SPM INFORMATION-PROTECTION DAEMON

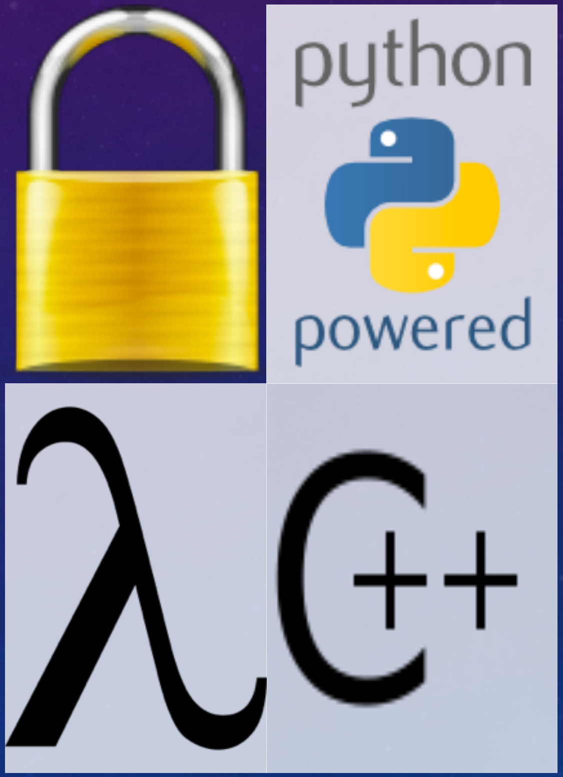
POLICY ENFORCEMENT AND FINE-GRAINED ACCESS CONTROL WITH DECIDABILITY AND SAFETY PROBLEM

FOREWORD

EXPECT SOME TURBULENCE

PRESENTATION OUTLINE

- Project Summary: Motivation, Problem and Solution
- System Architectures
  - Network: Client-Server Model
  - Database: Relational Database Model
  - Software: Lisp-Trampoline Event-Driven Model
  - Security: Rights management and Encryption
- Literature Sources
- Looking Forward

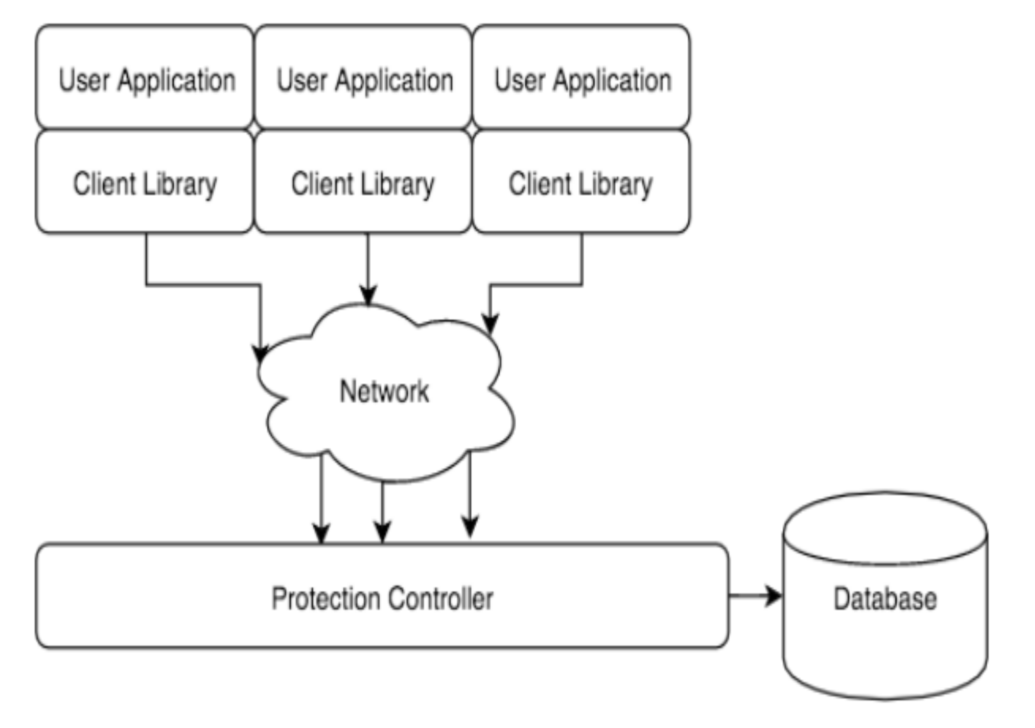


PROJECT SUMMARY

- Motivation
  - Unix permissions insufficient for modeling some access control policies
  - Remote-mounted network file systems rely on local policy enforcement
  - Only share resources modeled as file objects
  - Break down with large number of users
  - Learning experience: study while you work
- Problem
  - No single unified, cross-platform, portable, shared, and free SPM-like policy enforcement mechanism
- Solution
  - SPMd – portable and free (MIT License) SPM-like policy enforcement daemon

SYSTEM ARCHITECTURE: NETWORK POV

- Simple Client-Server Model
  - Distributed system fundamentally incompatible with centralized policy enforcement
  - Guarantees high consistency, integrity, and confidentiality at the cost of availability (relative to P2P)
  - Neither threaded nor forking (talk about this more later)
  - Database is *not* a separate process
  - Assume clients not willfully colluding



SYSTEM ARCHITECTURE: DATABASE POV

Table Name	Columns
subjects	subject, key, super
links	subject1, subject2
filters	subject1, subject2, ticket
rights	subject1, ticket, target, object
objects	localpath, dir

- Objects data stored in file system external to database for best performance
- Links and filters unidirectional
- Ticket: High-level rights, converted too and from objects on the fly

SYSTEM ARCHITECTURE: SOFTWARE POV

- Programming Languages
  - C++ (efficiency): establishes daemon, calls into python runtime library, low level operations
  - Python (safety and ease): implements safe parsing of untrusted messages, network event loop
  - SQL98 (support): database queries, CRUD operations determined by python parsing
- Execution Model
  - Continuations (Stackless!)
    - Recursion as deep as you could ever want…
  - Fully asynchronous: One thread = N clients
    - What happens when you have 10,000 connections?
  - Implementation: Trampoline loop over anonymous functions

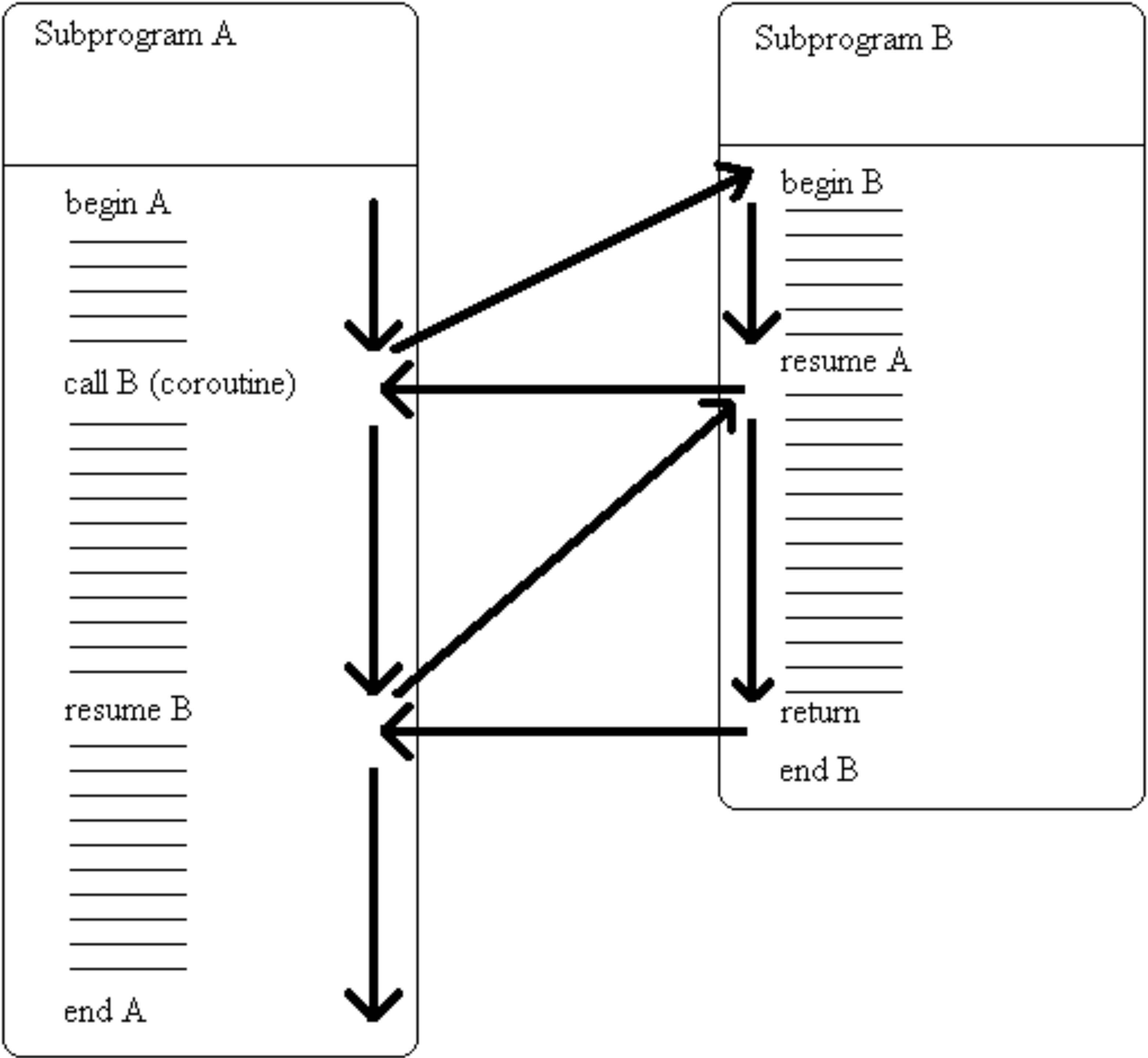
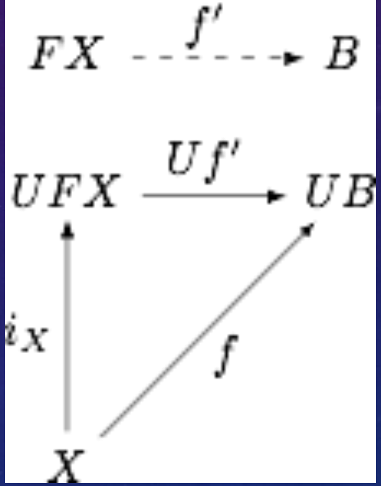


Image Credit:

Tevfik AKTUĞLU
CC by Attribution

SYSTEM ARCHITECTURE: SECURITY POV

- Rights Management
  - SPM-inspired model
  - More restrictive, decidable of super users are trusted
- Confidentiality
  - XTEA stream cipher negotiation upon authentication
  - Key generation with PKCS and nonce
  - Unique keystream for every user and every session
- Integrity
  - Shared-key on the user password
  - Assumes rights of one subject at a time
  - Each message contains authentication code to prevent tampering
- Availability Notes



$$\begin{tikzcd}
FX \arrow[r, dashed, "f'"] & B \\
UFX \arrow[r, "Uf'"] & UB \\
X \arrow[u, "i_X"] \arrow[ur, "f"'] &
\end{tikzcd}$$

LITERATURE SOURCES

Kaliski, B. "PKCS #5: Password-Based Cryptography Specification." *IETF*. The Internet Society, Sept. 2000. Web. 26 Feb. 2016.

McGrew, D. "AES-GCM and AES-CCM Authenticated Encryption in Secure RTP (SRTP)." *IETF Tools*. The Internet Society, 26 Jan. 2011. Web. 26 Feb. 2016.

Needham, Roger M., and David J. Wheeler. "Tea Extensions." (n.d.): n. pag. Oct. 1997. Web. 26 Feb. 2016.

"Python3 Documentation." *Python.org*. Python Software Foundation, n.d. Web. 26 Feb. 2016.

Sandhu, Ravinderpal Singh. "The Schematic Protection Model: Its Definition and Analysis for Acyclic Attenuating Schemes." *Journal of the ACM JACM J. ACM* 35.2 (1988): 404-32. *University of Pittsburgh*. Web. 26 Feb. 2016.

Watson, Devin. "Linux Daemon Writing HOWTO." *Linux Daemon Writing HOWTO*. N.p., May 2004. Web. 26 Feb. 2016.

FORWARD

EXPECT MORE TURBULENCE