2550 Intro to cybersecurity

L17: Authorization

abhi shelat

Thanks Christo for slides!

Authentication:

(Verifying a claim about identity by a subject on behalf of a principal.

- → Unforgeable
- Unguessable
- Revocable
- Resettable

} informal properties.

Authorization

After Authenticating a subject, what next?

→ determination of whether a subject can access a resource.

— "policy specification"

Principle-Subject-Object

resources, file, network, GPU

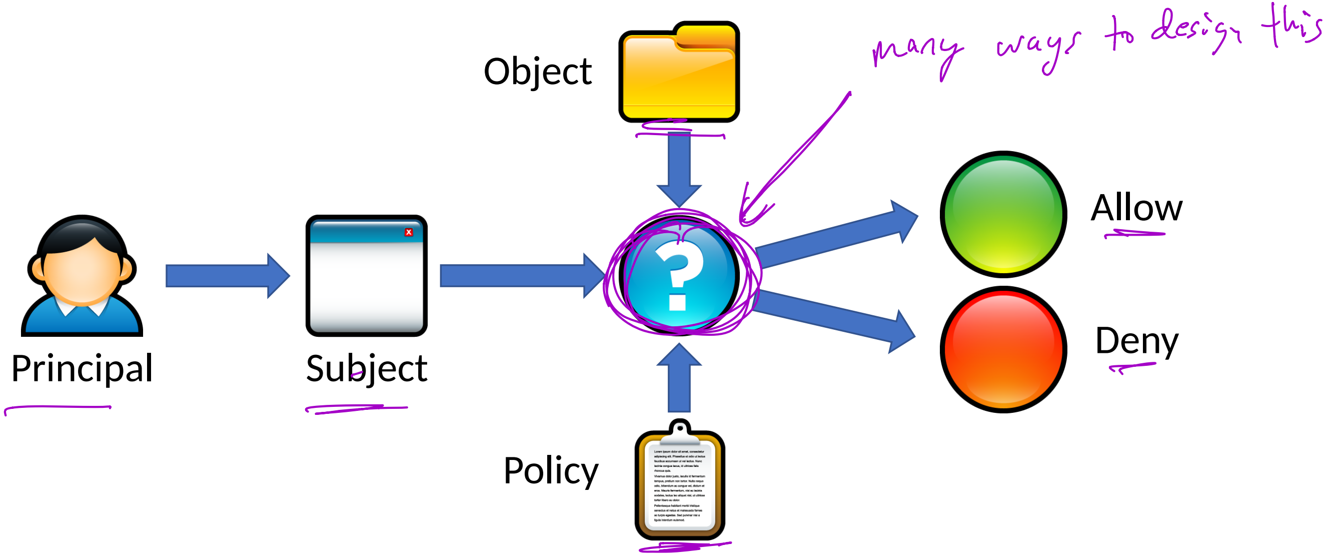
users "ideal notion of a decision maker, entity"

process, program operating on behalf of the principle.

e.g. browser, shell, email program

Access Control Check

- Given an access request from a subject, on behalf of a principal, for an object, return an access control decision based on the policy



Two main types of access control

- Discretionary access control

- Mandatory access control

Discretionary access control

- subjects are allowed to share the permissions they hold with other subjects

- common, e.g. UNIX, WINDOWS, dropbox, google docs. easy for collaboration.

ACL

- Access control list - column of the matrix

object.

subjects

	o_1	o_2	o_3
s_1	RW	RX	
s_2	R	RWX	RW
s_3		RWX	

Access control matrix

UNIX: "all", "group", "owner"

Capability-based systems

	o_1	o_2	o_3
s_1	RW	RX	
s_2	R	RWX	RW
s_3		RWX	

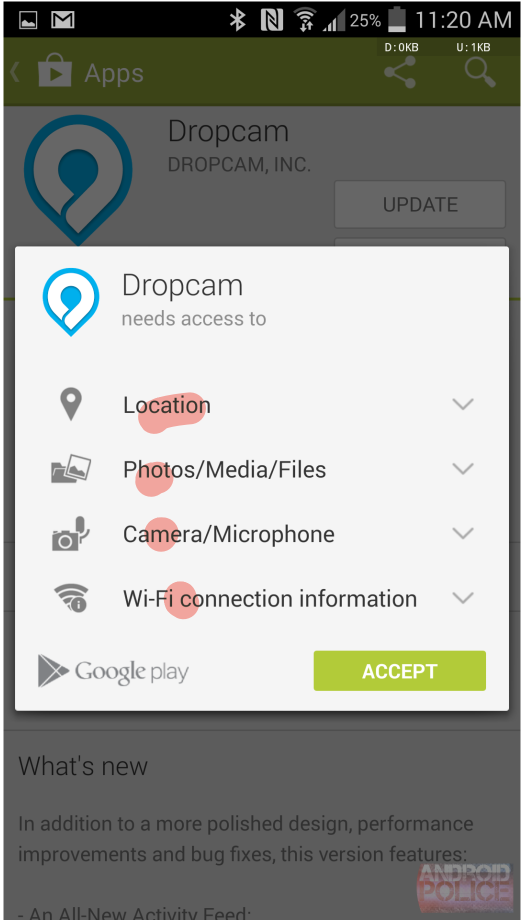
Authorization specified by enumerating the "capabilities" of each subject.

Capability-based Access Control

- Principals and subjects have capabilities which:
  - Give them access to objects
    - Files, keys, devices, etc.
  - Are transferable and unforgeable tokens of authority
    - Can be passed from principal to subject, and subject to subject
    - Similar to file descriptors
- Why do capabilities solve the confused deputy problem?
  - When attempting to access an object, a capability must be selected
  - Selecting a capability inherently also selects a master

Android/iOS Capabilities

- Android and iOS support (relatively) fine grained capabilities for apps
  - User must grant permissions to apps at install time
  - May only access sensitive APIs with user consent
- Apps can "borrow" capabilities from each other by exporting *intents*
  - Example: an app without camera access can ask the camera app to return a photo



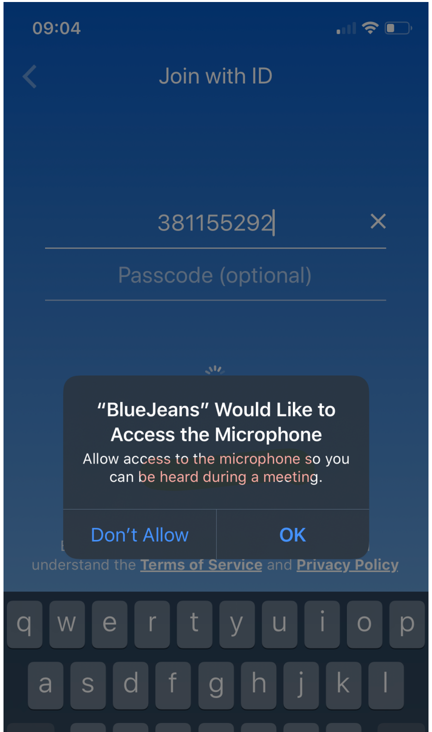
Dropcam
needs access to

What's new

In addition to a more polished design, performance improvements and bug fixes, this version features:

- An All-New Activity Feed:

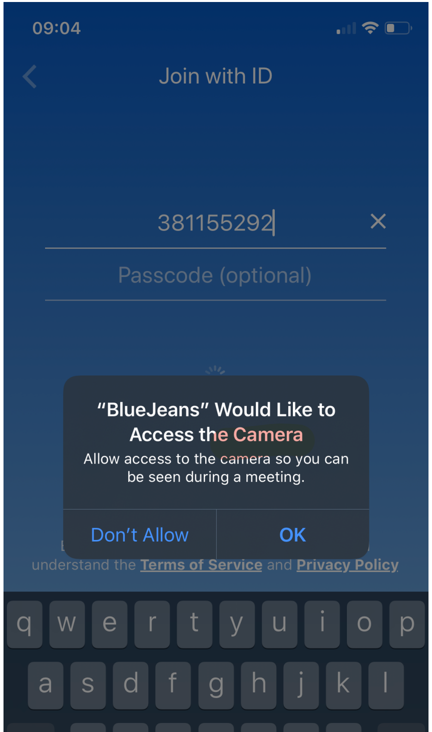
Android/IOS just-in-time capability



"BlueJeans" Would Like to Access the Microphone

Allow access to the microphone so you can be heard during a meeting.

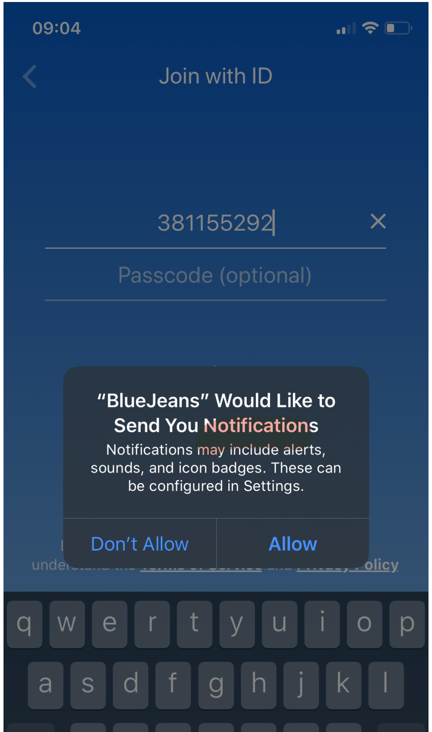
understand the Terms of Service and Privacy Policy



"BlueJeans" Would Like to Access the Camera

Allow access to the camera so you can be seen during a meeting.

understand the Terms of Service and Privacy Policy

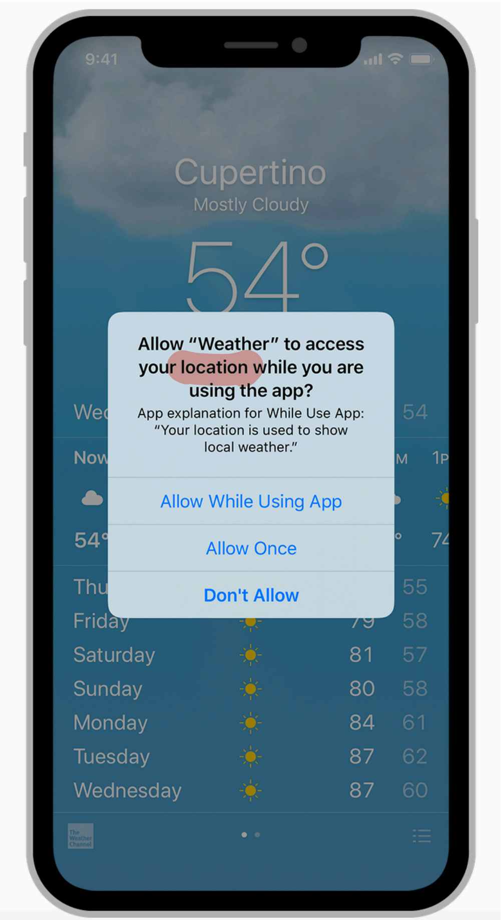


"BlueJeans" Would Like to Send You Notifications

Notifications may include alerts, sounds, and icon badges. These can be configured in Settings.

Per-event capability

- fine-grained capabilities
- UI/UX + security problem
- risk assessment
  - tradeoff usability/utility with security



Allow "Weather" to access your location while you are using the app?

App explanation for While Use App: "Your location is used to show local weather."

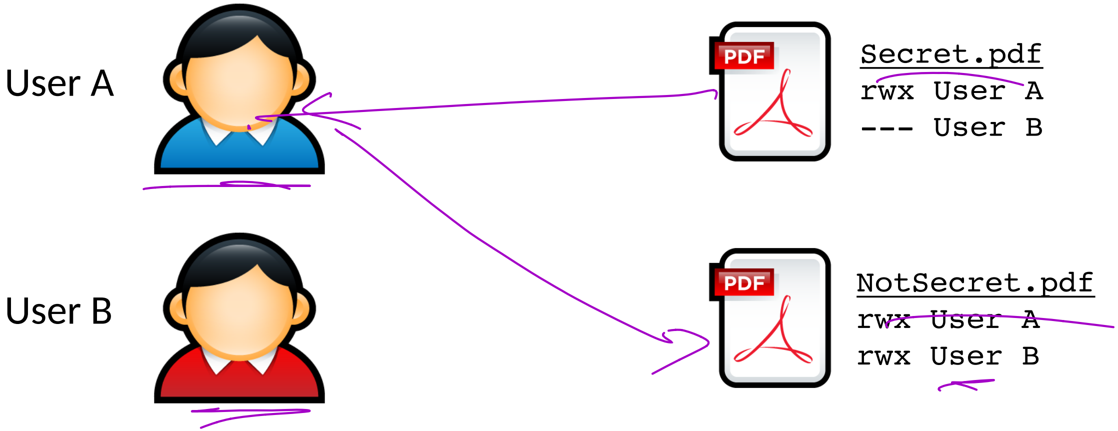
Allow While Using App

Allow Once

Don't Allow

Failure of DAC

- DAC cannot prevent the leaking of secrets



User A

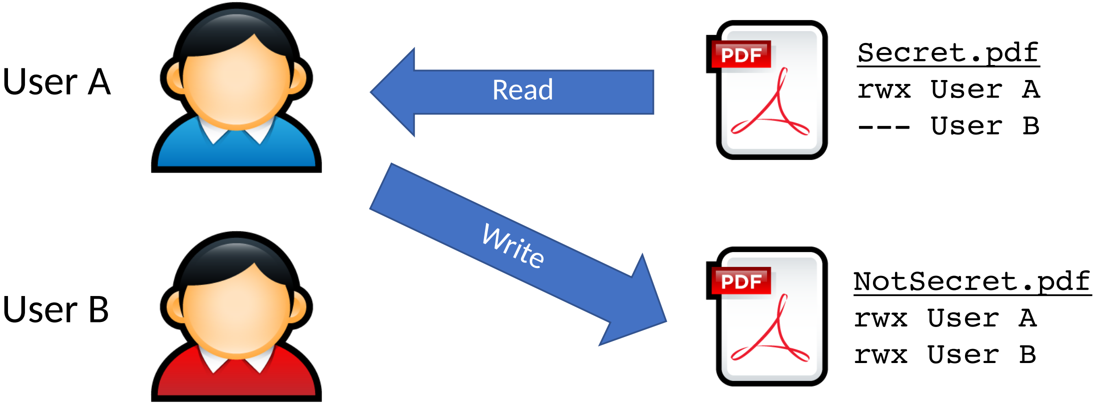
User B

Secret.pdf
rwx User A
--- User B

NotSecret.pdf
rwx User A
rwx User B

Failure of DAC

- DAC cannot prevent the leaking of secrets



User A

User B

Read

Write

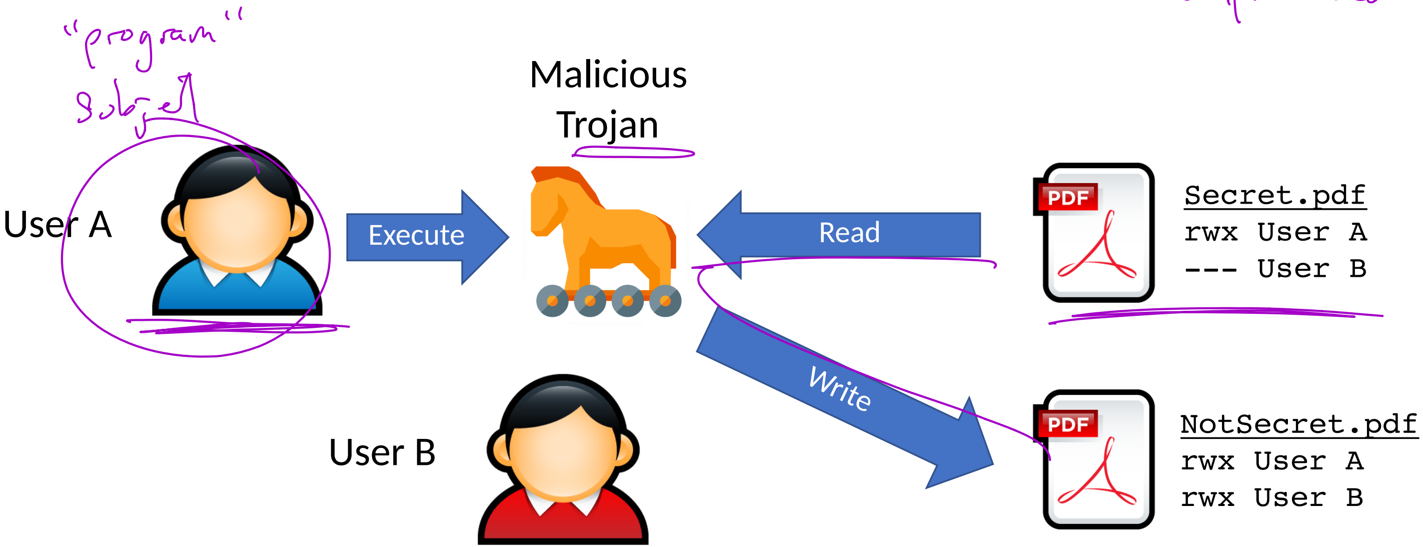
Secret.pdf
rwx User A
--- User B

NotSecret.pdf
rwx User A
rwx User B

Failure of DAC

· Loss of confidentiality if subject is compromised

- DAC cannot prevent the leaking of secrets



Secret.pdf
rwx User A
--- User B

NotSecret.pdf
rwx User A
rwx User B

Mandatory Access Control

- system policy determines access control.

  ~~users~~ subjects cannot share or give permissions to other subjects.

Mandatory Access Control Goals

- Restrict the access of subjects to objects based on a system-wide policy

Bell-Lapadula (1973)

"No read up, no write down"

System Model: abstract machine that captures the operation

Security Policy: what defines the security guarantee.

BLP System Model

Clearances: subjects s had clearances, "maximum clearances"

subjects could operate at any level ≤ their top clearance $L_c(s)$

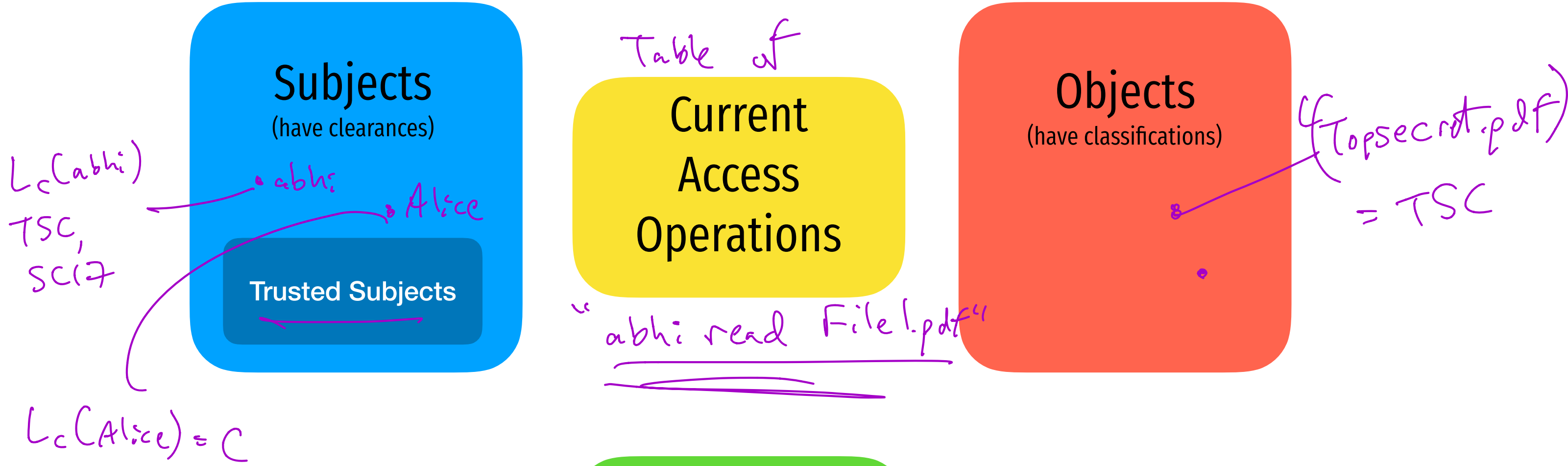
eg ?

$$\left\{ \begin{matrix} TS \\ S \\ C \\ P \end{matrix} \right\}$$

Classifications:

object o had classifications $L_c(o)$

BLP System State



Subjects

(have clearances)

Trusted Subjects

Table of

Current Access Operations

Objects

(have classifications)

L_c(abhi)
TSC,
SC(7

• abhi

• Alice

L_c(Alice) = C

"abhi read File1.pdf"

("Topsecret.pdf")
= TSC

ACL

	O1	O2	O3
S1	r		
S2			r
S3			
S4			

BLP Idea

A computer system is in a state, and undergoes state transitions whenever an operation occurs..

System is secure if all transitions satisfy 3 properties: *

"No read up"

Simple: $L_c(s) \geq L_c(o)$. "subjects only read file that have same or lower clearance"

No write down:

Star: when s writes $L_c(s) \leq L_c(o)$

Discretionary: All accesses also satisify the ACLs.

BLP Idea

A computer system is in a state, and undergoes state transitions whenever an operation occurs..

System is secure if all transitions satisfy 3 properties:

Simple: S can read O if S has higher clearance

Star: S can write O if S has lower clearance.

Discretionary: Every access allowed by ACL.

Users are trusted

Subjects are not trusted. (Malware)

Not Enough

Alice

① operates at TSC

reads this file →

② go down to ℓ levels

③ write your buffer to →

Bob.

TopSecret.pdf

rwx User A

--- User B

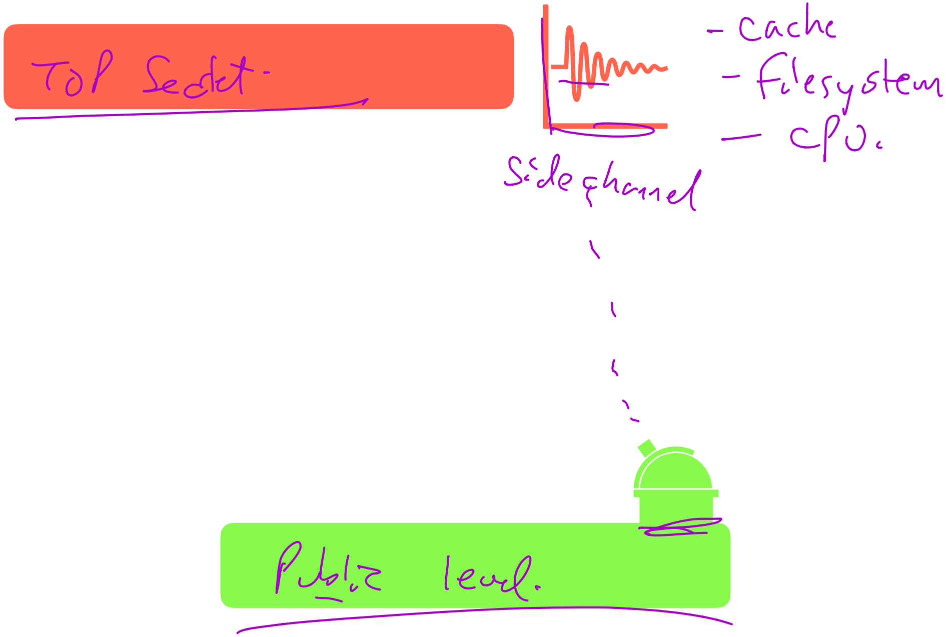
NotSecret.pdf

rwx User A

rwx User B

TRANQUILITY: processes can only ↑ in security clearance.

Not Enough: Covert channels

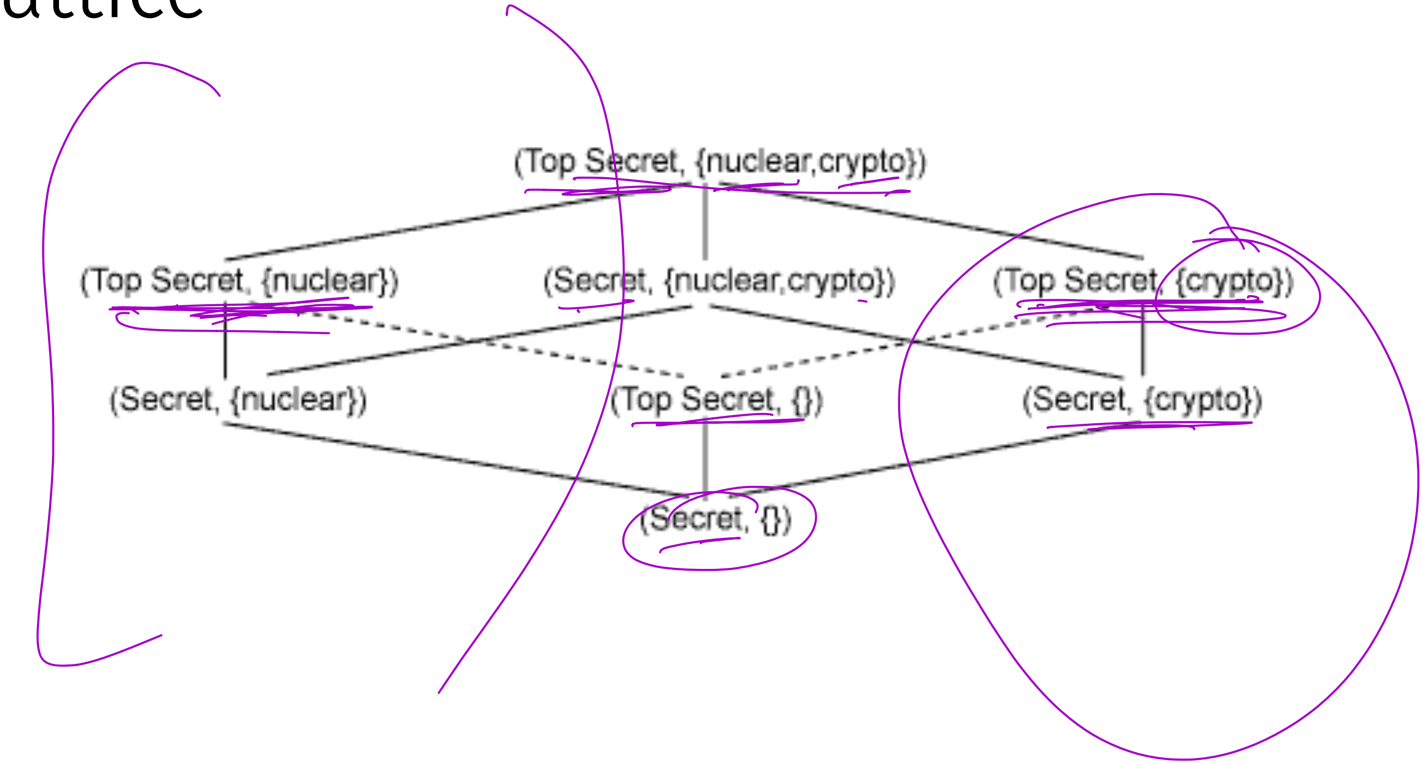


Security Lattice

Compartments: SIGINT, HUMINT, PINK FLAMINGO

Ordering between (Level, Compartment)

Lattice


(Top Secret, {nuclear,crypto})
(Top Secret, {nuclear})
(Secret, {nuclear,crypto})
(Top Secret, {crypto})
(Secret, {nuclear})
(Top Secret, {})
(Secret, {crypto})
(Secret, {})

Need-to-Know policy

Subjects only given access to objects that are necessary for functionality

Hybrid

SELinux, TrustedBSD: MAC + DAC system

Confidentiality? What else?

ensure

Integrity

Biba Integrity Policy

BLP - "No read up, No write down"

↕

Biba - "no read down, no write up"

Comparison

BPL

Biba

- Offers confidentiality
- “Read down, write up”
- Focuses on controlling reads
- Theoretically, no requirement that subjects be trusted
  - Even malicious programs can’t leak secrets they don’t know

Comparison

BPL

- Offers confidentiality
- “Read down, write up”
- Focuses on controlling reads
- Theoretically, no requirement that subjects be trusted
  - Even malicious programs can’t leak secrets they don’t know

Biba

- Offers integrity

Comparison

BPL

- Offers confidentiality
- “Read down, write up”
- Focuses on controlling reads
- Theoretically, no requirement that subjects be trusted
  - Even malicious programs can’t leak secrets they don’t know

Biba

- Offers integrity
- “Read up, write down”

Comparison

BPL

- Offers confidentiality
- “Read down, write up”
- Focuses on controlling reads
- Theoretically, no requirement that subjects be trusted
  - Even malicious programs can’t leak secrets they don’t know

Biba

- Offers integrity
- “Read up, write down”
- Focuses on controlling writes

Comparison

BPL

- Offers confidentiality
- “Read down, write up”
- Focuses on controlling reads
- Theoretically, no requirement that subjects be trusted
  - Even malicious programs can’t leak secrets they don’t know

Biba

- Offers integrity
- “Read up, write down”
- Focuses on controlling writes
- Subjects must be trusted
  - A malicious program can write bad information

Abstraction, Design, Impl.

Failures of Operation

Social engineering

Baiting

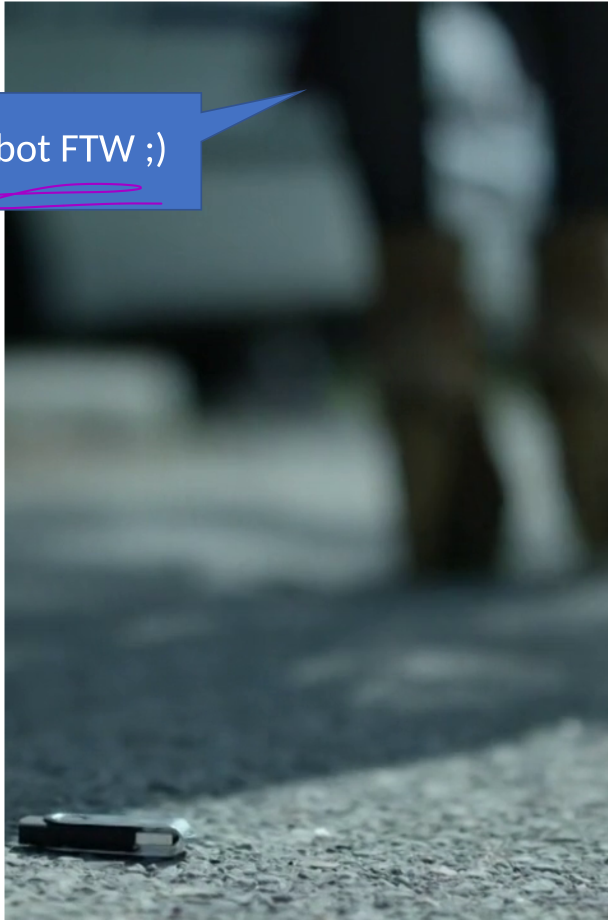
Very simple physical attack

1. Preload USB keys with malware
2. Drop the keys in public, near victims
3. Wait for victims to pick up and plug in
4. Victim executes malware
   - Either by accident due to curiosity
   - Or autorun by the OS (e.g. Windows)

Baiting

Very simple physical attack

1. Preload USB keys with malware
2. Drop the keys in public, near victims
3. Wait for victims to pick up and plug in
4. Victim executes malware
   - Either by accident due to curiosity
   - Or autorun by the OS (e.g. Windows)



Mr. Robot FTW ;)

Tailgating

Technique used by penetration testers

Goal: break in to a secure facility

- Security guards at the main entrance
- All doors have keycard access control

Idea:

1. Wait for an unsuspecting employee to open a door
2. Follow them inside
3. Leverages courtesy bias and ingroup bias

NO TAILGATING
Each person must present his/her Apple I.D. Badge to the card reader to confirm valid access.