Lec 23

OS Security Theory

Security is hard

- security vs usability tradeoff
- attacker/defender asymmetry

Fair to say computer security is unsolved problem
- analogy to aviation safety

Security terminology

Principal: agent making a request: user, process, administrator, ...

Authorization: policy for who can do what — permissions

Authentication: mechanism for determining user is who they say

Encryption: mechanism to achieve privacy even in public spaces

Auditing: log of which users made which requests

Security goals (OS)

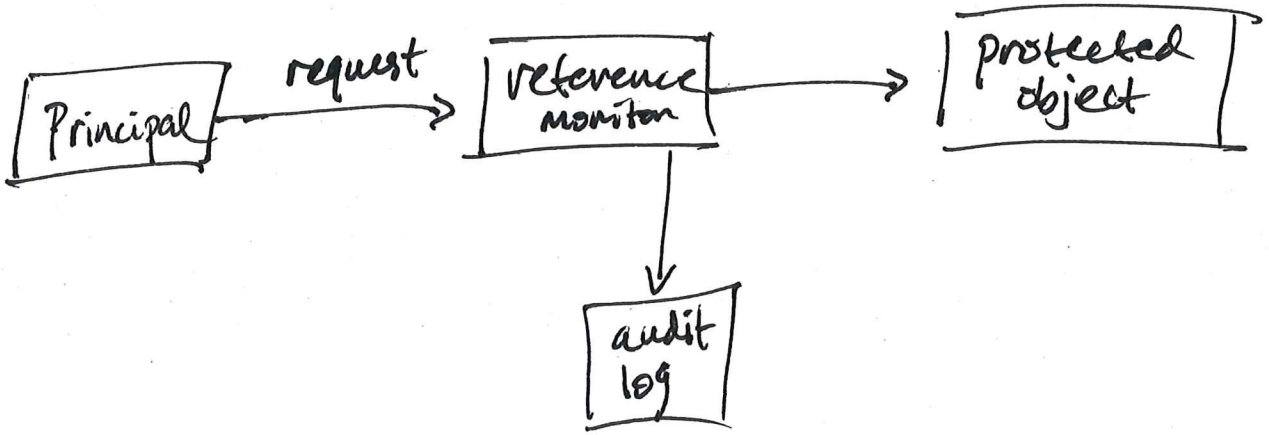
Data confidentiality:
- private data remains private
- no unauthorized reads

Data integrity:
- attacker cannot corrupt data
- no unauthorized writes

System availability: users can get their work done even if system is attacked w/ denial of service

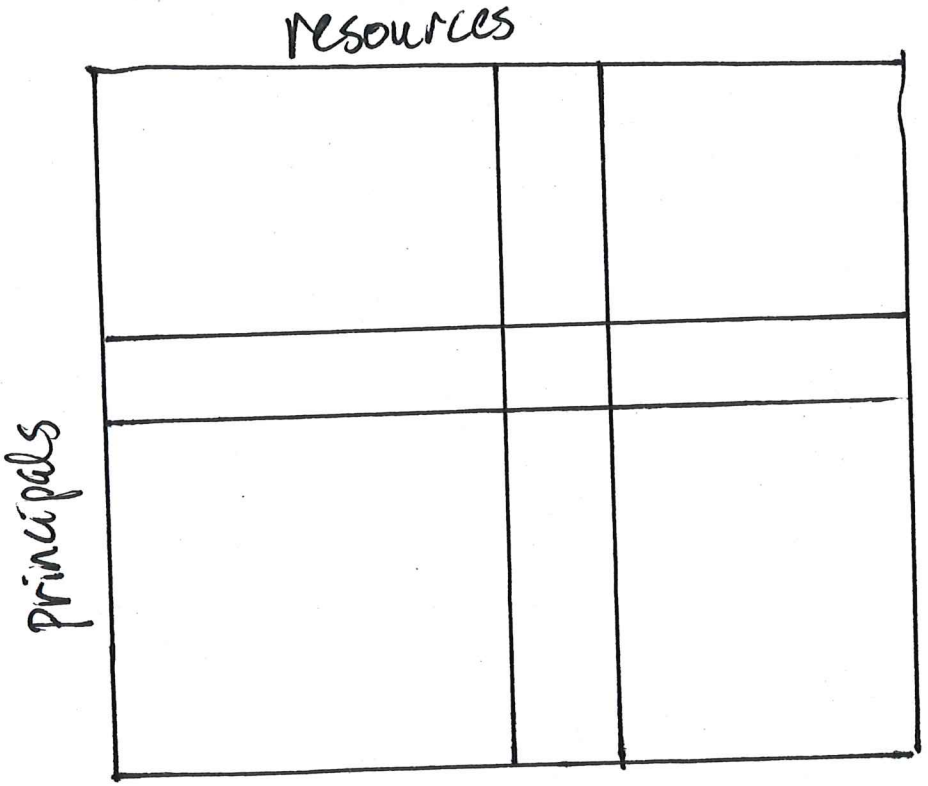
Access Control Model

Complete mediation: every request is explicitly checked for authorizat



Example: Kernel mediates user processes' access to files

Access Matrix



each cell lists the operations that principal can perform on that resource

Example:
James can read + write course web
but students can only read

Example: Unix file permissions

Example: Smartphone app permissions

Too big to store explicitly (and too unwieldy!)

Instead: exploit sparsity + regularity
- for each resource (column), list the authorized (principal, operation) pairs
  - access control lists
- for each principal (row), list the authorized (resource, operation) pairs
  - capabilities

Principle of least privilege

Grant each principal smallest set of permissions that they need to do their job

↳ minimize code running in Kernel mode

↳ minimize code running as administrator

Processes execute as their user...

Authorization w/ intermediaries

Trusted Computing Base (TCB):

set of assumptions needed to guarantee security policy enforced

↳ often larger than you think!

Confused Deputy:

- services often have permission to do things on behalf of many users
- can trick the service into doing something that you couldn't do on your own
- example: application-level user names probably don't map to OS usernames
  ↳ bug in app-level authorization can leak/corrupt all users' data!

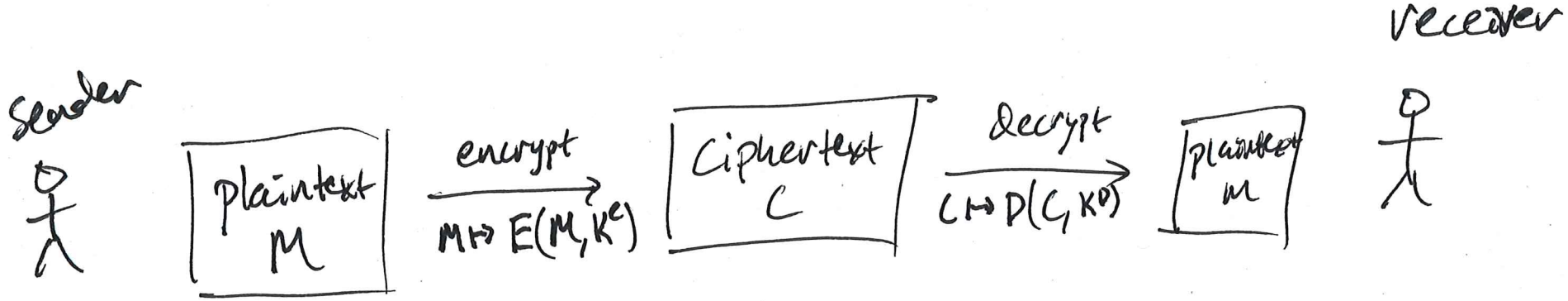
Authentication

- common approach: passwords
  - short passwords are easier to remember

- where are passwords stored?
  - in a file?
  - encrypted in a file?
  - salt before hashing

- typical human chosen passwords are terrible

Encryption



K^e, K^d encryption/decryption keys

must assume attacker knows E() and D()

Symmetric encryption

$K^e = K^d$ is a shared secret

example (bad): $E(M, K) = M \text{ xor } K$

$D(C, K) = C \text{ xor } K$

examples (good): AES, ...

Asymmetric encryption

- Keys come in (public, private) pairs
- each principal has their own pair
- each public key is published; private key kept secret

authentication: encrypt msg w/ private key

secrecy: encrypt msg w/ public key

can combine: encrypt w/ public key of receiver
and private key of sender