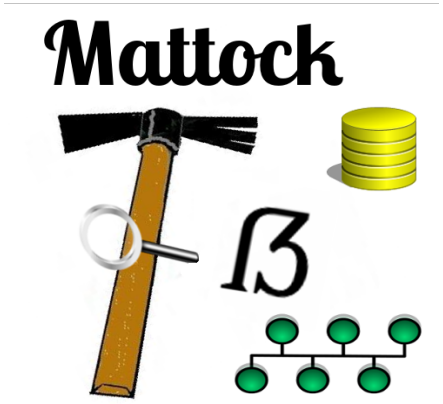
# Todays schedule

- Asynchronous processing & tool-chain approach
- **Integrity, privilege separation and capabilities.**
- CarvFS & MinorFS
- MattockFS core design
- MattockFS as distributed-framework building block
- Installation (hands on)
- File-system as API (hands on)
- Python API (hands on)




# MattockFS


Computer-Forensics File-System

Integrity, privilege separation & the capability based security model.

# Principle Of Least Authority

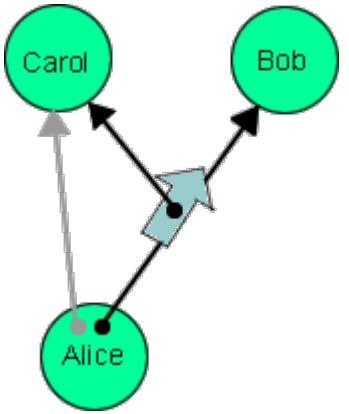
- Only grant permissions required to complete the task requested.
- Principle can be applied at each granularity level.
- Privilege separation: minimize reach potential breach.
- Revocation: Assure temporal least-authority.
- Need to know vs need do; secrets vs powers

# Public/global mutable state

- Makes composite systems hard to analyze or review.
- Makes composite systems hard to test.
- High potential for violating the Principle of Least Authority.
- Giant hurdle for reducing trusted code-base size.

# Capabilities

- Authority-only security tokens (no identity)
- Capabilities both designate and authorize
- In fact: designation IS authorization
- Dynamic least authority.

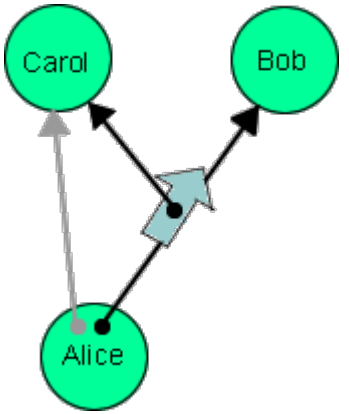


# Sparse capabilities

- Capability 'as data'.
- Basically a strong password without an identity
- Some well known examples
  - Rumpelstiltskin
  - https://docs.google.com/file/d/**0B_sqxoHXGz5ZcEhCOMJWRzRjbmM**/edit
- MinorFS:
  - /minorfs/cap/**rw-5BA7FFDP4HM1L887BZAK7G857GKJHAUEAK992HCH5DGH59GHWB1S**
- Non human-memorable 'authority' token

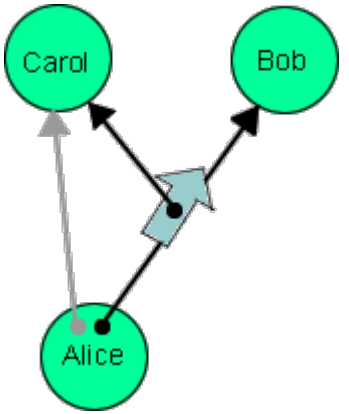
# Object capabilities

- More secure than sparse capabilities
- Basically object-reference as capability
- Ambient-authority and static-mutable-state free subset of Object Oriented Design principles.



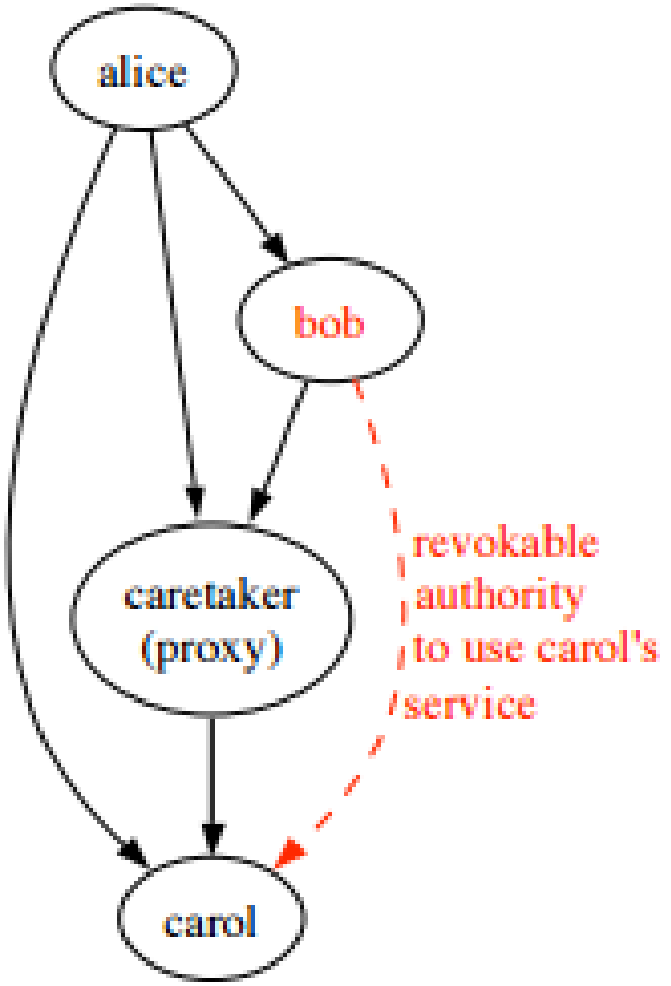
# Object capabilities

- More secure than sparse capabilities
- Basically object-reference as capability
- Ambient-authority and static-mutable-state free subset of Object Oriented Design principles.



# Caretaker pattern

- Revocation of delegated authority.



# Linux process granularity facilities

- Running under different UIDs
  - Base file-system access controls
  - UID based firewall rules.

- Unix domain sockets for delegating file handles.
- Mandatory Access Controlls
  - AppArmor
  - SELinux


- User-space file-systems
  - Sparse capabilities

# More capabilities

- Object capability languages
  - E
  - Joe-E
  - Caja’s Secure Ecma Script
- Networked
  - E
  - Foolscap (python)
  - Cap’n proto

# Sealed Digital Evidence Bag

- Storage format concepts for data integrity
- Identified need for immutability
- Identified need for trusted provenance logs

# Forensic frameworks: why care?

- The integrity of the (computer) forensic process is of foremost importance.
- Provenance-logs should be 100% trustworthy.
- A bad (*1) module might compromize the integrity of more than just it’s own tasks.

# Shared mutable data in OCFA

- Meta-data
- Evidence data
- Derived evidence data
- Provenance logs

# Shared mutable data in OCFA

- Meta-data
- Evidence data
- Derived evidence data
- Provenance logs

- Anti forensics ?
- Forensic process integrity ?
- Forensic frameworks need evidence sealing too!

# Bad modules

- We can’t reinvent or security-review every wheel!
- Large scale forensic data processing WILL use many 3th party tools and libraries. Each with its own bugs.
- A bug may cause a module to execute arbitrary code. This may affect any reachable shared mutable state.
- Anti-forensic data might actively exploit such bugs, directly and accurately target the integrity of the forensic process.

# Conclusions

- In the light of process integrity and modern anti-forensics techniques, privilege separation is a must.
- Ideally everything would be implemented in a capability secure programming language.
- A memory secure language mitigates only  part of the threath.
- Using OS facilities combined with sparse capabilities and user space file-systems can accommodate existing tools and libraries.
- We need “need to do” based least authority for integrity, NOT “need to know” for confidentiality between modules.