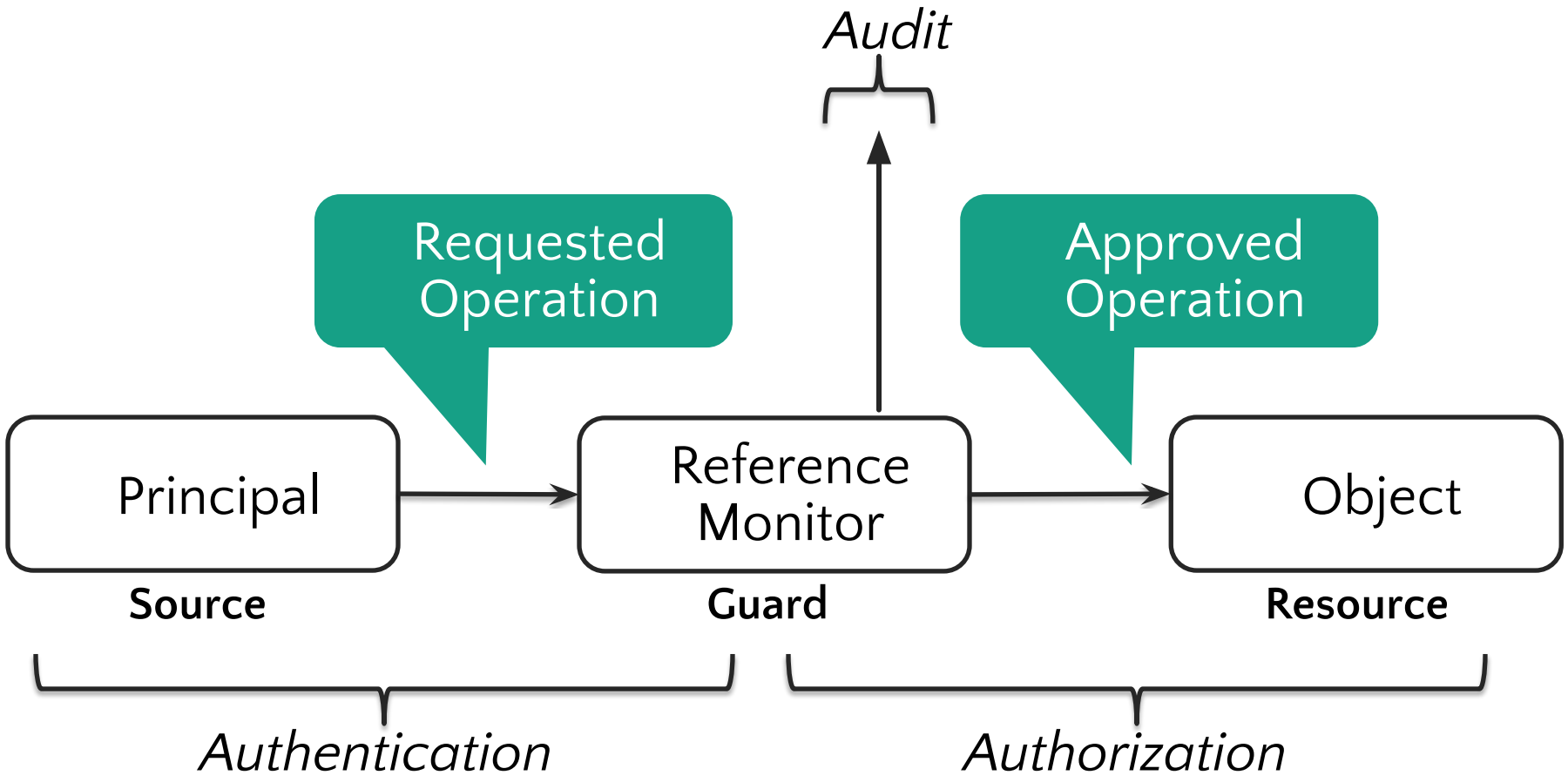
Authentication vs Authorization



Authentication: Who a user is
Example: Drivers license authenticates your identity

Authorization: What users can access
Example: Boarding pass authorizes you to board a flight

Abstract Access Control Model



Principals for Authentication

- "Who did that" or "Who is getting access"
- Commonly: user
- Also:
  - Program / phone app
  - Network port
  - Process
  - Machine

Mechanisms for Authentication

and sometimes authorization

- For humans:
  - Something you know, something you have, something you are
    - E.g., password, YubiKey, fingerprint
  - Two-factor or multi-factor mechanisms
    - Ideally, using factors from different categories
- For code / data / processes:
  - Trusted metadata
    - E.g., process ID
  - Cryptographic authentication
    - E.g., digital signature, MAC

Authorization

Who is trusted to perform “what” operations on this object

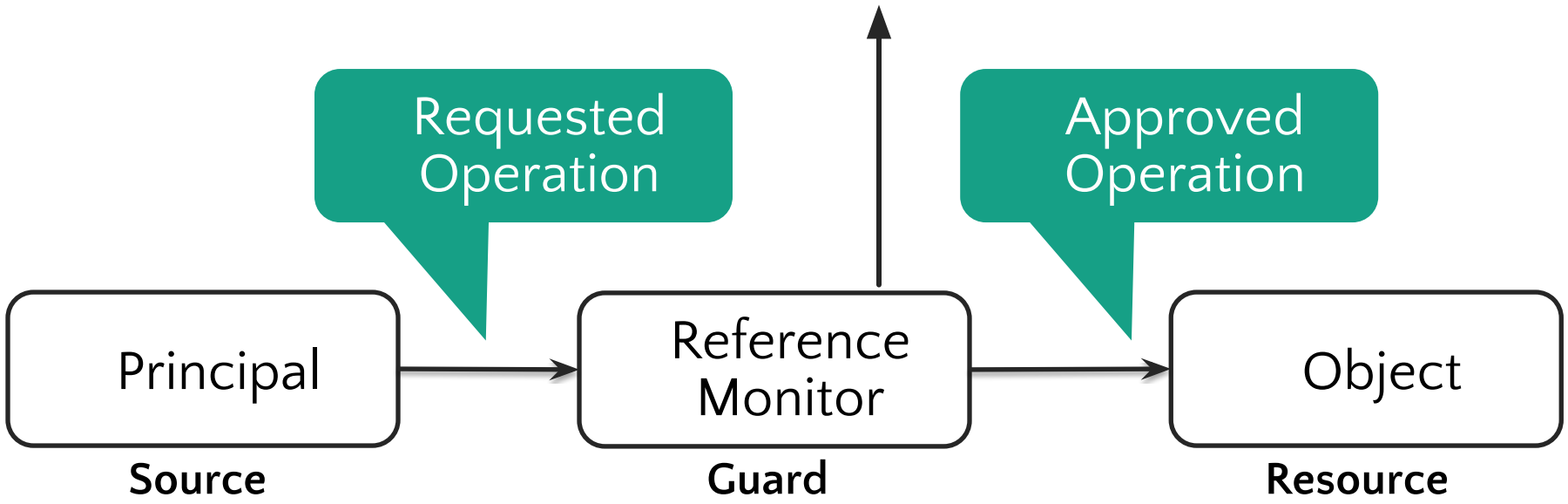
Important question: How do we specify the policy?

(spoiler: Access Control!)

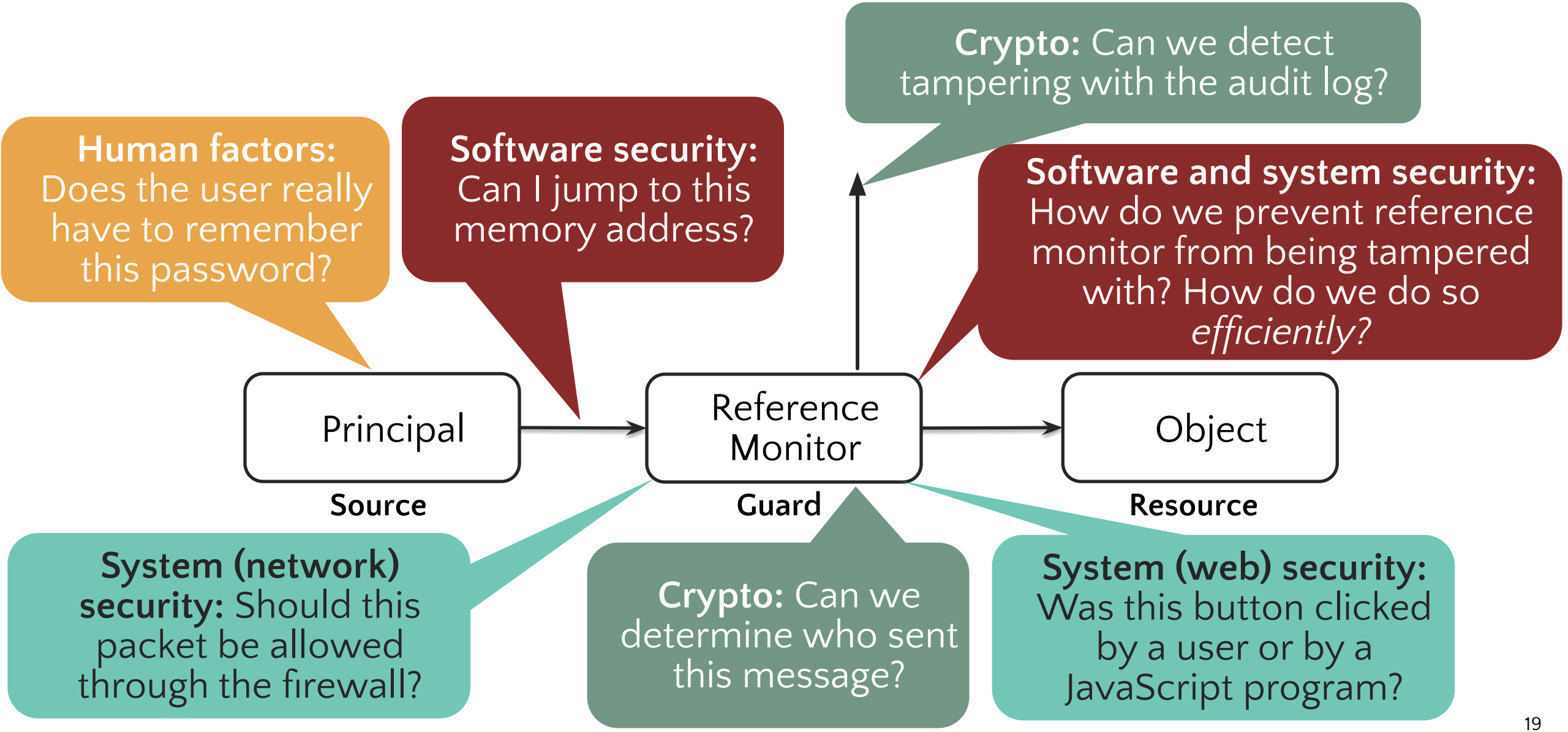
Audit

- Evidence of and for decisions being made
- Why do we need audit?
  - Useful for forensics
  - Useful as a diagnostic tool
  - Audit trail can help track attacks
- Extremely important, but often forgotten

Abstract Access Control Model



Abstract Access Control Model



Participation Question

Which of the following is an example of an authorization check?

A. Consulting the system log to see which admin introduced a backdoor

B. my-studies checks to see if you are registered as a professor for the course before allowing you to change the grade of a student

C. Showing your driver's license at the airport's security checkpoint

D. An ice skating rink wants you to sign a waiver before you start skating

Principles of Access Control

Protection State

- ***State of system:*** current values for all resources of the system
- ***Protection state:*** subset of state that deals with protection

- ***Security policy***: Characterizes states in Q
- ***Access control matrix***: One kind of precise representation of Q
- ***Security mechanism***: Prevents system from entering P–Q

Subjects, Objects, Rights

- Objects (o): Set of protected entities relevant to system
  - Files
  - Directories
  - Memory
  - Processes
- Subjects (s): set of active objects $S \subseteq O$
  - Running processes, users, …
- Rights (r):
  - Read
  - Write
  - Execute
  - Append
  - Own

Examples

UNIX

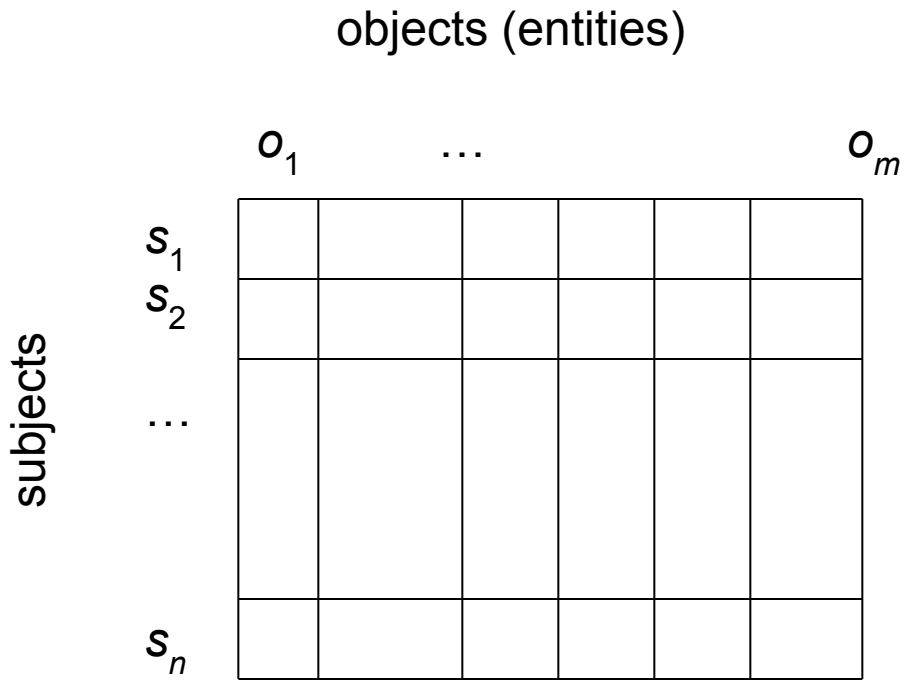
- Subjects: Running processes
- Objects: Files, directories, processes,…
- Rights:
  - read
  - write
  - execute

AFS

- Subjects: Kerberos principals
- Objects: Files, directories, processes, …
- Rights
  - Lookup – List contents of directory
  - Insert – Add new files to directory
  - Delete – Remove Files
  - Administer – Change access controls
  - Read
  - Write
  - Lock - Programs that need to flock

Lampson's Access Matrix

- Subjects are row headings, objects are column headings
- Access control entry [*s*,*o*] determines rights for subject *s* when accessing object *o*



- Subjects $S = \{ s_1,\ldots,s_n \}$
- Objects $O = \{ o_1,\ldots,o_m \}$
- Rights $R = \{ r_1,\ldots,r_k \}$
- Entries $A[s_i, o_j] \subseteq R$
- $A[s_i, o_j] = \{ r_x, \ldots, r_y \}$ means subject s_i has rights $r_x, \ldots, r_y$ over object o_j

* From: Computer Security, Art & Science

Example: Processes and files

- Processes p,q (subjects & objects)
- Files f,g (objects)
- Rights r,w,x,a,o

	f	g	p	q
p	rwo	r	rwxo	w
q	a	ro	r	rwxo

Playing with Unix

Types of Access Control

- **Discretionary Access Control (DAC):**
  User can set an access control mechanism to allow or deny access to an object.
  - It's at the user's discretion what to allow
  - Example: UNIX file ownership
- **Mandatory Access Control (MAC):**
  System mechanism controls access to an object and an individual user cannot alter that access.
  - What is allowed is mandated by the system (or system administrator).
  - Example: Even the author of a TOP SECRET file cannot make it PUBLIC

Types of Access Control

- **Role–Based Access Control (RBAC):**
  Access based on *role*, not *identity*
  - A role defines a set of permissions
  - Thanassis as Faculty @ DI vs Thanassis as member of UOA vs Thanassis as Hacker vs …

- *Orthogonal* to DAC vs MAC

- Roles and groups (e.g., UNIX groups) are related, but not identical
  - Role R $\supseteq$ Role S $\Longrightarrow$ Role S has *at most* the permissions of Role R
  - Group G $\supseteq$ Group H $\Longrightarrow$ Group H has *at least* the permissions of Group G

Access Control Mechanisms

- Ways of specifying and enforcing policy

- Access control matrices:
  - Can be precise
  - Can be huge
  - Two common implementation strategies:
    - Access Control Lists
    - Capabilities

Access Control Lists

	f	g	Ethan	Andre
Ethan	rwo	r	rwxo	w
Andre	a	ro	r	rwxo

An ACL corresponds to a column in an Access Control Matrix

- f: {(Ethan, rwo), (Andre, a)}
- g: {(Ethan, r), (Andre, ro)}

Formally: an ACL for an object o is a list of pairs: $\{(s_i, r_i)\}$

Default Permissions

- Normal: if not named in ACL, *no* rights over file
  - Recall from last time: Principle of Fail-Safe Defaults!
- Unusual but possible: If not explicitly denied, has rights
- If many subjects, may use groups or wildcards in ACL

Design Decisions for ACLs

1. Which subjects can modify an ACL?
2. Which, if any, ACLs apply to privileged users (e.g., root)?
3. Does the ACL support groups or wildcards?
4. How are contradictory access control permissions handled?
5. If a default setting is allowed, does it apply only when subject is not explicitly mentioned?

Aside: abbreviations & UNIX

- ACLs can be long … so combine users into classes
  - UNIX: 3 classes of users: Owner, Group, All
  - rwx rwx rwx
  - e.g., `chmod 644 /var/www/index.html`
  - Ownership (and default permissions) assigned based on creating process
- Limitations
  - Suppose Anne wants:
    - all rights for herself
    - Beth to have read access
    - Caroline to have write access
    - Della to have read and write
    - Elizabeth to execute
  - 5 desired arrangements, so three triples insufficient

Capabilities

	f	g	Ethan	Andre
Ethan	rwo	r	rwxo	w
Andre	a	ro	r	rwxo

A Capability corresponds to a row in an Access Control Matrix

- Ethan: {(f, rwo), (g, r), (Ethan, rwxo), (Andre, w)}
- Andre: {(f, a), (g, ro), (Ethan, r), (Andre, rwxo)}

Formally: a Capability for subject s is a list of pairs: $\{(o_i, r_i)\}$

Semantics of a capability

- Like a bus ticket
  - Possession indicates rights that subject has over object
  - Object identified by capability (as part of the token)
    - Name may be a reference, location, or something else
- Must prevent process from altering capabilities
  - Otherwise, subject could change rights encoded in capability or object to which they refer

Example

- UNIX `open()` call returns a file descriptor
- This file descriptor is a capability!
  - Even if file is deleted and a new file with the same name is created, the capability still works
  - Aside: this is a common wait of creating a temporary file that will be deleted as soon as it's closed. See `man unlink`

Capability Implementations

1. Tags:
   - Capabilities are stored in memory words with an associated tag bit that can only be modified in kernel mode
   - e.g., CHERI capability machine (coming soon to a RISC-V near you!)
2. Protected memory:
   - Capabilities stored in kernel memory and can only be accessed indirectly (i.e., via syscall)
3. Cryptography
   - Capabilities are cryptographically authenticated and cannot be modified by user process
   - Can be stored in user space

How can we revoke capabilities?

- Scan all outstanding capabilities, delete or invalidate relevant ones
  - Expensive!
- Use indirection
  - Each object has entry in a global object table
  - Names in capabilities name the entry, not the object
    - To revoke, zap the entry in the table
    - Can have multiple entries for a single object to allow control of different sets of rights and/or groups of users for each object

Comparing ACLs and Capabilities

- Both can be used to describe the same access control policy
- Consider these questions
  - Given a subject, what objects can it access, and how?
    - Capabilities more efficient
  - Given an object, what subjects can access it, and how?
    - ACLs more efficient
- Tracking which subjects can access a given object is more common, thus ACLs are more popular because they are more efficient for this case
- Other trade-offs may be worthwhile, e.g., revocation on per-subject basis is easier with capabilities

Participation Question

Which of the following is an example of a capability in the real world?

A. The bouncer at a party looks at your ID and checks if you name is on the invite list

B. You use your dorm room key to unlock and enter your room

C. You run a 5-minute mile to show that you're qualified to join the track team

Process effective user id (EUID)

- Each process has three Ids (+ more under Linux)
  - Real user ID (RUID)
    - same as the user ID of parent (unless changed)
    - used to determine which user started the process
  - Effective user ID (EUID)
    - from set user ID bit on the file being executed, or sys call
    - determines the permissions for process
      - file access and port binding
  - Saved user ID (SUID)
    - So previous EUID can be restored
- Real group ID, effective group ID, used similarly

Process Operations and IDs

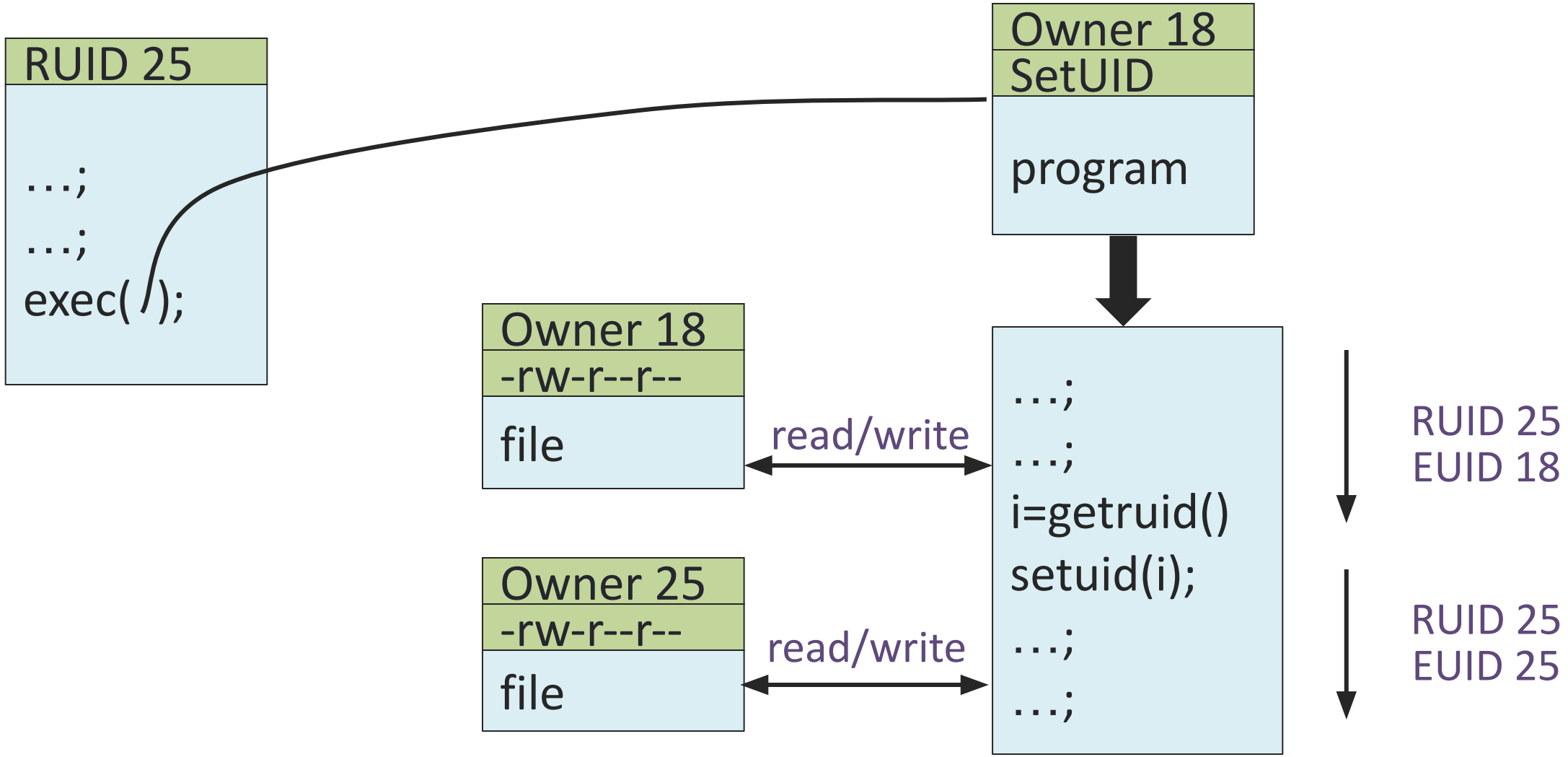
- Root
  - ID=0 for superuser root; can access any file
- Fork and Exec
  - Inherit three IDs, except exec of file with setuid bit
- Setuid system call
  - seteuid(newid) can set EUID to
    - Real ID or saved ID, regardless of current EUID
    - Any ID, if EUID is root

- Details are actually more complicated
  - Several different calls: setuid, seteuid, setreuid

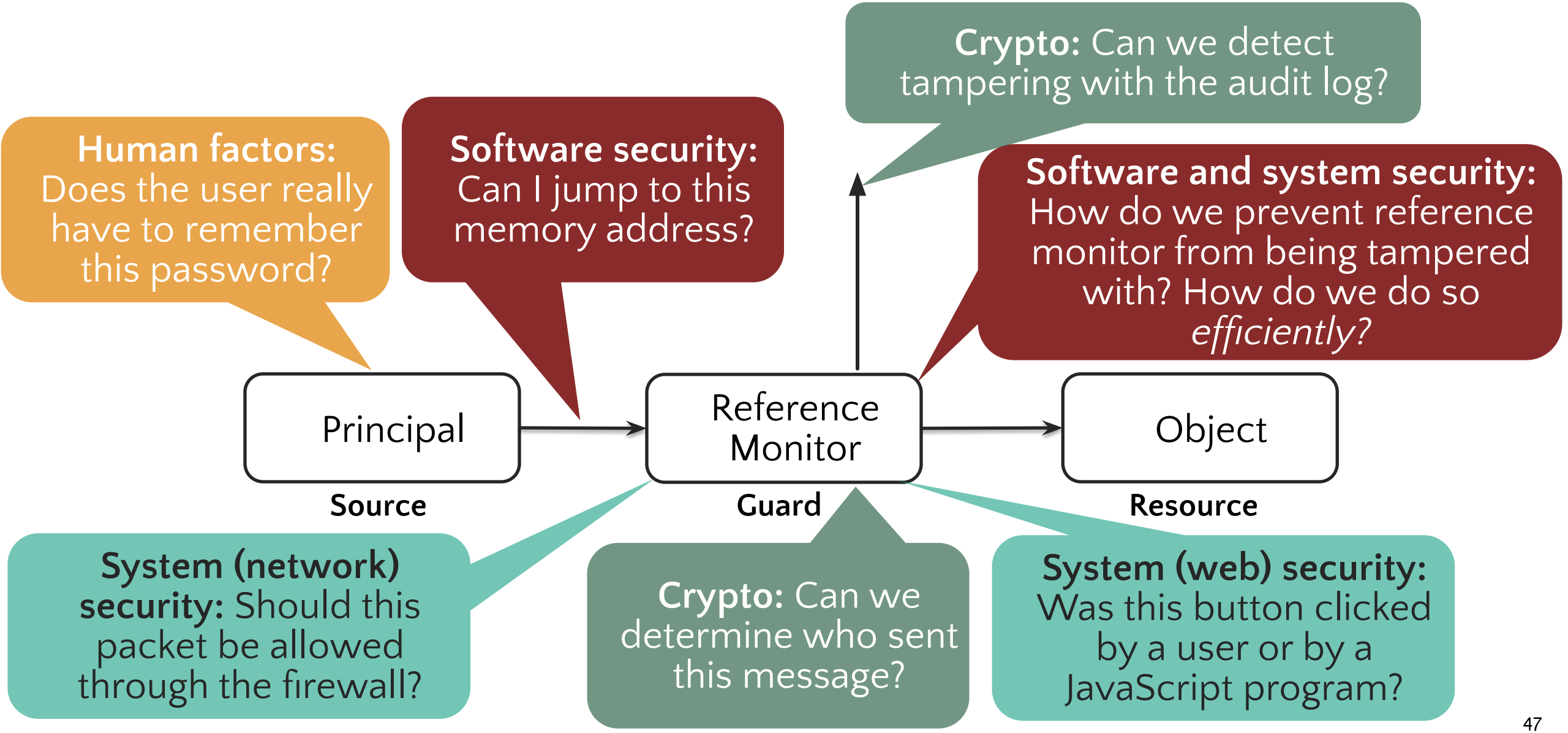
Setid bits on executable Unix file

- Three setid bits
  - Setuid – set EUID of process to ID of file owner
  - Setgid – set EGID of process to GID of 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

Example


RUID 25
…;
…;
exec( );
Owner 18
SetUID
program
Owner 18
-rw-r--r--
file
read/write
Owner 25
-rw-r--r--
file
read/write
…;
…;
i=getruid()
setuid(i);
…;
…;
RUID 25
EUID 18
RUID 25
EUID 25

Abstract Access Control Model



Takeaways

- A reference monitor is the security mechanism in an abstract model of access control
- The “Gold” standard
  - Authentication, authorization, audit
  - Relevant to every corner of security!
- Know types of authorization mechanisms
  - Mandatory vs discretionary, capabilities vs ACL
- Principal ≠ principle

Ευχαριστώ και καλή μέρα εύχομαι!

Keep hacking!