

Principles of Information Security

CS 5323 Lecture 22

Prof. William Winsborough
November 29, 2007

Business

- Final Exam: Thursday 6 December 8-10:30pm in regular lecture room
- The final is not comprehensive, but will cover material since the last exam
 - Material presented by guest lecturers will not be covered
- Today we will continue to study:
 - The Flask Security Architecture: System Support for Diverse Security Policies
 - <http://www.nsa.gov/selinux/papers/flask-abs.cfm>

Flask

- A security architecture
 - Implementation: A microkernel-based operating system
- Provides mechanisms to support a wide range of security policies
 - Reported implementation supports:
 - MLS (Bell-LaPadula)
 - Type enforcement
 - Identity-based (discretionary) access control
 - Dynamic RBAC
- Architecture addresses issues such as
 - Prompt revocation of permissions
 - Atomicity in interleaving of policy updates and controlled operations

Revocation

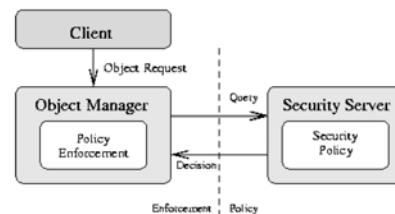
- Atomicity requires that once a permission has been revoked, the corresponding service must no longer be provided
 - Problem: permissions migrate
 - E.g., Unix file permissions are checked on open, not on read/write
 - Similar problems arise with capabilities, access rights in page tables, open Inter-Process Communication (IPC) connections, operations already in progress
 - In progress operations
 - Abort
 - Restart, forcing a new check of the revoked permission
 - Wait (only safe when known will no block indefinitely— otherwise the revocation may not take effect)

Flask Architecture

- Flask implementation uses a microkernel-based OS (derived from Fluke)
- Flask architecture requires only that the underlying OS include a reference monitor and provide separation between processes

Flask Architecture

- The primary goal of the architecture is to provide for flexibility in the security policy by ensuring that the subsystems always have a consistent view of policy decisions regardless of how those decisions are made or how they may change over time.



What the Object Manager Gets

- Interfaces for retrieving from a security server:
 - Access decisions
 - Labeling (security attributes of objects)
 - Polyinstantiation decisions
- Access Vector Cache (AVC)
 - Minimize performance overhead
 - Managed by the security server
- The ability to register callbacks to receive notifications of policy changes

29 November 2007

Winsborough CS 5323 Lecture 22

7

What Object Managers Must Do

- Mechanism for assigning labels to objects
- Provide callbacks—routines to handle policy changes
 - Will be called by the security service when policy change occurs

29 November 2007

Winsborough CS 5323 Lecture 22

8

Object Labeling

- Each object has a set of security attributes
 - Two representations:
 - Variable-length string (*security context*)
 - Security identifier (SID)
 - Lightweight
- When an object is created
 - It is assigned an SID represented the object's security context
 - That security context depends on the client that requested the object's creation and the environment (e.g., directory in which a file is created)

29 November 2007

Winsborough CS 5323 Lecture 22

9

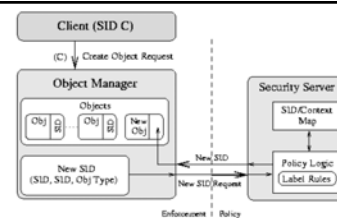


Figure 2: Object labeling in Flask. A client requests the creation of a new object from an object manager, and the microkernel supplies the object manager with the SID of the client. The object manager sends a request for a SID for the new object to the security server, with the SID of the client, the SID of a related object and the object type as parameters. The security server consults the labeling rules in the policy logic, determines a security context for the new object, and returns a SID that corresponds to that security context. Finally, the object manager binds the returned SID to the new object.

29 November 2007

Winsborough CS 5323 Lecture 22

10

AVC

- Access vector cache (AVC) module
 - Common library shared by object managers
 - Coordinates policy between object manager and security server
 - Requests from object manager for policy decisions
 - Requests from security server for policy changes

29 November 2007

Winsborough CS 5323 Lecture 22

11

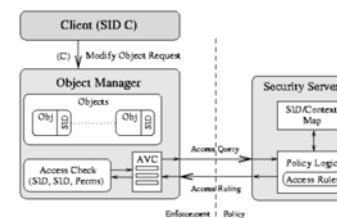


Figure 3: Requesting and caching security decisions in Flask. A client requests the modification of an existing object from an object manager. The object manager queries its access vector cache (AVC) module for an access ruling for the (client SID, object SID, requested permissions) triple. If no valid entry exists, then the AVC module sends an access query to the security server. The security server consults the access rules in the policy logic, determines an access ruling, and returns the access ruling to the AVC module.

29 November 2007

Winsborough CS 5323 Lecture 22

12

Revocation Support Mechanisms

- Object managers keep local copies of security decisions
 - Explicitly in the AVC
 - Implicitly in the form of migrated permissions
- Atomicity is achieved by
 - When policy change is complete, object manager behavior reflects the change
 - Object managers complete policy change in a timely manner
- Protocol
 - Security server notifies all object servers of policy change
 - Object servers update their state accordingly
 - Object manager notifies the security server that change is complete (allowing subsequent policy changes to then be performed in order)

29 November 2007

Winsborough CS 5323 Lecture 22

13

Security Server Must

- Provide security policy decisions
- Maintain mapping between SIDs and security contexts
- Provide SIDs for newly created objects
- Manage AVCs

29 November 2007

Winsborough CS 5323 Lecture 22

14