Red Hat - SELinux

Deepak Bansal
Sr. Technical Consultant

Agenda

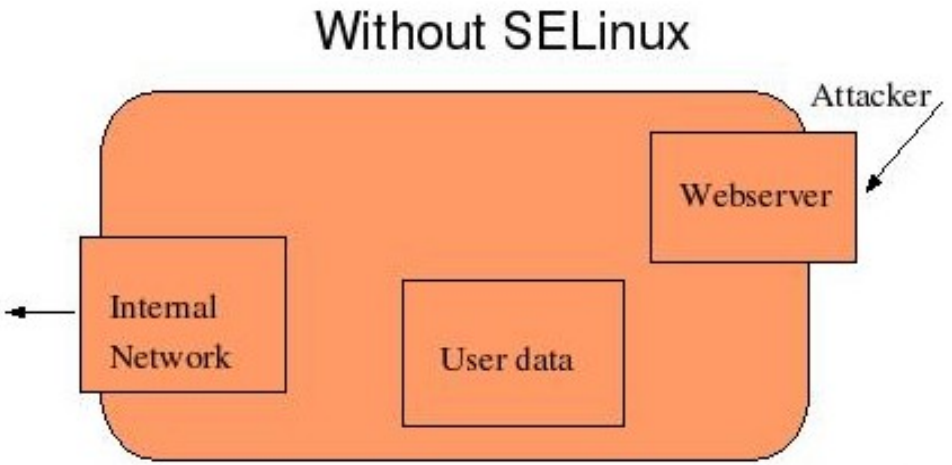
- Why do we need SELinux? What are the principal concepts?
- SELinux Details
  - How does it work?
  - Security Architecture
  - Domain/Type
  - What are the available policies?
  - What's a policy actually made of?
- Usage
  - Admin/User perspective

Why Do We Need SELinux?

- Discretionary Access Control (DAC) is the standard mechanism for Linux security.
- Under DAC, all processes execute with an correlated user and group. That process has access to all files and directories that the user and group can access. Thus an errant process could destroy all files that belong to the user!
- **For Example:** A user can expose a file or directory to a security or confidentiality breach with a misconfigured **chmod** command and an unanticipated propagation of access rights.
- A process started by that user, such as a CGI script, can do anything it wants to the files owned by the user. A compromised Apache HTTP server can perform any operation on files in the apache group. Malicious or broken software can have root-level access to the entire system, either by running as a root process or using setuid or setgid.

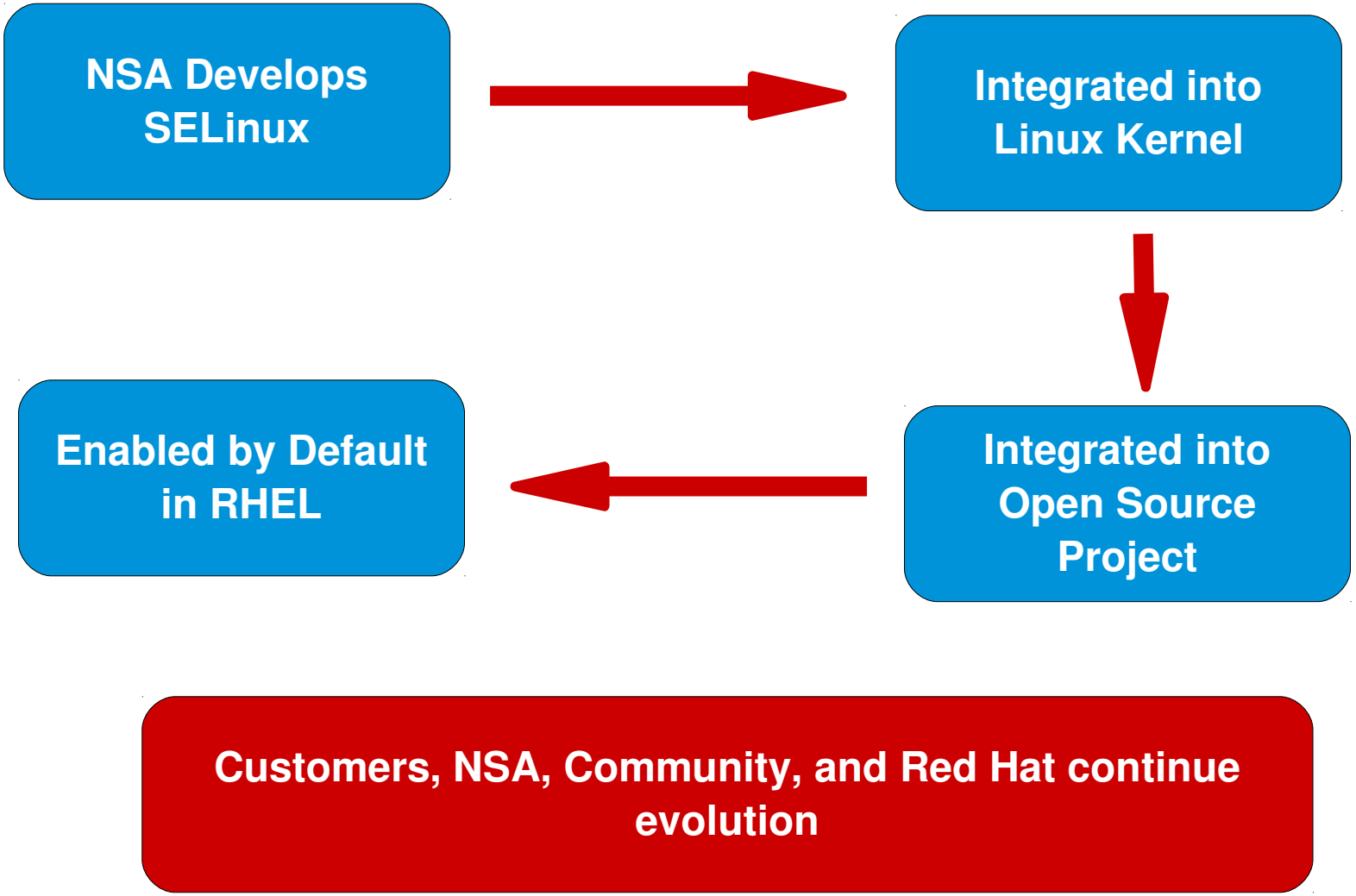
Traditional DAC System

- Without SELinux, an attacker who gained access through a security hole in a public available server, like the web server, would have more broad access to the system.



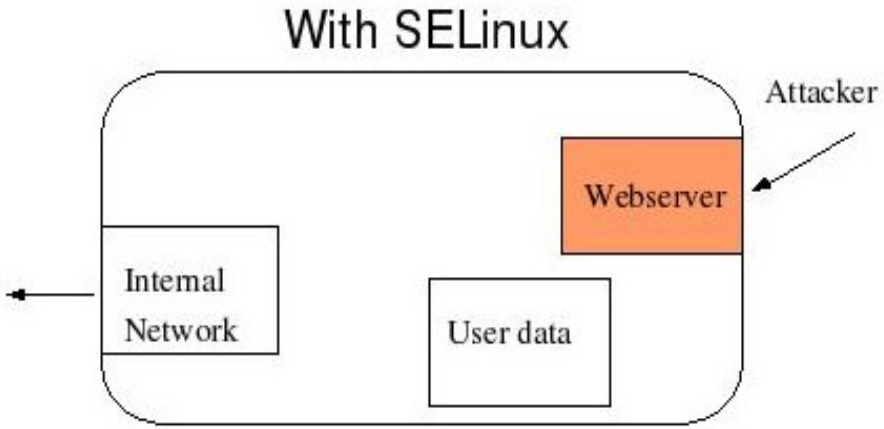
- The attacker would then have a platform from which to exploit other vulnerabilities on the system, perhaps gaining root access, and the attacker would be able to launch attacks on other systems within the internal network.

SELinux: History



With SELinux Enabled

- With SELinux enabled, an attacker who exploited a security hole in the web server, would only have access to the files that the web server normally has access to.



- The Mandatory Access Control allows the web server only to access files of a certain type. Data from other confined processes are not accessible.

How does it work?

SELinux is all about labeling (Security Context)

- Processes gets labels

  Virtual machines are processes!!
- Files/Devices Gets Labels

  Virtual images are stored on files/devices!!!!
- Rules govern how Process Labels interact with Process/Files Labels.
- Kernel Enforces these Rules.

What SELINUX Can Do

- Confine programs to minimum privilege required
- Protect against exploits
- Prevent system access to users' private information

The SELinux kernel enforces mandatory access control policies that confine programs, particularly system services, to the minimum amount of privilege they require to do their jobs successfully.

What SELINUX Can Do

SELinux implements role-based access control and type enforcement. The main security mechanism implemented in SELinux can protect against exploiting vulnerabilities in software products:

- Privilege escalation
- unauthorized reading and/or modification of data and programs
- logging of security breaches
- fine granulated access control implementation
- offers type enforcement
- offers role based access control

Collectively, these features limit the damage that can be done by a system service, particularly a network service, to user data.

What Selinux Cannot Do

- Cannot perform code audits
- No encryption of data between nodes
- Services still need to be updated

SELinux is not a monolithic security guarantor. It cannot solve all security problems that can occur on a system. As an element in the security infrastructure, it can offer a considerable benefit, but it is no replacement for good coding and for firewalls and service protections.

RPMs still need to be updated with the latest security and bug fixes. It is best to keep as many security exploits closed as possible.

Security Architecture

Security Context

- User Identity
- Role
- Domain / Type
- Sensitivity
- Category

Security Policy

Security Architecture

- Every object (files and other items) and every subject (process) has a security context. This security context has three attributes: a user identity, a role, and a type. Collectively, these attributes limit the authority of the subject over the object. Typically, the security context is displayed as a colon-delimited triplet in this format:
- user_identity:role:type

  For example, the following security context is set for web server content:
- system_u:object_r:httpd_sys_content_t
- Security context of subject & object passed to SELinux
- Kernel/SELinux check & verify access
  - Grant access. Record allowance in AVC (Access Vector Cache)
  - Deny access, log error

Role Based Access Control (RBAC)

- “root” really isn't “root”
  - root_u:WebServerAdmin_r:SysAdmin_t
  - root_u:OracleDBAdmin_r:SysAdmin_t
- Each file/process has a context
  - user:role:type:sensitivity:category
  - Provides for multiple layers of protection
  - Most systems haven't implemented sensitivity or category
- Roles defined for various processes
- Permissions assigned to roles rather than individual users

User identity and role

- SELinux user account correlated with object, typically ends in _u, as in the_system_u identity in the previous slide
- Role defines which SELinux user identities are permitted in which domains, typically ends in _r as in object_r in the previous slide
- Role can be changed using newrole
- Role has access to types or domains

Domain / Type

- Processes (subjects) execute in domains
- Resources (objects) are correlated with a type
- Subjects and objects have a domain or a type, ending in _t as in the httpd_config_t and etc_t types in the earlier slide
- Webserver Example:
  - Apache executable /usr/sbin/httpd has a type httpd_exec_t

    system_u:object_r:httpd_exec_t
  - Apache process httpd is running in domain httpd_t

Domain / Type

- Processes (subjects) under SELinux are executed in sandboxes called domains.
- Resources (objects) live in protected domains called types. Types are the workhorse security attribute: an SELinux policy usually defines only a handful of users and roles, but often hundreds or thousands of types. This policy defines which domains have access to which types.
- Normally subjects and objects which are associated with a domain or a type typically end with an _t.
- **For example,** the web server binary has a type called httpd_exec_t. The web server process, belongs to a domain called httpd_t. The web server data is of the type httpd_sys_content_t. The policy permits subjects in the domain httpd_t to access files with the type httpd_sys_content_t. If it is a different type, the web server will get an AVC error, and will not be permitted access to those objects.
- **For example**, the binary executable file object at /usr/bin/postgres has the type postgresql_exec_t. All of the targeted daemons have their own *_exec_t type for their executable applications. postgresql_exec_t transitions to the postgresql_t domain, upon execution.

Sensitivities and Categories

- Used in the **Strict** and **MLS** policies
- Sensitivity and category are hierarchical
- Category is optional
- No read up, no write down

SELinux Policy

- A policy is a set of rules that guides the SELinux security engine.
- It defines types for file objects and domains for processes, uses roles to limit the domains that can be entered, and has user identities to specify the roles that can be attained.
- There are 3 SELinux Policy Types
  - Targeted
  - Strict
  - MLS

Targeted Policy

- Default Policy in RHEL5. Supported by GSS
- Targets specific applications/daemons to lock down
- Allows all other applications to run in the unconfined domain (unconfined_t)
- Applications running in the unconfined domain runs as if SELinux were disabled

Strict Policy

- Denies access to everything by default
- Complete protection for all process on the system
- Requires that policies be written for **all** applications, often requires customization
- Strict is type enforcement with added types for users (e.g. user_t and user_firefox_t).
- Not enabled/installed by Red Hat as default
  - yum install selinux-policy-strict

Multi-Level Security (MLS)

- Focuses on confidentiality (i.e. Separation of multiple classifications of data)
- Ability to manage {process, users} with varying levels of access {the need to know}
- Uses category & sensitivity levels
- Not installed by default
  - yum install selinux-policy-mls

Policy Booleans

- Allow runtime modification of the security policy without having to load a new policy
- The policy defines a default value for each boolean typically false
- Use getsebool and setsebool to manage the booleans
- setsebool -P will recompile the policy with the changes
- Keeps a "reminder" in /etc/selinux/targeted/modules/active/booleans.local

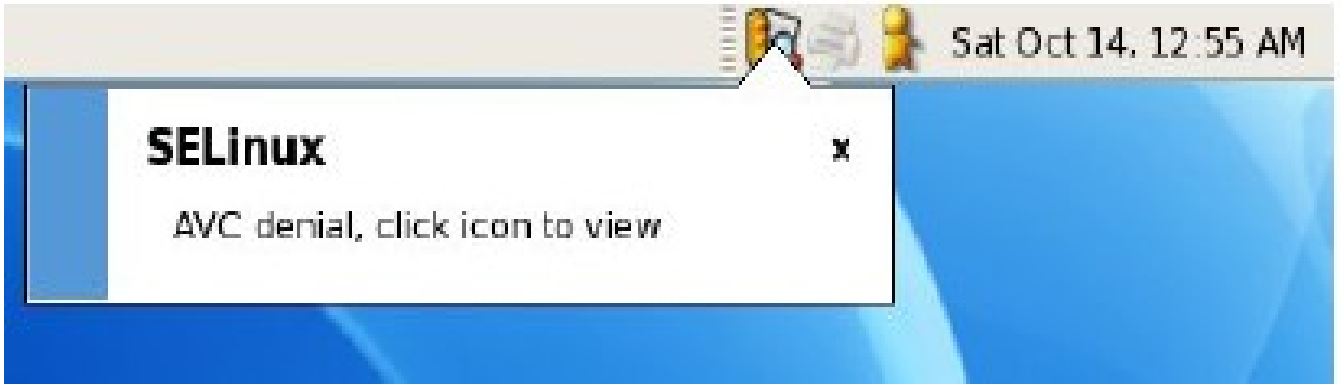
Security Context Information

- To view the security context information, correlated with files you can use the ls command with the -Z option:
- [root@stationX ~]# ls -Z
- -rw-r--r-- root root system_u:object_r:boot_t   grub
-
- [root@stationX ~]# id -Z
- root:system_r:unconfined_t:SystemLow-SystemHigh
- To view the security context information correlated with processes, you can use ps with the -Z option:
- [root@stationX ~]# ps -ZC httpd
- system_u :system_r:httpd_t   2901 ?   00:00:00 httpd
- [root@stationX ~]# mkdir -Z system_u:object_r:tmp_t dir
-

SELinux Usage
GUI & Console

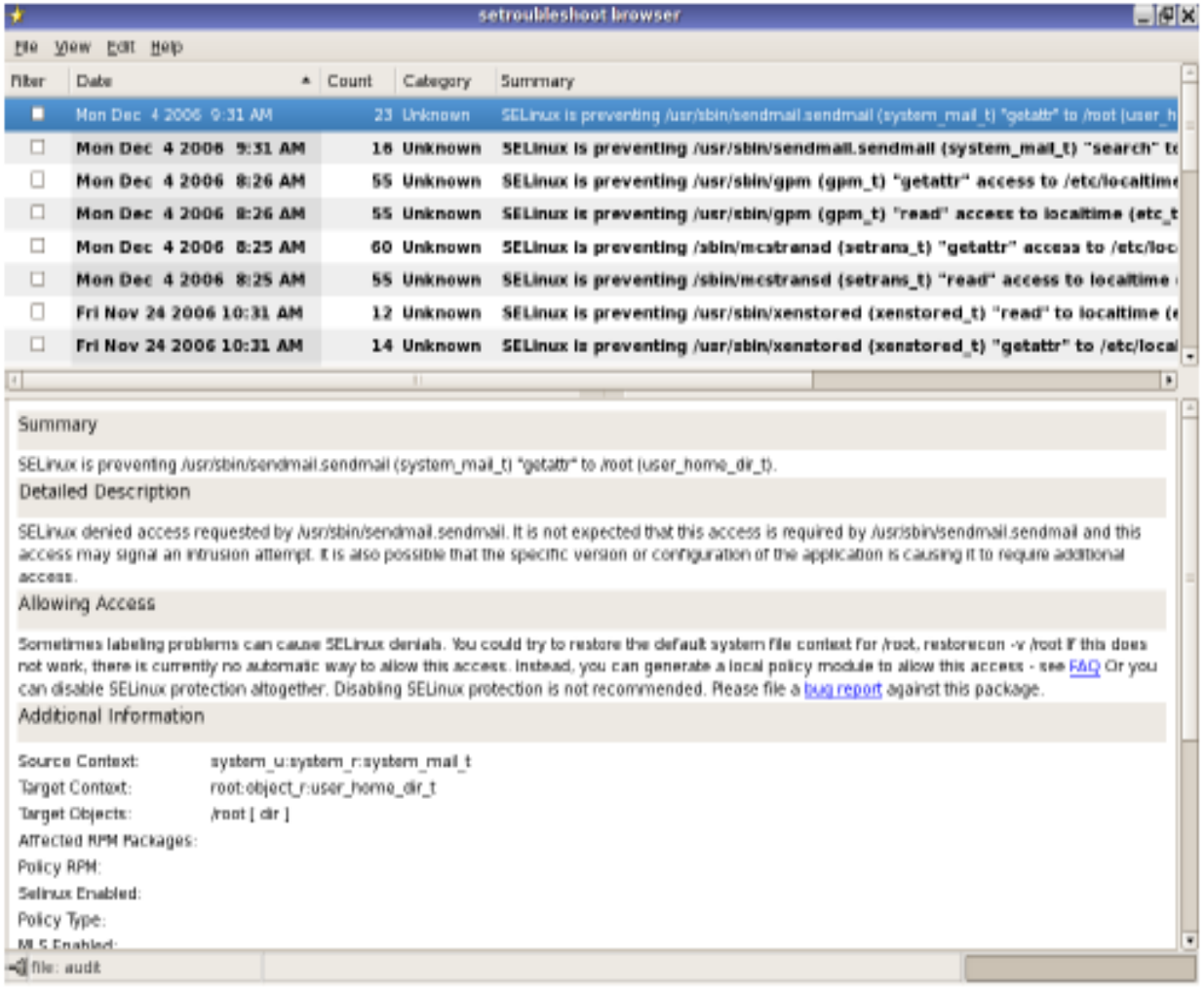
End-User Perspecitve

- sealert Notifications



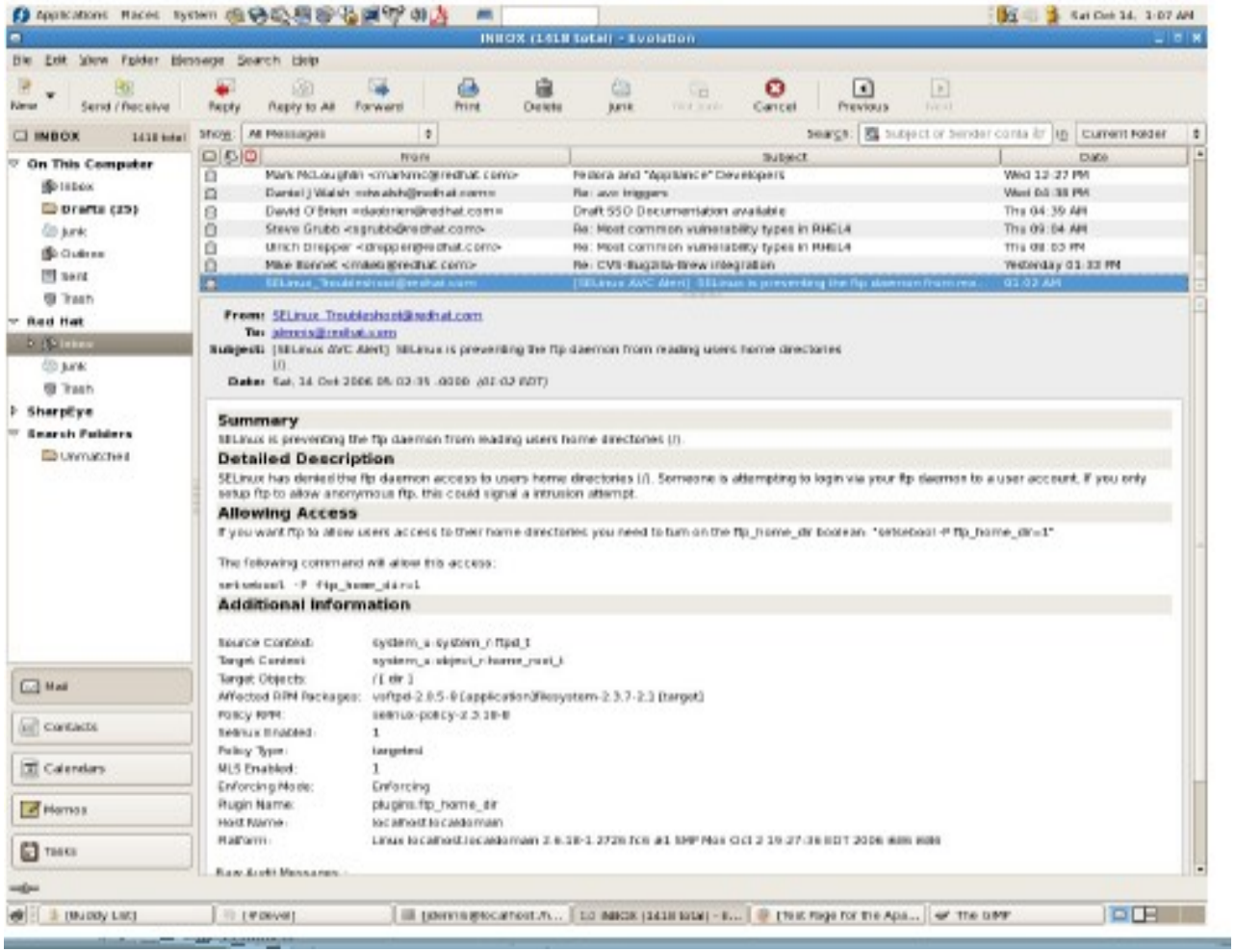
End-User Perspecitve

- sealert Browswer



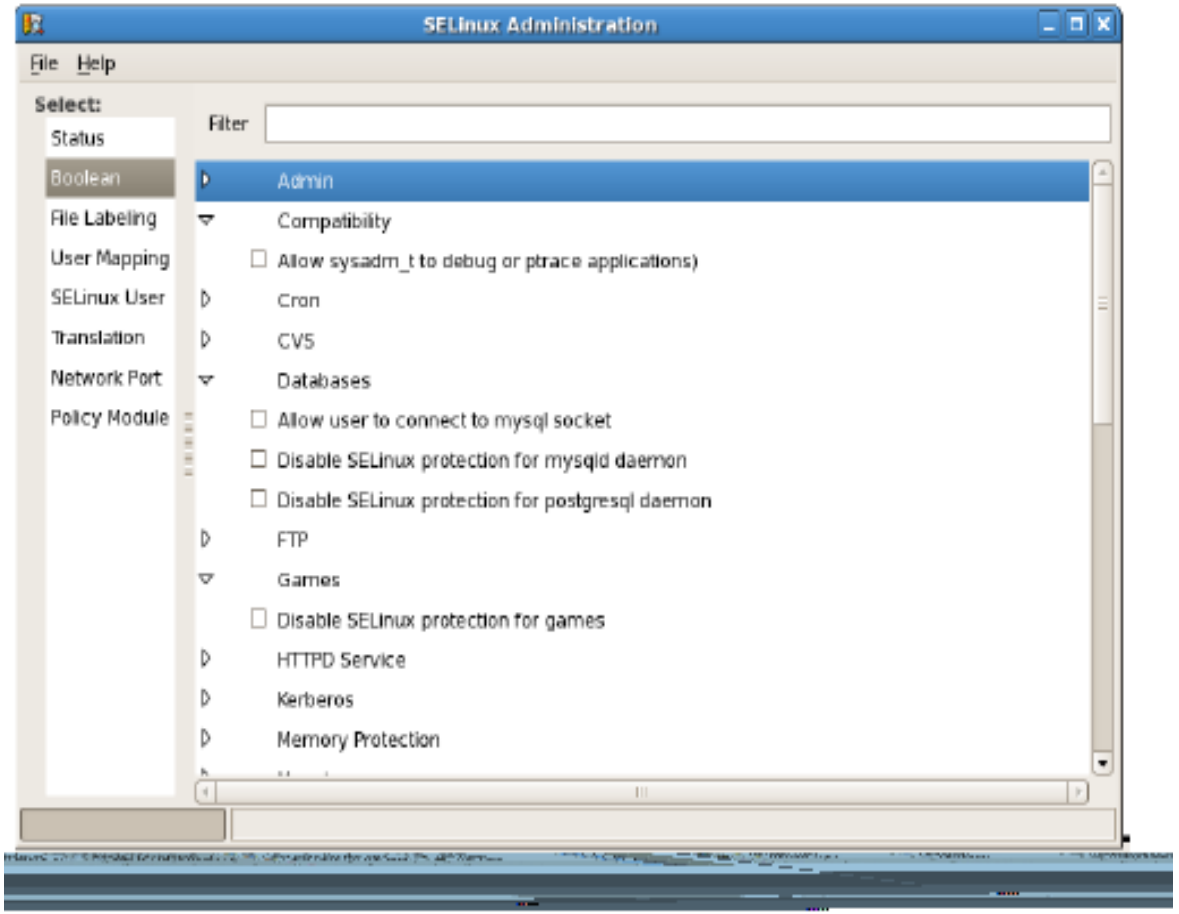
System Administrator Perspective

- sealert + Email notifications



System Administrator Perspective

- system-config-selinux



Questions?