Design and Implementation of the SELinux Policy Management Server

Joshua Brindle, Karl MacMillan, Frank Mayer, David Caplan, and Jason Tang, Tresys Technology, LLC

Policy Management

- What is policy management?
  - Includes production system tasks
  - Deployment and customization of policy
- Loadable policy modules provide foundation
  - Allows policy to be added to a running system
  - 3rd party policy now possible
- Present in soon to be released FC5
- Makes SELinux much easier to use
- But is this enough?

Policy Access Control

- Why control access to policy?
  - Currently all-or-nothing model
  - Distribute policy administration
  - Allow third party applications to install policy
- Access Control Granularity
  - Adding and removing users, roles, types
  - Adding types to roles, roles to users and so on
  - Adding and removing policy rules
  - Local Customizations such as network contexts

Policy Access Control

- Access control must be comprehensive
  - Every component of policy (e.g., types, users)
- SELinux model used to control policy access
  - Policy components become objects
  - Each with unique permissions – examples:
    - class user has add_role, create, remove
    - class type has use_src_allow, add, etc
- Policy components labeled
  - `role user_r system_u:object_r:user_role_t`
  - `user root system_u:object_r:root_user_t`
- Standard SELinux policy used to control access
- A policy hierarchy used to protect policy intent

Policy Object Model

- Policy components must be labeled
- Type enforcement rules written for the policy components

Metapolicy

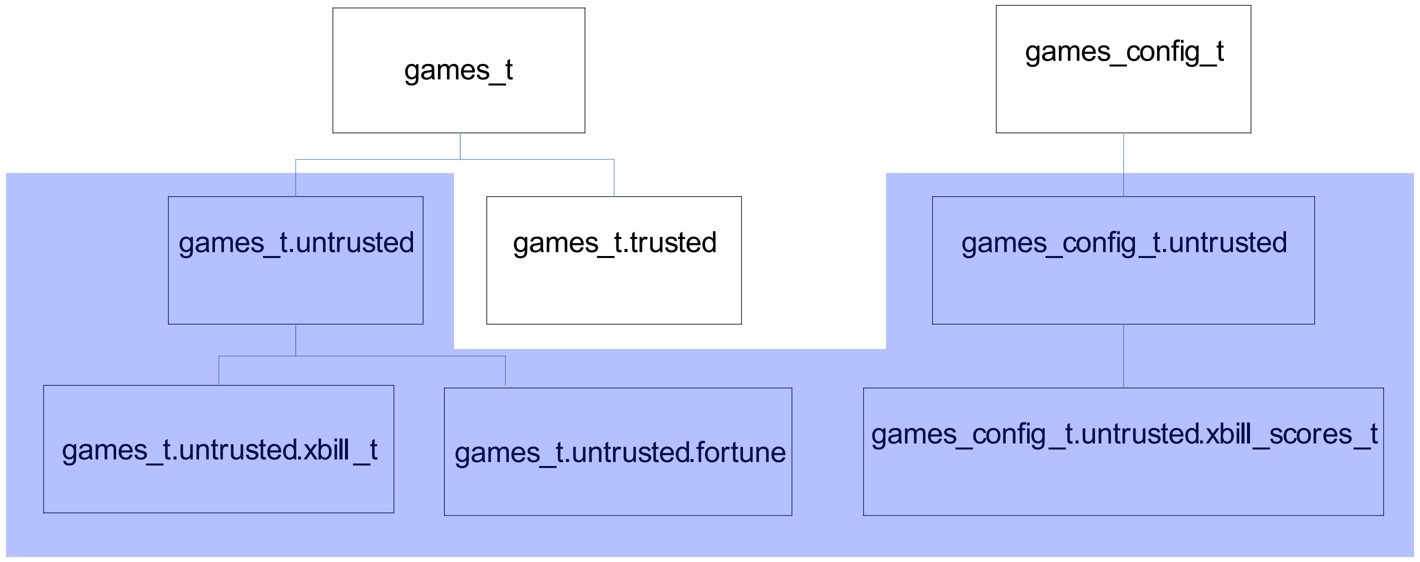
```
type games_t                      system_u:object_r:games_type_t
type games_t.untrusted            system_u:object_r:games_bad_type_t
type games_config_t.untrusted system_u:object_r:games_bad_type_t

allow rpm_unsigned_t games_bad_type_t :
                     type { use_src_allow use_tgt_allow add };
```

New policy module for xbill

```
type games_t.untrusted.xbill_t, domain;
type games_config_t.untrusted.xbill_scores_t, file_type;
allow games_t.untrusted.xbill_t
    games_config_t.untrusted.xbill_scores_t : file { read write };
allow games_t shadow_t : file { read };
```

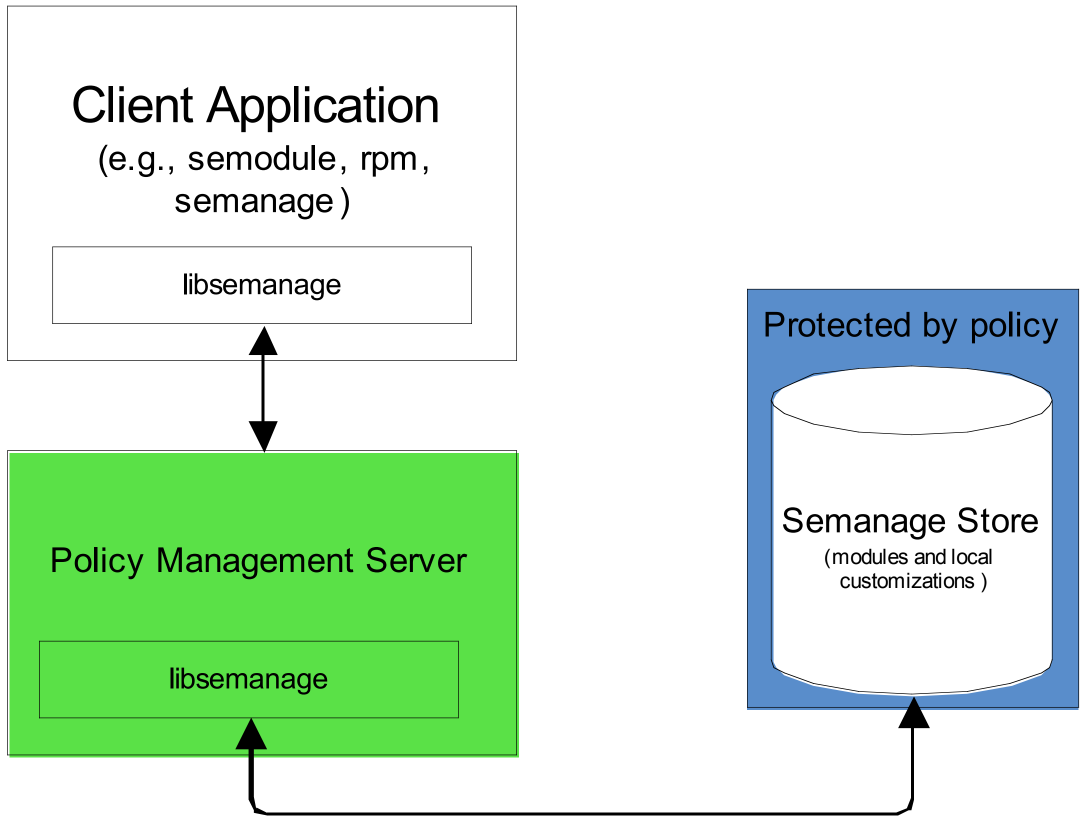
Policy Hierarchy

- The hierarchy preserves the policy intent
- Constrains which permissions can be added
- Constrains how symbols may be modified
- Allows delegation of parts of the policy


games_t
games_config_t
games_t.untrusted
games_t.trusted
games_config_t.untrusted
games_t.untrusted.xbill_t
games_t.untrusted.fortune
games_config_t.untrusted.xbill_scores_t

Policy Management Server

- Daemon that encapsulates policy



Policy Server Implementation

- PMS sits between client applications
  - Such as semanage, semodule, possibly rpm
- PMS enforces access control on policy changes
  - Using metapolicy written by the policy adminstrator
  - With objects labeled via a policy server config file
  - Enforces site specific restrictions (neverallow rules)
- PMS then writes the policy to the store
  - The store is protected by SELinux policy
  - Contains all policy components, including local customizations

Policy Management Server status

- Policy management framework
  - Lots of work in the past year
  - Now integrated into Fedora Core 5
  - Allows management of most aspects of policy
  - Implements API for all management
- Policy management server
  - Also lots of work in the past year
  - Object model partially implemented
  - Some parts of object model already upstream
  - Seamless transition to server from managed

Future Work

- Local and network settings
  - Maintaining and managing local settings
  - Enforcing network settings
- Enforcing a comprehensive network policy
- Networked policy management
  - Atomic policy change
  - Sharing modifications
  - Secure policy distribution
  - Disparate policy management

QUESTIONS?