Reference Policy

Karl MacMillan, Chris PeBenito, Frank Mayer
Tresys Technology

2006 SELinux Symposium

Motivation

- Creating SELinux policy is challenging
  - developers must be intimately familiar with
    - SELinux enforcement mechanisms
    - Linux and application implementation
  - in currently available policies
    - modules are often closely coupled
    - developers must be familiar with the entire policy
    - creating third-party modules is difficult
- Understanding policy is *more* challenging
- *This has a negative impact on security*

What is Reference Policy?

- A new SELinux policy that
  - reduces the complexity of
    - writing, maintaining, and analyzing policy
  - leverages years of community development and testing
  - uses modern software engineering principles
  - is well documented, modular, and configurable
  - provides a single source for all the policy variants
    - targeted, strict, MLS, MCS
- Together this will make a policy that is
  - maintainable
  - verifiable
  - usable

Status

- Core infrastructure and policy mature
  - in development for over a year
  - received significant community feedback
- Large number of modules available
  - ~70% of example policy modules
- Will be released as part of Fedora Core 5
  - received significant testing from rawhide
  - worked closely with Red Hat on migration
  - included regression analysis with Sediff

Security Goals

- Security is first priority of policy
  - clear security goals required for success
- Reference policy primary security goals
  - operating system self-protection
  - assurance
  - secure extensibility
  - role separation
- Other goals defined per application module

Security Goals

- Operating System self-protection
  - protect the RVM / kernel
  - resources that should be protected
    - raw devices and resources, kernel files, policy
- Assurance
  - confidence that the policy is correct and complete
  - assurance is gained through
    - extensive use of least privilege
    - limitations on error propagation
    - reduction in complexity

Security Goals

- Secure extensibility
  - provide extension via application policies
    - Refpolicy is a base for building application policies
    - potentially focusing on differing security goals
  - maintain integrity of all policies
    - protect base from applications
    - protect application from base or other applications
- Improved role separation
  - optionally remove powerful admin domains
  - allow the creation of new roles through
    - combining fine-grained role definitions
    - flexible and centralized
  - not there just yet

Functional Goals

- Refpolicy has many functional goals
  - support security goals
  - add understandability and maintainability
- Primary functional goals
  - managed complexity
  - loadable module support
  - enhanced support for tools
  - improved comprehension
  - single unified source

Functional Goals

- Managed complexity
  - reduce details exposed to policy author
  - eliminate need to be familiar with underlying policy
- Loadable module support
  - support modular and monolithic policies
    - from same source tree
  - ease creation of third-party modules
- Enhanced support for tools
  - create structures usable by tools
  - rigorously define a policy structure

Functional Goals

- Improved comprehension
  - most often cited need in policy development
  - allow policy writers to more understand policy
- Single unified source
  - multiple policy types
    - strict, targeted, MLS, MCS
    - modular, monolithic
  - multiple distributions
  - large number of configuration options

Design Concepts

- Functional goals require strong design
  - consistently applied to entire policy
- Refpolicy uses several design concepts
  - layering
  - modularity
  - encapsulation
  - abstraction
- Enforced by convention
  - future work may include validation tools

Layering

- Organizational tool, not strict layering
  - create functionality-based groupings of modules
- Lower layer modules
  - modules associated with most system policies
  - kernel - kernel resources, devices, networking
  - system - init, login, system logging
- Higher layer modules
  - modules associated with optional components/applications
  - administration - log tools, RPM, su
  - services - Apache, BIND, DHCP
  - applications - gpg, Mozilla, Webalizer

Reference Policy Modules

- Primary organizational tool
  - smallest policy component
  - encapsulation based on modules
- Reference policy modules have 3 files – e.g.,
  - bind.te – private types, attributes, and rules
  - bind.if – public module interfaces
  - bind.fc – labeling statements
- Types / attributes private to modules
  - no more global types / attributes
  - interfaces allow controlled inter-module access

Modularity & Encapsulation

```
policy_module(bind,1.1.0)
```

Module declaration

```
type named_t;
type named_exec_t;
init_daemon_domain(named_t,named_exec_t)
type named_cache_t;
files_type(named_cache_t)
type named_conf_t;
files_type(named_conf_t)
type named_zone_t;
files_type(named_zone_t)
```

Private Type Declarations

- includes interface calls

```
allow named_t named_cache_t:file manage_file_perms;
allow named_t named_conf_t:file r_file_perms;
allow named_t named_zone_t:file r_file_perms;
```

Private access

```
kernel_read_system_state(named_t)
kernel_read_network_state(named_t)
corenet_non_ipsec_sendrecv(named_t)
corenet_udp_sendrecv_generic_if(named_t)
corenet_udp_sendrecv_generic_nodes(named_t)
corenet_udp_sendrecv_all_ports(named_t)
corenet_udp_bind_all_nodes(named_t)
corenet_udp_bind_dns_port(named_t)
logging_send_syslog_msg(named_t)
```

Interface calls

- allows access to other module's resources

Abstraction

```
interface(`logging_send_syslog_msg',`
        gen_require(`
                type syslogd_t, devlog_t;
        ')

        allow $1 devlog_t:lnk_file read;
        allow $1 devlog_t:sock_file rw_file_perms;

        # the type of socket depends on the syslog
daemon

        allow $1 syslogd_t:unix_dgram_socket sendto;
        allow $1 syslogd_t:unix_stream_socket connectto;
')
```

Require block for modules

Permissions granted via the interface

QUESTIONS?