
# Application of the Flask Architecture to the X Window System Server


Eamon Walsh

ewalsh@tycho.nsa.gov

National Security Agency

National Information Assurance Research Laboratory (NIARL)


# Overview of Talk


- X Window System overview, security issues, and proposed solution.
- Userspace object managers: how they fit into the Flask architecture and SELinux implementation.
- Description of changes to libselinux, X server.
- Examples of SELinux policy for X applications.
- Conclusion


# X Window System

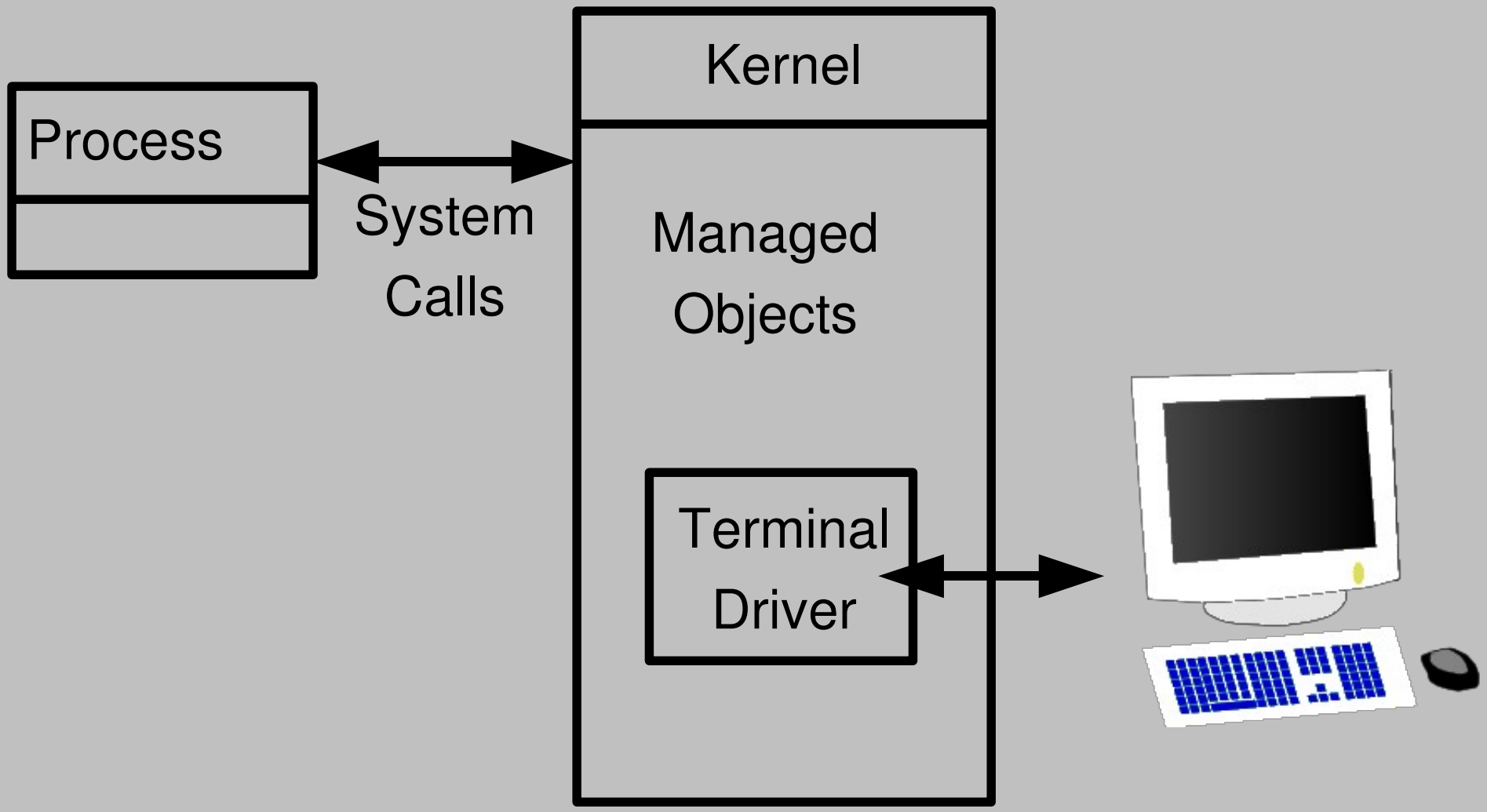

- Graphical capability provided by a server process.
- Three important functions:
  - Provides window objects and methods for drawing graphics primitives.
  - Links mouse and keyboard input events to windows.
  - Supports inter-client communications (cut & paste, drag & drop).


# X Security Problems


- Once connection to X server is allowed, SELinux has no further control over X operations.
- Objects are globally accessible: can read or draw into other windows, capture keyboard events, etc.
- Processes can exchange arbitrary data using window properties.
- Current solution is to deny connection entirely. This is too coarse-grained.

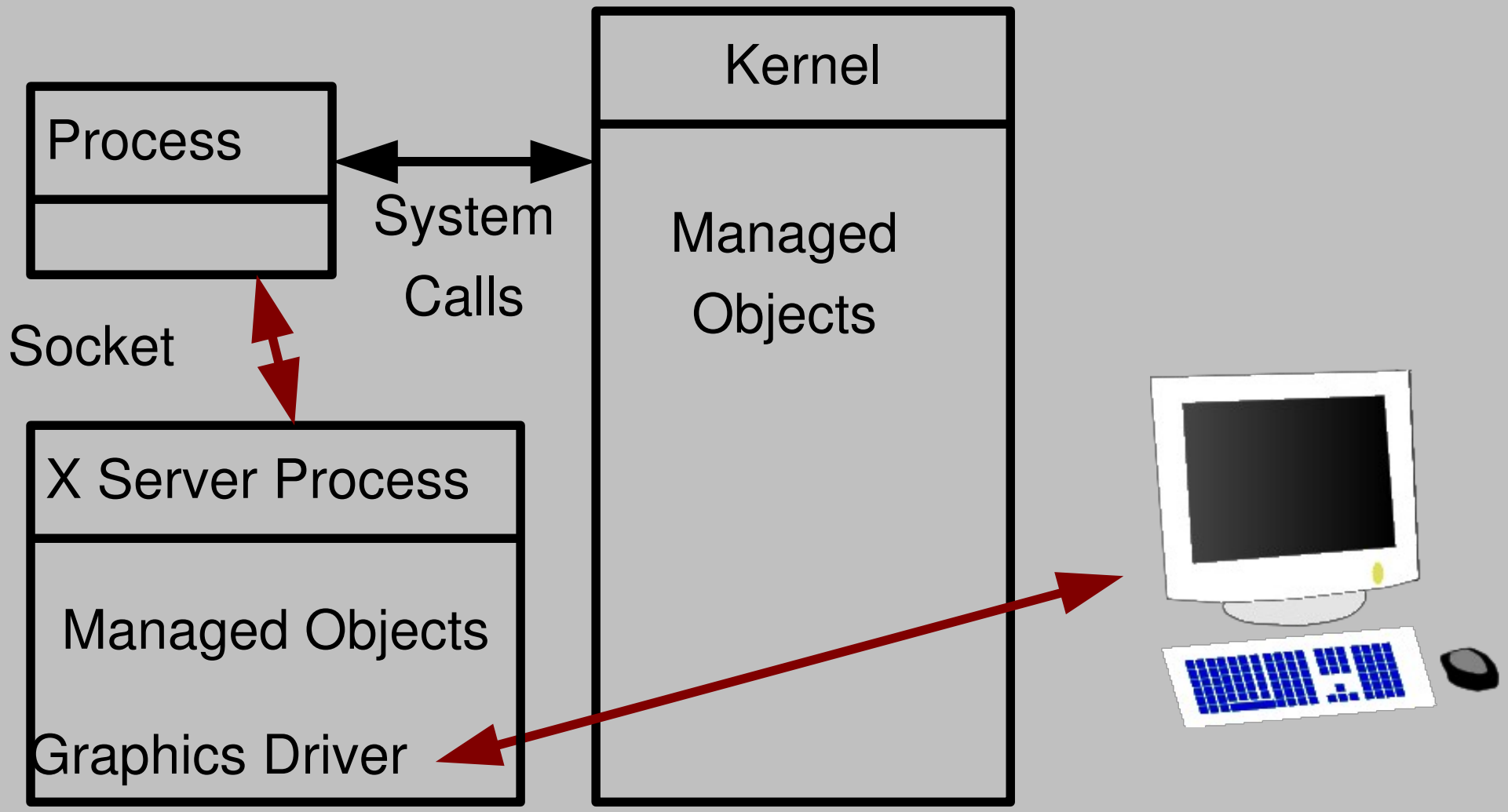

# Traditional Text Console

NATIONAL INFORMATION ASSURANCE RESEARCH LABORATORY




# X Server Model

NATIONAL INFORMATION ASSURANCE RESEARCH LABORATORY

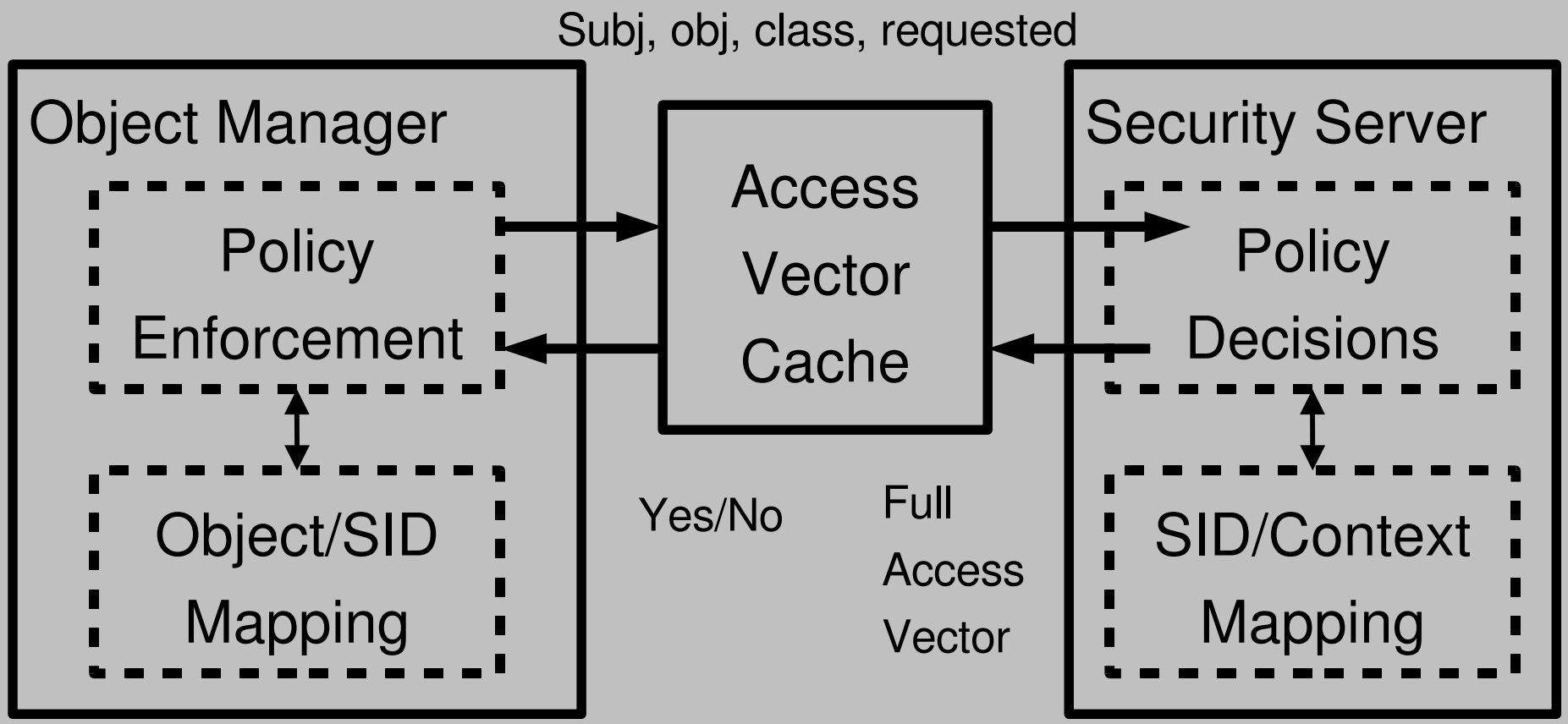



# X Security Solution


- Fine-grained control over X objects.
  - Flask object classes and permissions describing them.
  - Enforcement logic able to enforce policy decisions.
  - SELinux policy and means for querying it.
- Other security goals specific to user interface:
  - Label displayed objects so that spoofing is prevented.
  - Trusted input stream, secure attention mechanism.

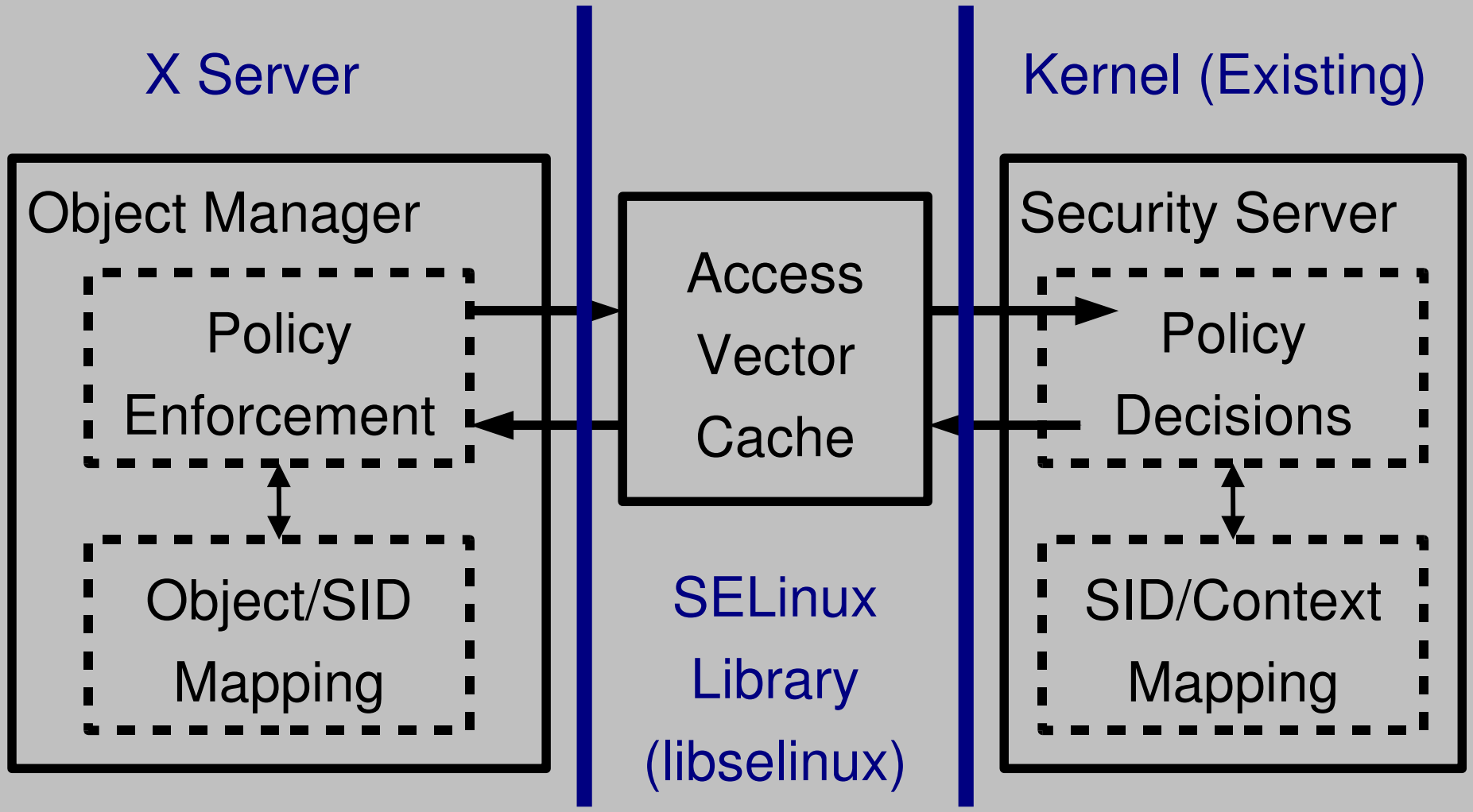

# The Flask Security Architecture

NATIONAL INFORMATION ASSURANCE RESEARCH LABORATORY



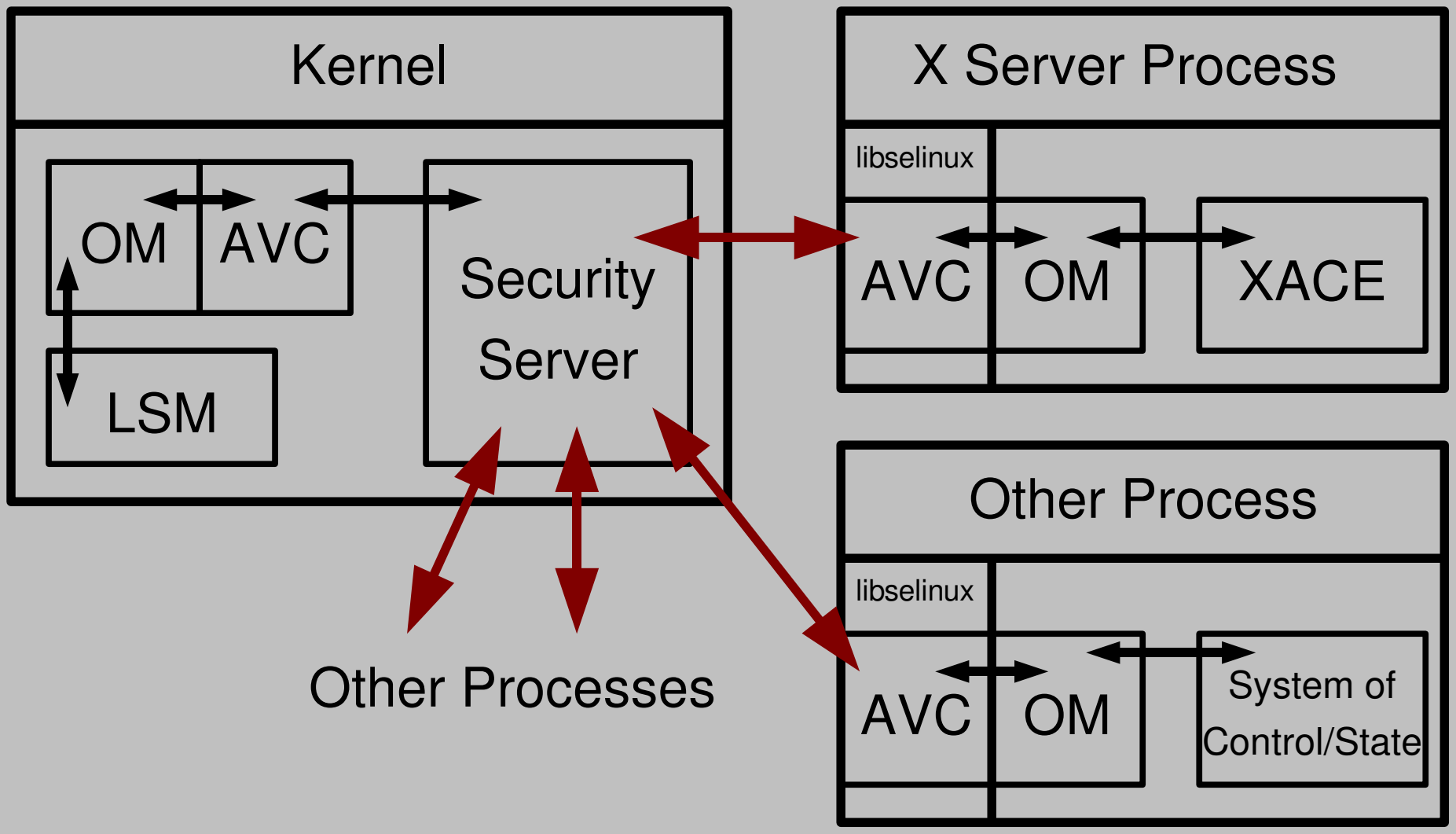

# Userspace Object Manager Concept

NATIONAL INFORMATION ASSURANCE RESEARCH LABORATORY




# Single Security Server, Multiple Object Managers

NATIONAL INFORMATION ASSURANCE RESEARCH LABORATORY




# SELinux Library Work


- Ported AVC code from kernel to libselinux.
- Uses selinuxfs to retrieve decisions from kernel security server.
- NETLINK support for asynchronous notification of policy reloads, invalidation.
- Provides `avc_has_perm` function for simple yes/no answers.


# Enforcement Logic

NATIONAL INFORMATION ASSURANCE RESEARCH LABORATORY

- Need to interrupt normal flow of execution and pass control to enforcement code.
- Chose the LSM model – generic security hooks.
- 1996 “Security” Extension, D. P. Wiggins (X Consortium) served as basis.
- Hooks allow interception of arriving protocol requests, resource lookups, and at other points.
- X Access Control Extension (XACE) accepted in X11R7.2.


# State Storage


- Need to store SID's with objects.
- Again, chose general mechanism rather than adding specific structure fields.
- DevPrivates allows driver writers to store extra data, works well for security too!
- Work is ongoing to extend this to additional server structures.


# XSELinux Flask Module


- Ties everything together.
- Uses XACE to intercept incoming requests.
- Formulates permission requests and obtains decisions using `avc_has_perm`().
- Enforces decision by returning BadAccess errors to connected client.
  - Need improved Xlib error handling to manage this.


# Policy Examples


- Allow an app to draw into its own windows:
  ```
  allow app_t self:drawable draw;
  ```
- Allow a window manager to reparent windows:
  ```
  allow wm_t app_t:window chstack;
  ```
- Allow an app to use advanced graphics:
  ```
  allow app_t
  accelgraphics_xext_t:xextension use;
  ```
- Basic strict and unconfined policies in Refpolicy.


# Window Labeling


- Make SID's available for reading by clients.
- Example: use a window property to store context of window object, and allow windowmgr to read:

```
allow wm_t app_t:seclabel_xprop_t:property read;
```

- Windowmgr can then display context to user.
- Provide X protocol allowing SELinux-aware clients to relabel their objects.


# Development Timeline


- XACE accepted to X.org server 1.2.
  - Further work on XACE/DevPrivates is in progress.
- XSELinux Flask module available on a branch.
- Target for merging to trunk is X.org server 1.4
- Whether target is met depends on speed of work and X release schedule.


# How to Get Started


- Example policy available in refpolicy.
  - 'xwindows_object_manager' tunable must be enabled.
- X Server source code available for download.
  - XACE-SELINUX branch of xserver git repository.
  - “ModularDevelopersGuide” explains how to compile.
- RPM's for Fedora, other distros?
  - Upstream package repos must track 1.4 development.


# Contact Information


- Eamon Walsh, ewalsh@tycho.nsa.gov
- The X.Org Foundation:
  http://xorg.freedesktop.org/wiki/
- Kilpatrick et. al. X analysis paper:
  http://www.nsa.gov/selinux/info/docs.cfm
- XACE Documentation:
  http://gitweb.freedesktop.org/?p=xorg/doc/xorg-docs.git;a=tree
  - Browse to sgml/security