

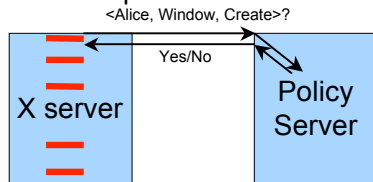
- Enable Application-level Reference Monitors
- Application control is necessary for its objects
  - Reference Monitor implementation is complex
  - User-level Reference Monitor support is there
  - Can we build tools to support implementation?

- Challenges
- SELinux depends on the Xserver to control copies
  - Bugs found in LSM design; 3 years to acceptance
  - SELinux Policy Server enables authorization checks
  - Need to develop an approach to automate placement

## Reference Monitor Design and Implementation

### Security Architecture Requirements

- Hook placement is crucial.
- Hooks must pose the correct authorization request.



Security-enhanced Operating System

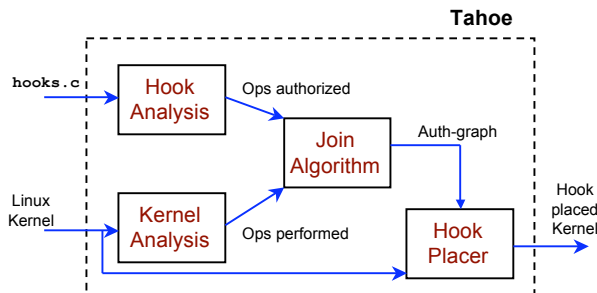
### So... what is our contribution?

- A technique for automatic *Application-Level Authorization Policy Enforcement (ALPEN)*.
- Builds on prior work: TAHOE [GJJ, CCS 2005]
- TAHOE: Automated hook placement compared to LSM
  - Input: Hooks for a security policy (e.g., SELinux), and a non-hook-placed Linux kernel
  - Output: Hook-placed Linux kernel.
  - Key Idea: Policy operations can be associated with kernel functions.

## Mechanisms

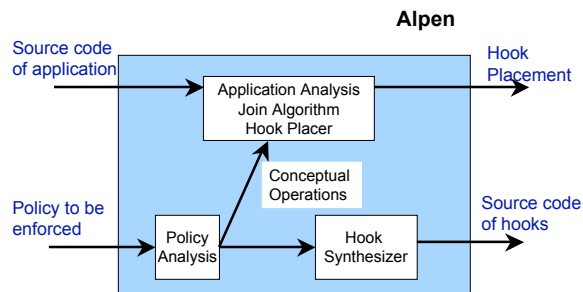
## Results and Challenges

### Tahoe Mechanism and Results



- *Hook analysis* is flow and context-sensitive
  - Determines parameter requirements for authorization
- *Kernel analysis* uses descriptions of conceptual operations
  - Called *idioms* that describe when code implements an operation
- *Join analysis* matches
  - *Hook specs* that describe how a conceptual operation is authorized
  - *Kernel specs* that describe when conceptual operation is run
- 4 false negatives and 9 false positives compared to LSM
  - *For all file and socket hooks*

### Alpen Challenges over Tahoe



- *Identify where the application performs the conceptual operations.*
  - Key idea: Each conceptual operation involves a canonical set of events:
    - Modifying/accessing a resource attribute.
    - Calling a helper function to achieve its job.
- We call these canonical set of events *idioms*.

- *No hook implementations*
  - Policy determines conceptual operations
- *Identification of conceptual operations in code*
  - *Domain-specific metaphors* (e.g., run the conceptual operation)
  - *Results of conceptual operation met*
- *Define complete reference monitor*
  - Include security state maintenance
  - Ensure complete authorization for *security goals*
  - Optimize hook placements

### Ultimate Goals

- *Automate Generation of Security Code*
  - Memory Reuse -- scrub for secrecy
  - Key Handling -- ensure secret communication
- *Automate Generation of Filtering Interfaces*
  - Hook analysis can derive legal calls to authorization