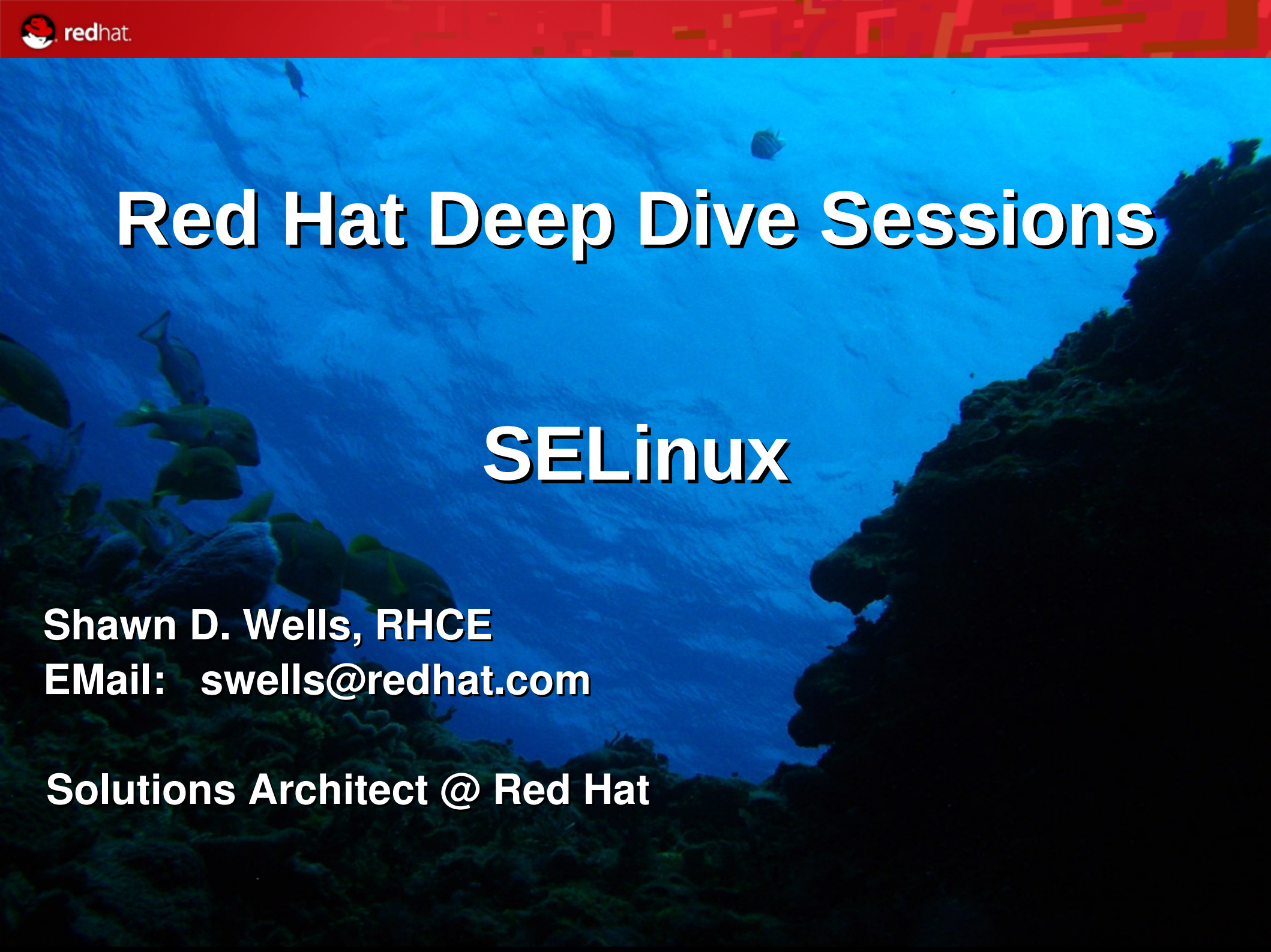

redhat.
Red Hat Deep Dive Sessions
SELinux
Shawn D. Wells, RHCE
EMail: swells@redhat.com
Solutions Architect @ Red Hat

Agenda

1) **Why do we need SELinux? What are the principal concepts?**
2) **SELinux Details**
   - **Type Enforcement**
   - **What are the available policies?**
   - **What's a policy actually made of?**
   - **How do I {add, change} a policy?**
   - **What's the associated overhead?**
3) **Usage**
   - **User Perspective**
   - **Admin Perspective**
4) **Scenarios**
   - Fixing the RHT Corporate VPN “update”

redhat.

Why do we need SELinux?

Linux Access Control Problems

1) **Access is based off users' access**

Example: Firefox can read SSH keys

```
# ps -x | grep firefox
shawn 21375 1 35 11:38 ? 00:00:01 firefox-bin
```

```
# ls -l id_rsa
  -rw------- 1 shawn shawn 1743 2008-08-10 id_rsa
```

Fundamental Problem: Security properties not specific enough. Kernel can't distinguish applications from users.

Linux Access Control Problems

2) Processes can change security properties

Example: Mail files are readable only by me..... but Thunderbird could make them world readable

Fundamental Problems:

- Standard access control is discretionary
- Includes concept of “resource ownership”
- Processes can escape security policy

Linux Access Control Problems

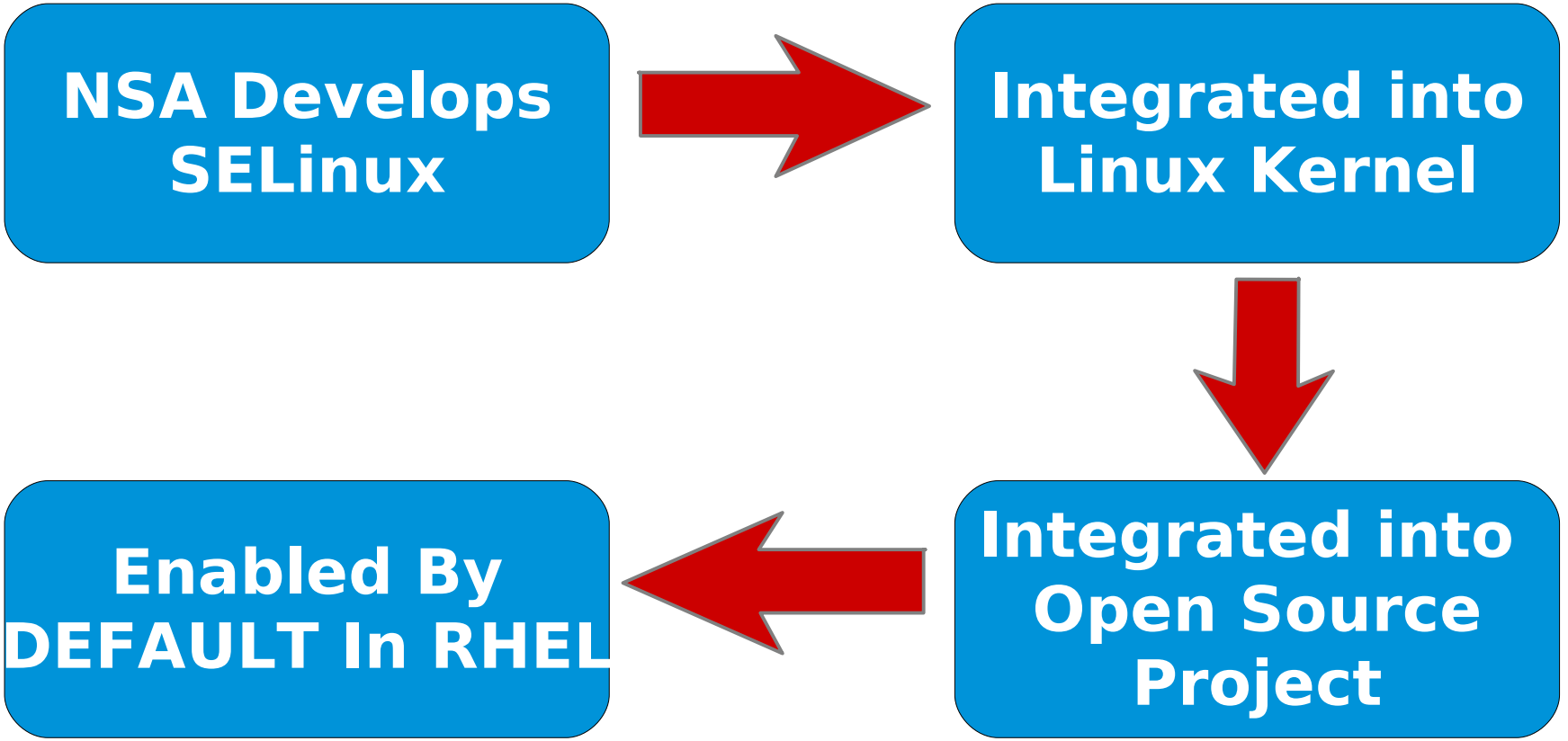
3) Only two privilege levels: User & root

Example: Apache gets hacked, allowing remote access to root. Entire system is compromised.

Fundamental Problems:

- Simplistic security policy
- No way to enforce least-privilege

SELinux: Building Security Openly



Customers, NSA, Community, and Red Hat continue evolution

Red Hat Security Certifications

NIAP/Common Criteria: The most evaluated operating system platform

- Red Hat Enterprise Linux 2.1 – EAL 2 (Completed: February 2004)
- Red Hat Enterprise Linux 3 EAL 3+/CAPP (Completed: August 2004)
- Red Hat Enterprise Linux 4 EAL 4+/CAPP (Completed: February 2006)
- Red Hat Enterprise Linux 5 EAL4+/CAPP/LSPP/RBAC (Completed: June 2007)

DII-COE

- Red Hat Enterprise Linux 3 (Self-Certification Completed: October 2004)
- Red Hat Enterprise Linux: First Linux platform certified by DISA

DCID 6/3

- Currently PL3/PL4: ask about kickstarts.
- Often a component in PL5 systems

DISA SRRs / STIGs

- Ask about kickstarts.

FIPS 140-2

- Red Hat / NSS Cryptography Libraries certified Level 2

Security Standards Work

Extensible Configuration Checklist Description Format (XCCDF)

- Enumeration for configuration requirements
- DISA FSO committed to deploying STIG as XCCDF
- Others working with NIST
- Security policy becomes one file

Open Vulnerability & Assessment Language (OVAL)

- Machine-readable versions of security advisories

Common Vulnerability and Exposures (CVE) Compatibility

- Trace a vulnerability through multiple vendors

How's it work?

Linux Access Control Introduction

Linux access control involves the kernel controling

- **Processes** (running programs), which try to access...
  - **Resources** (files, directories, sockets, etc)

For example:

- Apache (process) can read web files
- But **not** the /etc/shadow file (resource)

Traditional methods do not clearly separate the privileges of users and applications acting on the users behalf, increasing the damage that can be caused by application exploits.

So, how should these decisions be made?

Security Architecture

Every subject (i.e process) and object (i.e. data files) are assigned collections of security attributes, called a **security context**

1) Security context of subject & object passed to SELinux

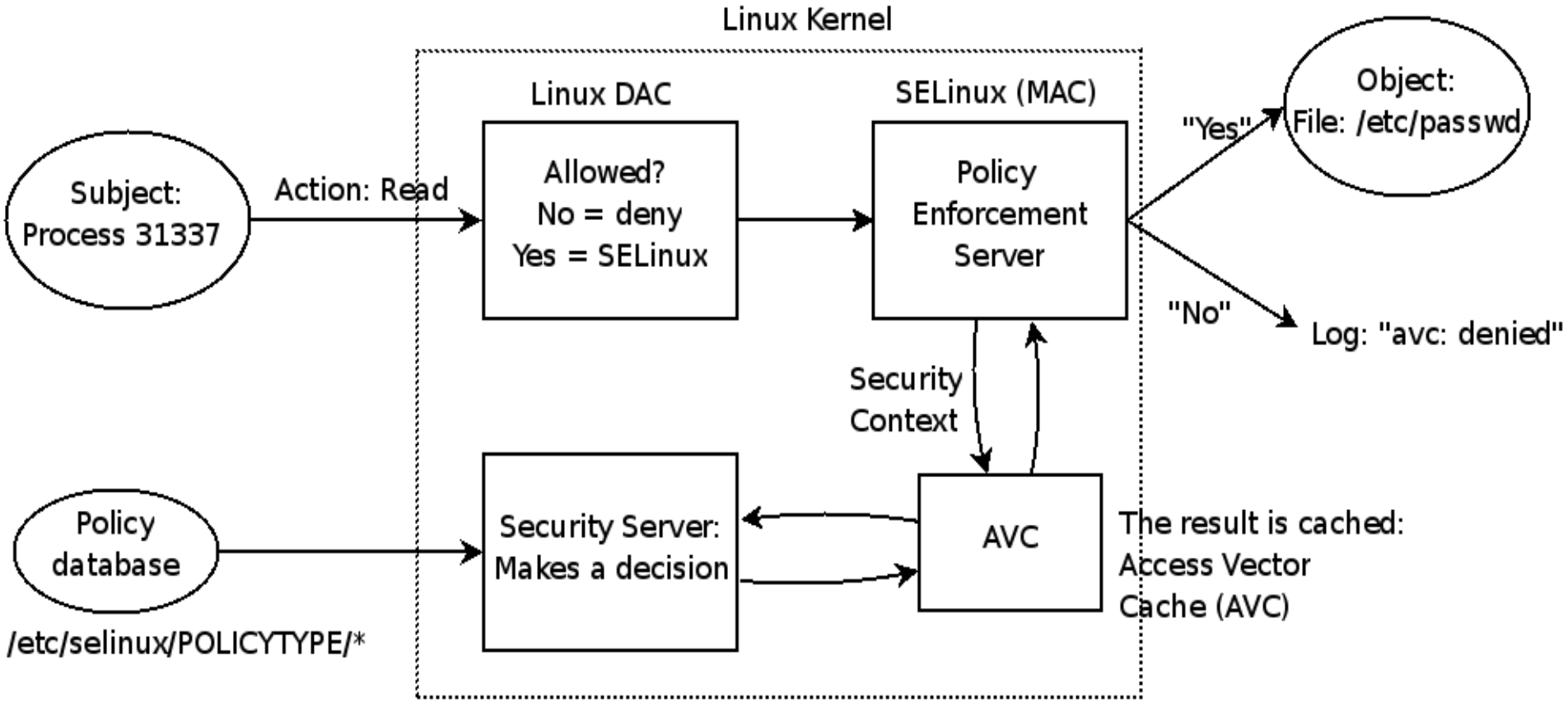
2) Kernel/SELinux check, verify access

2a) Grant access. Record allowance in AVC (Access Vector Cache)

2b) Deny access, log error

Security Architecture

Or in picture view...



Role Based Access Control (RBAC)

“root” really isn't “root”

i.e:

root_u:**WebServerAdmin_r**:SysAdmin_t

root_u:**OracleDBAdmin_r**:SysAdmin_t

SELinux Details

Type Enforcement

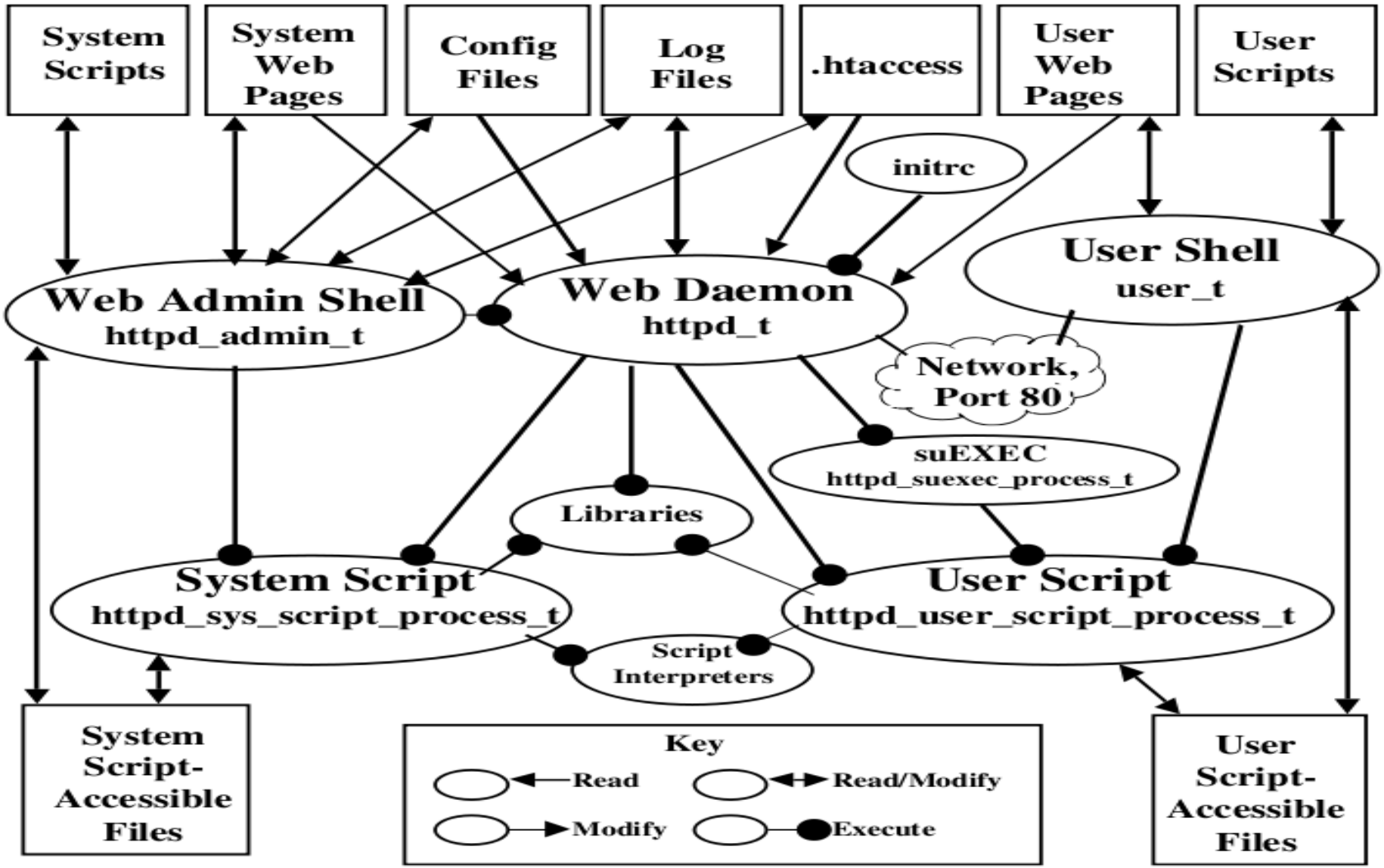
- SELinux implements the MAC model through type enforcement.
- In RHEL5, SELinux also provides RBAC and Bell-LaPadula (MLS), but it uses type enforcement to implement them.
- Type Enforcement involves defining a type for every subject, that is, process, and object on the system.
- Permissions are checked between the source type and the target type for each access.
- Objects include (but are not limited to):
  - Network Sockets
  - Shared Memory Segments
  - Files
  - Processes
  - etc.

SELinux Contexts

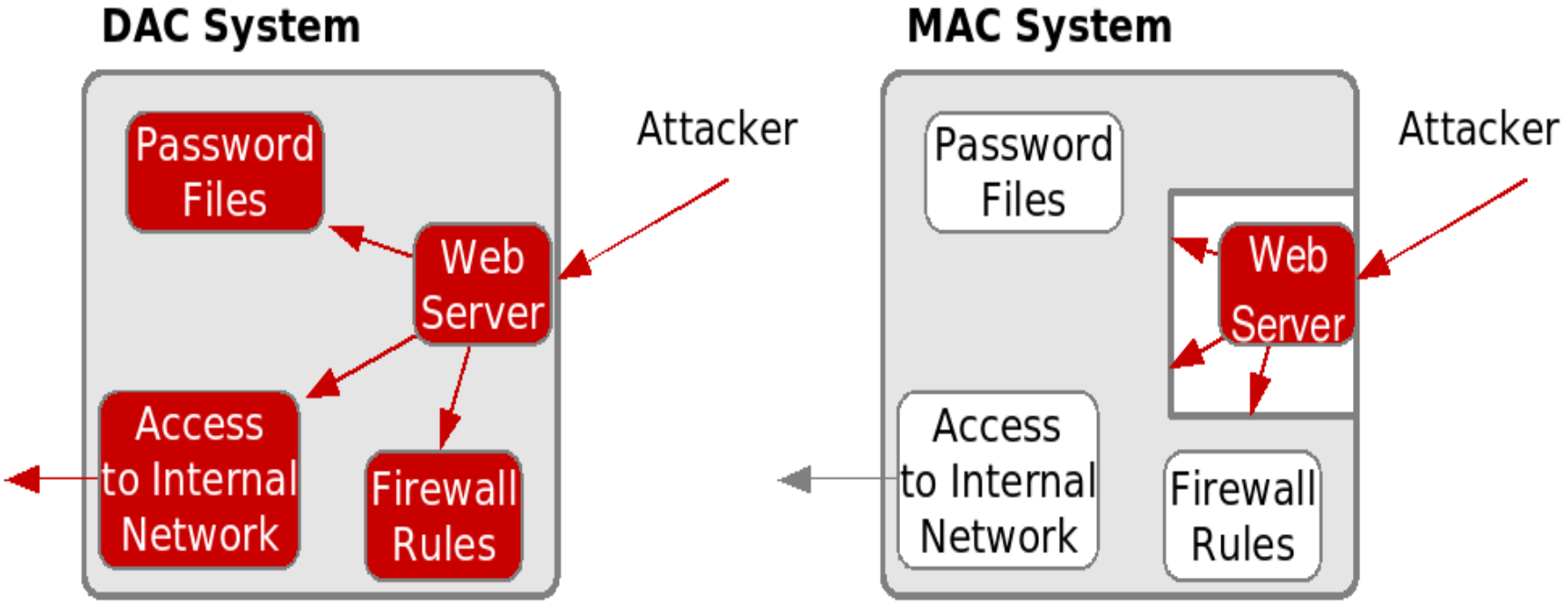
root:object r:sysadm home t:s0:c0

- The above is an SELinux context
- user_t
- role_t
- file_t
- Sensitivity
- category

SELinux Contexts



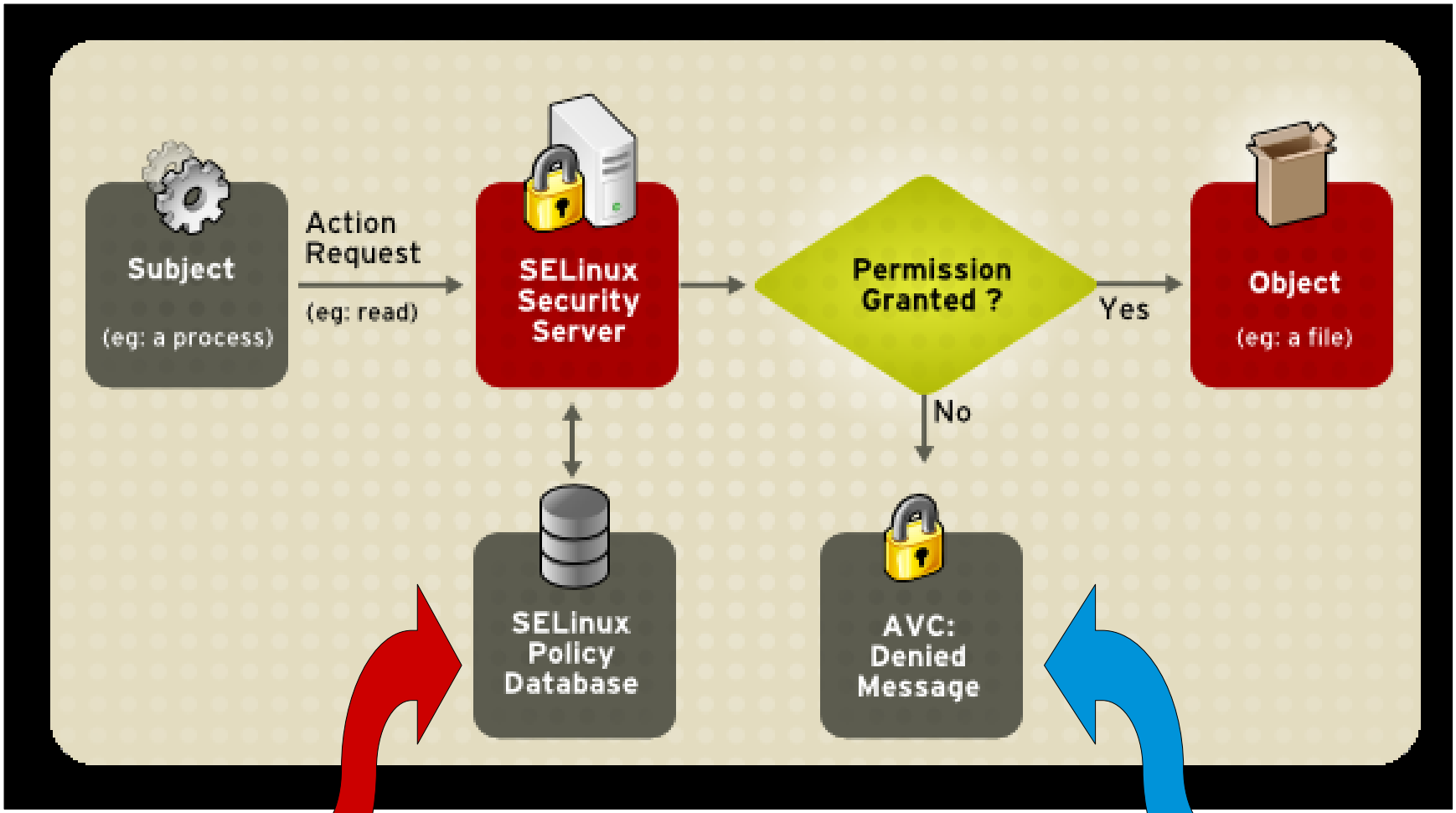
DAC vs MAC



- Application can change attributes
- User privileges
  =
  Process privileges

- Orthogonal to DAC
- Roles, Contexts, Types

How does SELinux Work?



/etc/selinux/{targeted,strict}/policy

/var/log/messages

SELinux Policy

- Policies are matrices of statements which tell SELinux if certain actions are allowed based on the context of the objects attempting those actions.

- There are three SELinux Policy Types

The Three SELinux Policy Types

1) **Targeted Policy**

- *Default policy in RHEL5. Supported by HelpDesk.*
- Targets specific applications to lock down.
- Allows all other applications to run in the unconfined domain (`unconfined_t`)
- Applications running in the unconfined domain run as if SELinux were disabled

The Three SELinux Policy Types

2) Strict Policy

- Denies access to everything by default
- Complete protection for all processes 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 by Red Hat as default

The Three SELinux Policy Types

3) Multi-Level Security (MLS)

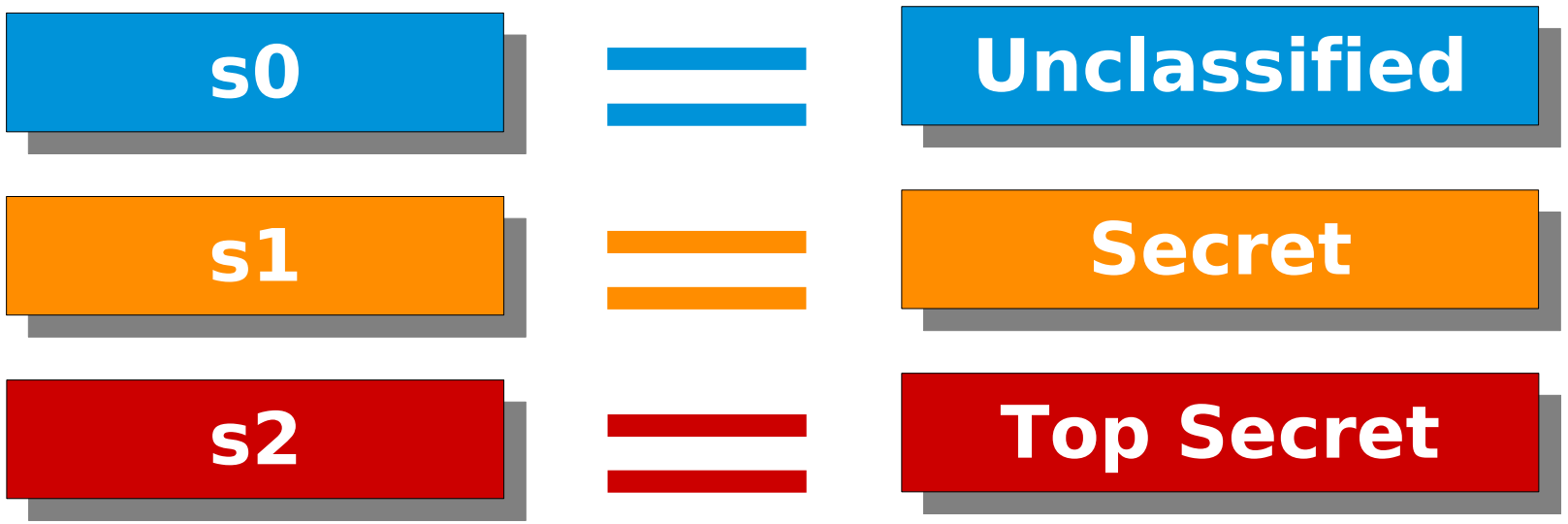
- Focuses on confidentiality (i.e. separation of multiple classifications of data)
- Ability to manage {processes, users} with varying levels of access. (i.e. "*the need to know*")
- Uses category & sensitivity levels

The Three SELinux Policy Types

3) Multi-Level Security (MLS)

(a) Sensitivity Labels

- Mostly used by the government – Top Secret, Secret, Unclassified, etc

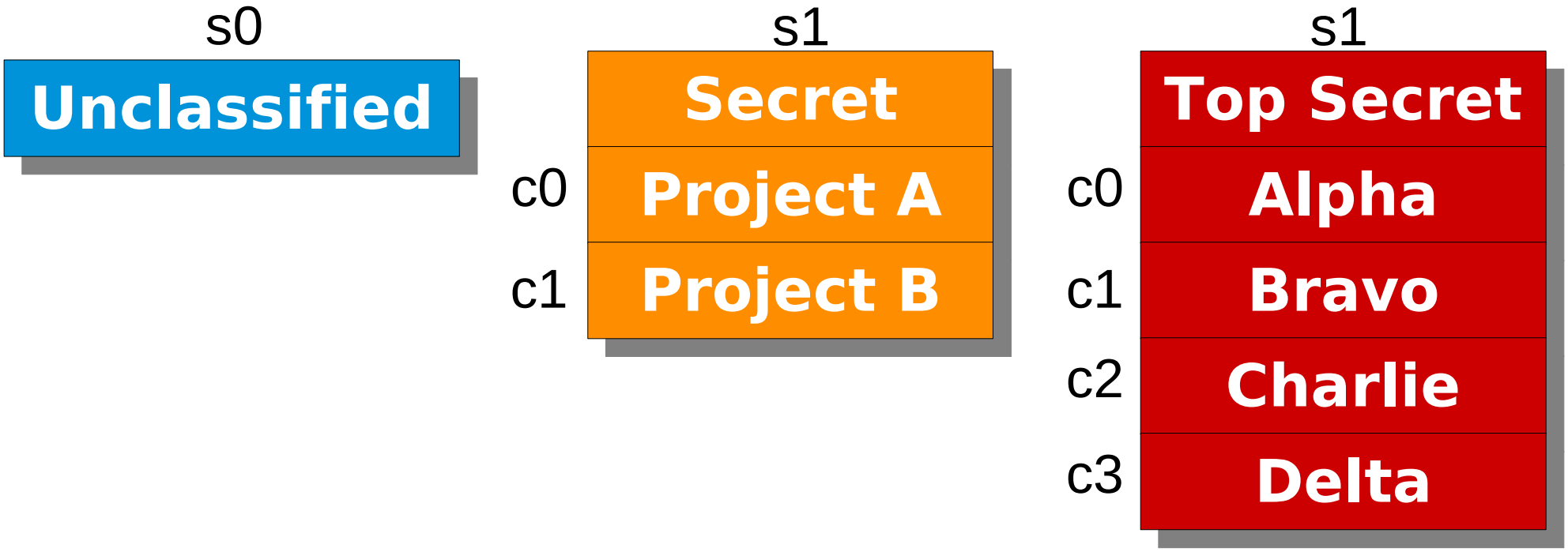


The Three SELinux Policy Types

3) Multi-Level Security (MLS)

(b) Category Labels

- Separation of data types, compartments, projects, etc



The Three SELinux Policy Types

3) Multi-Level Security (MLS)

(b) Polyinstantiation & pam_namespace

- The pam_namespace PAM module sets up a private namespace for a session with polyinstantiated directories
- A polyinstantiated directory provides a different instance of itself based on user name, or when using SELinux, user name, security context or both

The Three SELinux Policy Types

3) Multi-Level Security (MLS)

(b) Polyinstantiation & pam_namespace

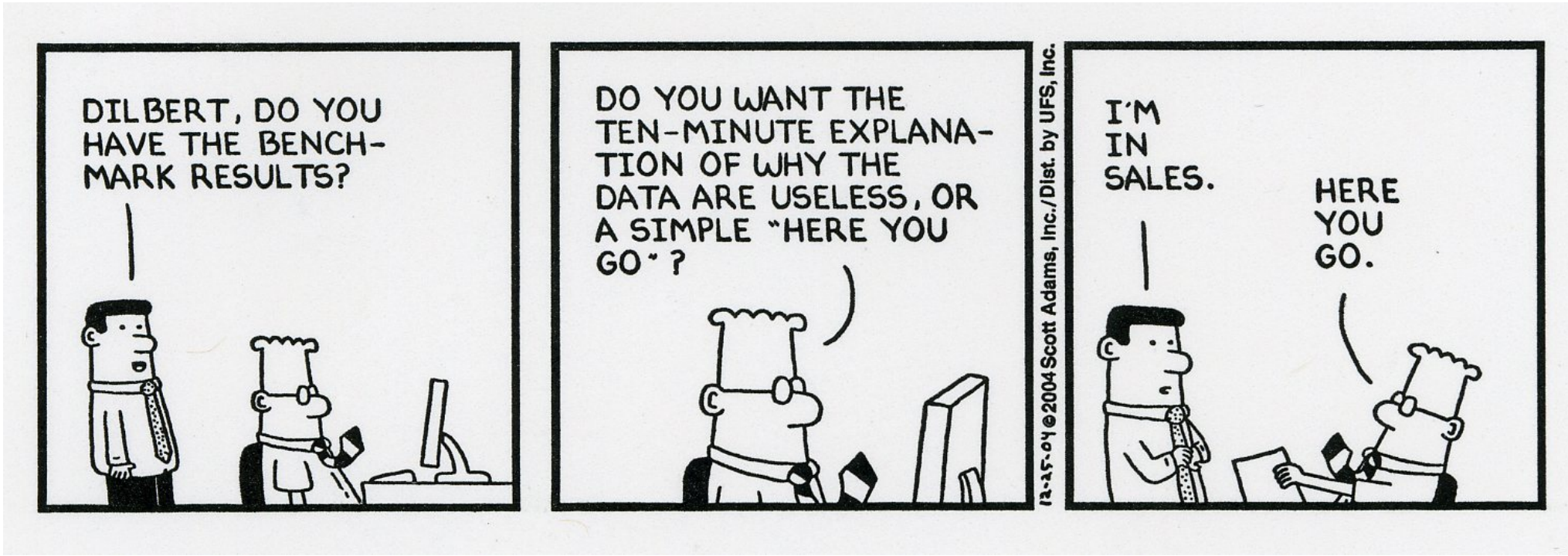
```
# id -Z
staff_u:WebServer_Admin_r:WebServer_Admin_t:s0:c0
# ls -l /data
secret-file-1
secret-file 2

# id -Z
staff_u:WebServer_Admin_r:WebServer_Admin_t:s1:c0
# ls -l /data
secret-file-1
secret-file 2
top-secret-file-1
```

The Three SELinux Policy Types

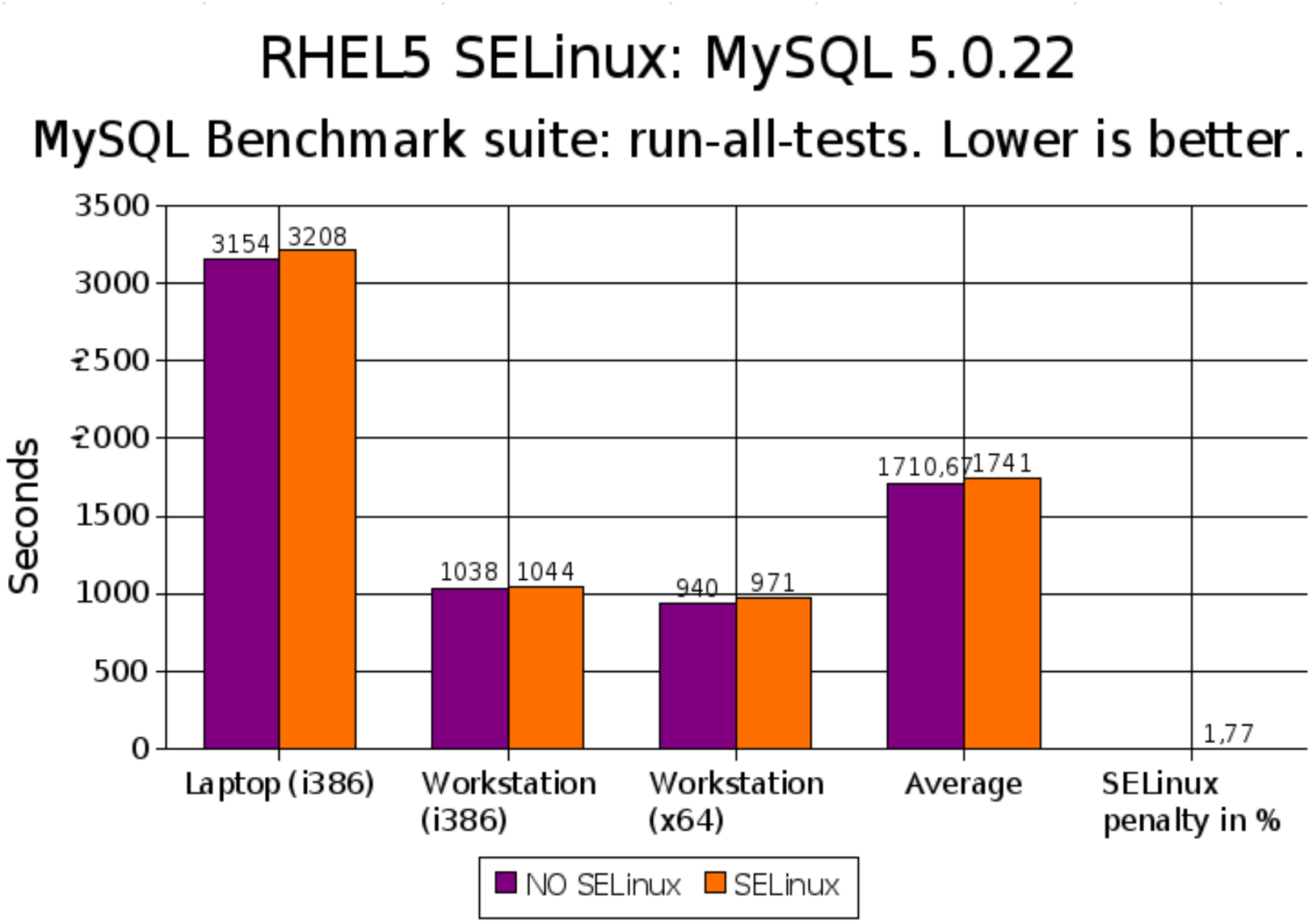
Multi-Level Security (MLS) & Common Criteria

- The Common Criteria (CC) is an international security standard against which systems are evaluated. Many government customers require CC evaluated systems.
- Red Hat Enterprise Linux 5 meets EAL4+ with RBAC/LSPP/CAPP endorcements

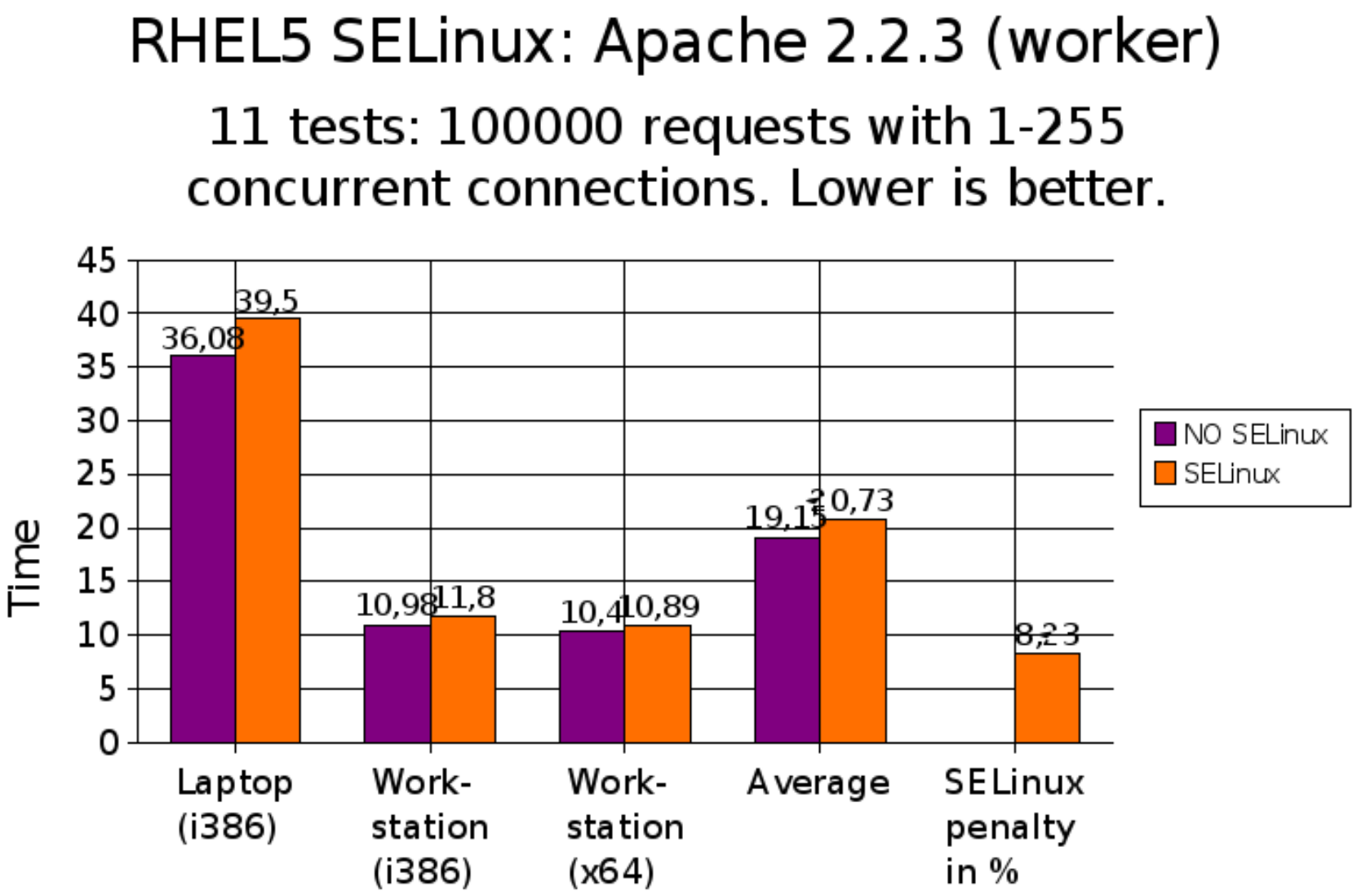
What's the Performance Overhead?



What's the Performance Overhead?



What's the Performance Overhead?



What's the Performance Overhead?

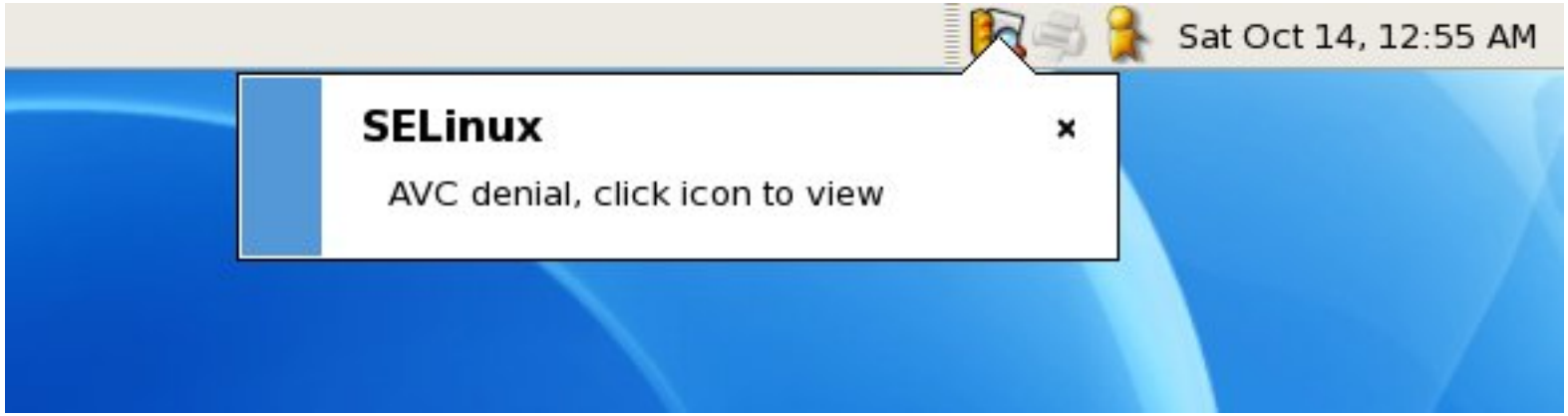
- Not official statistics
- Laptop = 2GHz, 2x 1GB RAM
- Workstation = 2.13GHz, 4x 1GB RAM
- Apache = Lots of threads
- MySQL = Lots of disk I/O

SELinux Usage

(GUI & console)

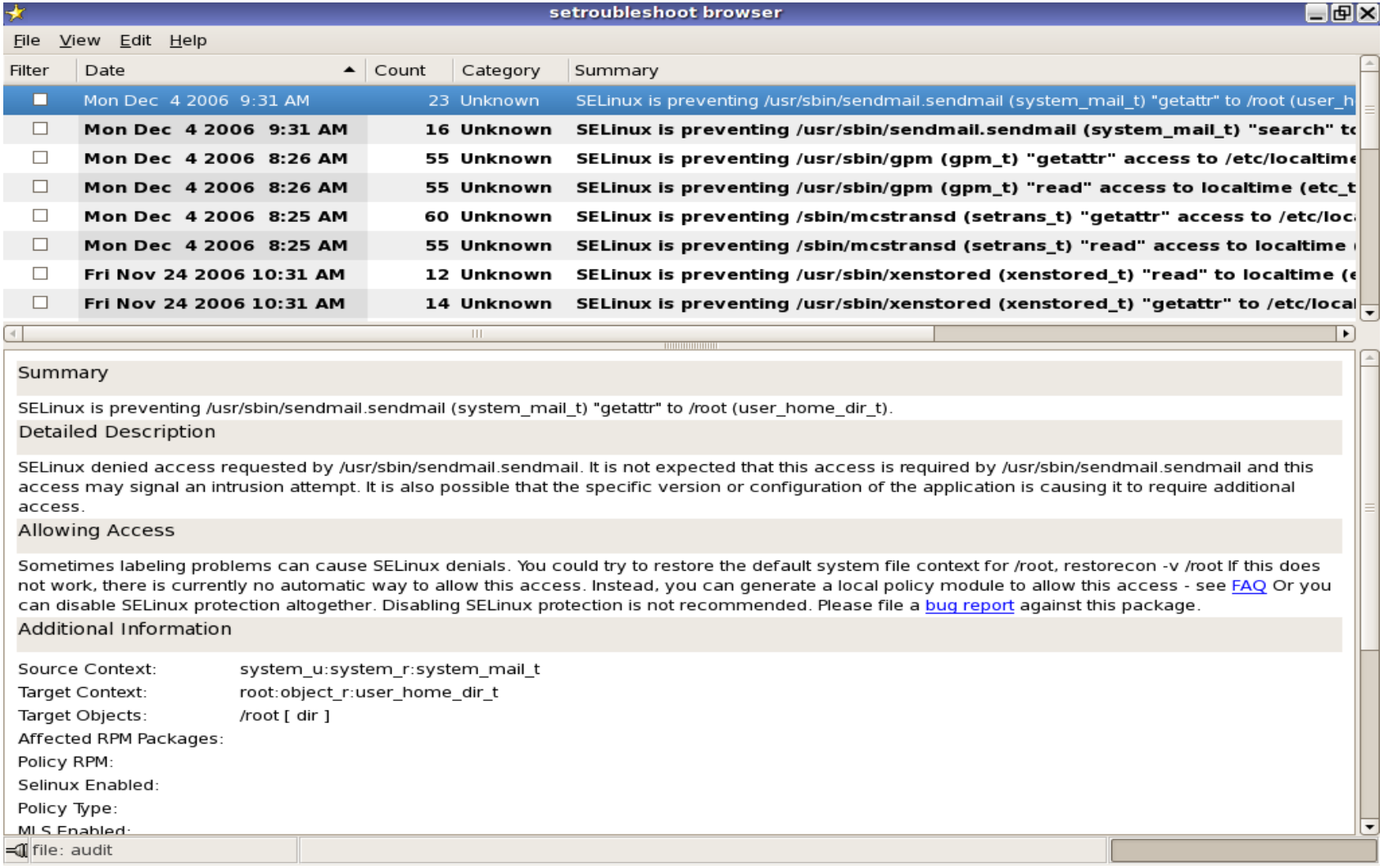
End-User Perspective

- **sealert Notifications**



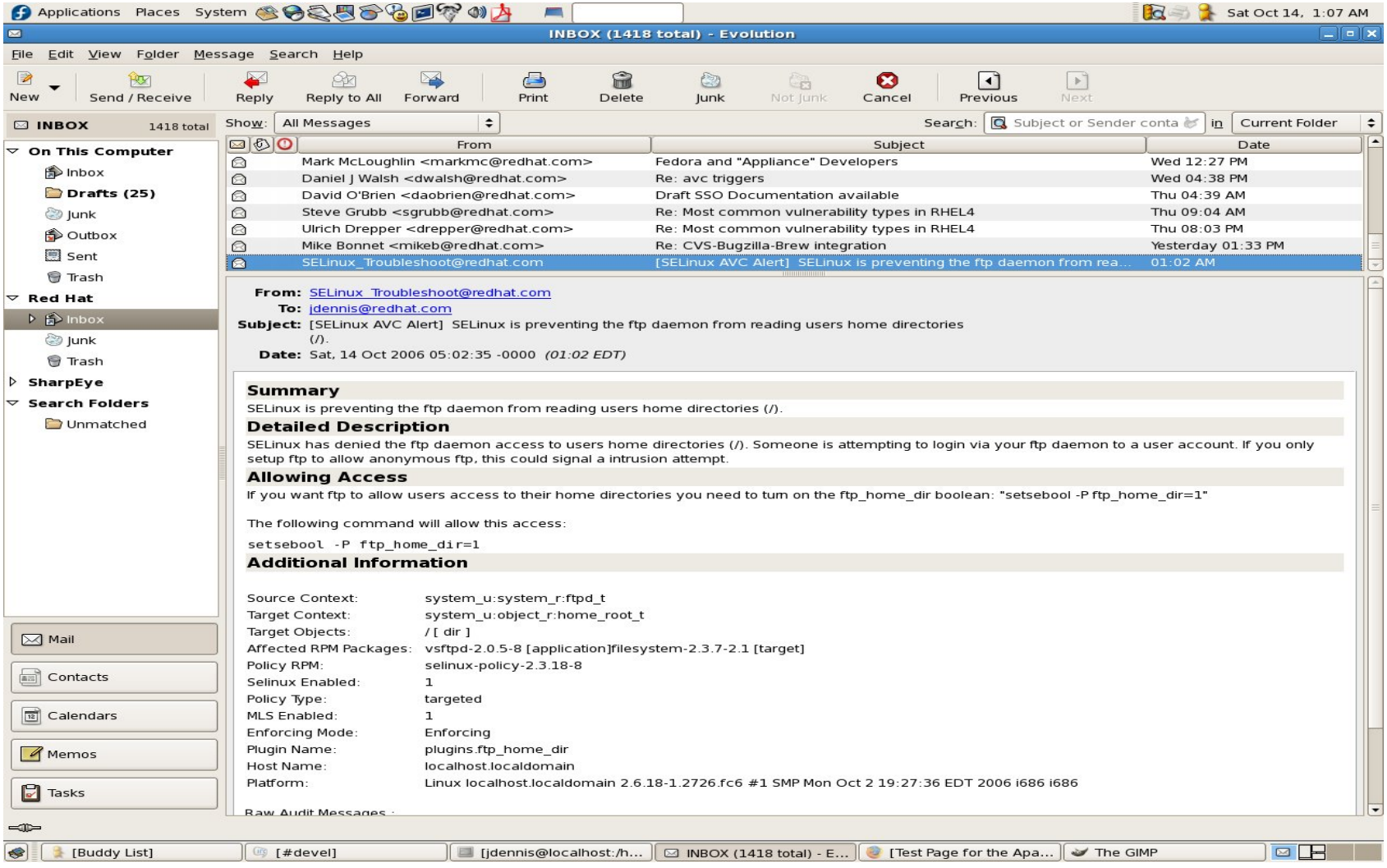
End-User Perspective

- sealert Browser



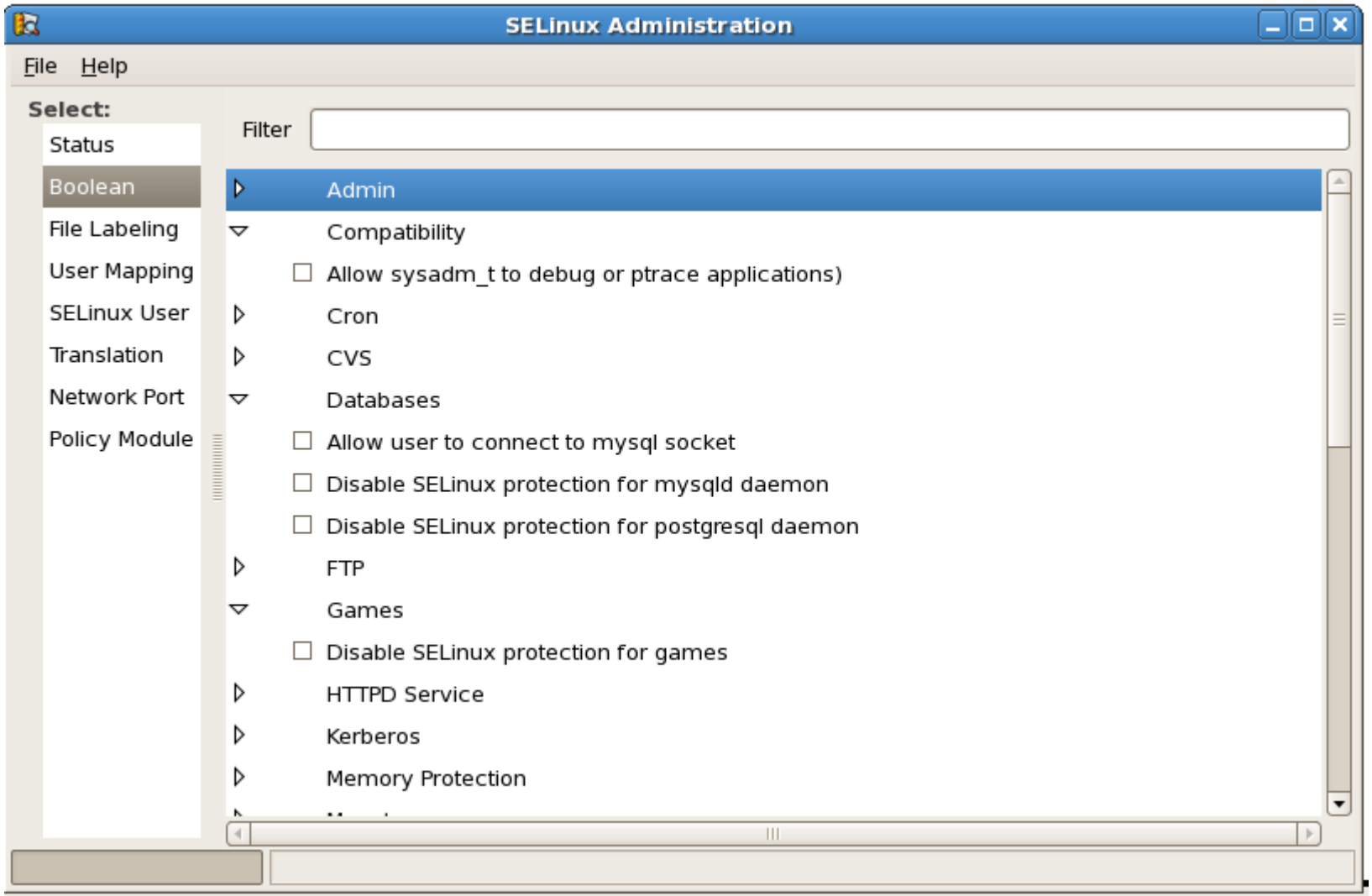
System Administrator Perspective

- **sealert + EMail Notifications**



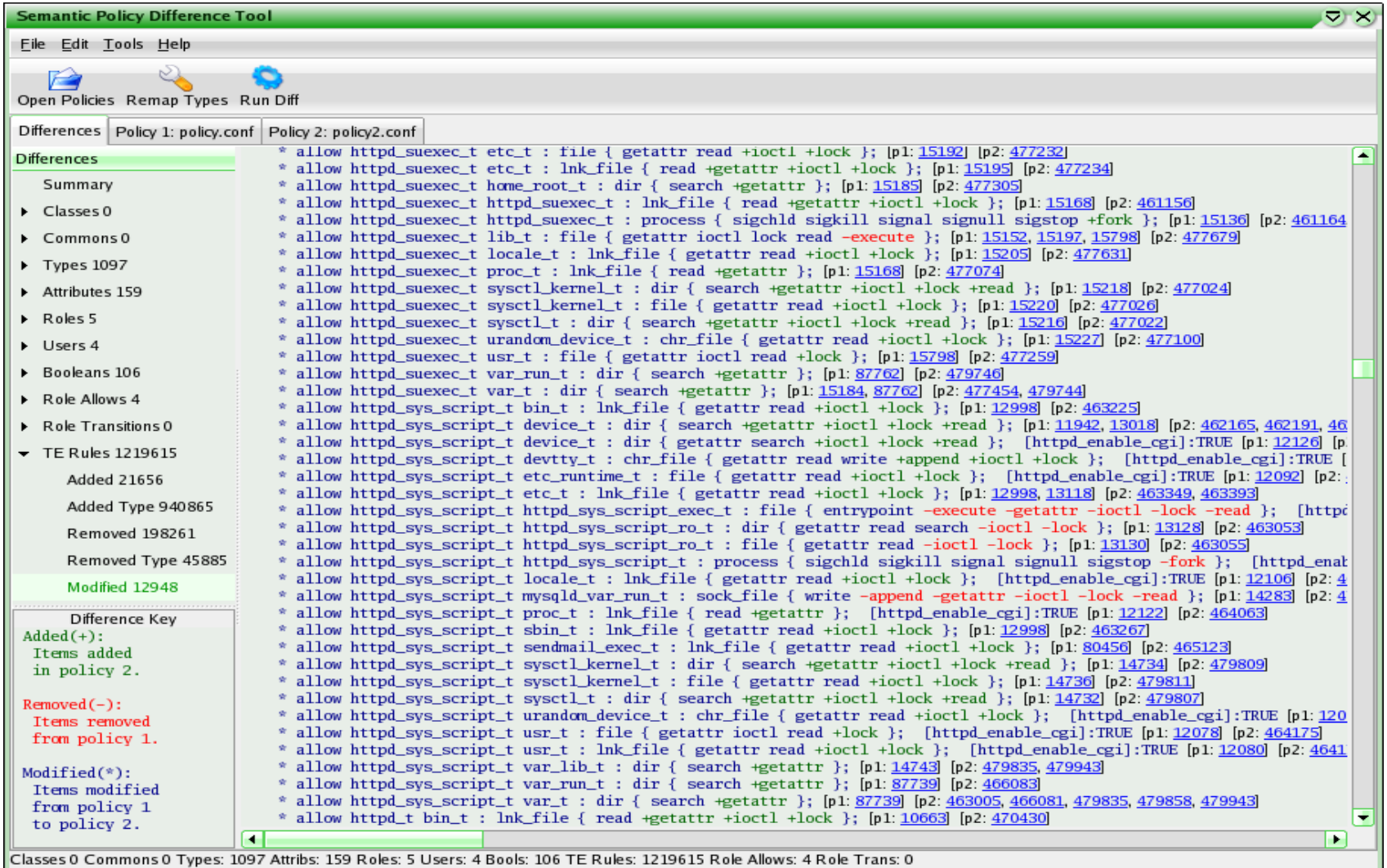
System Administrator Perspective

- system-config-selinux



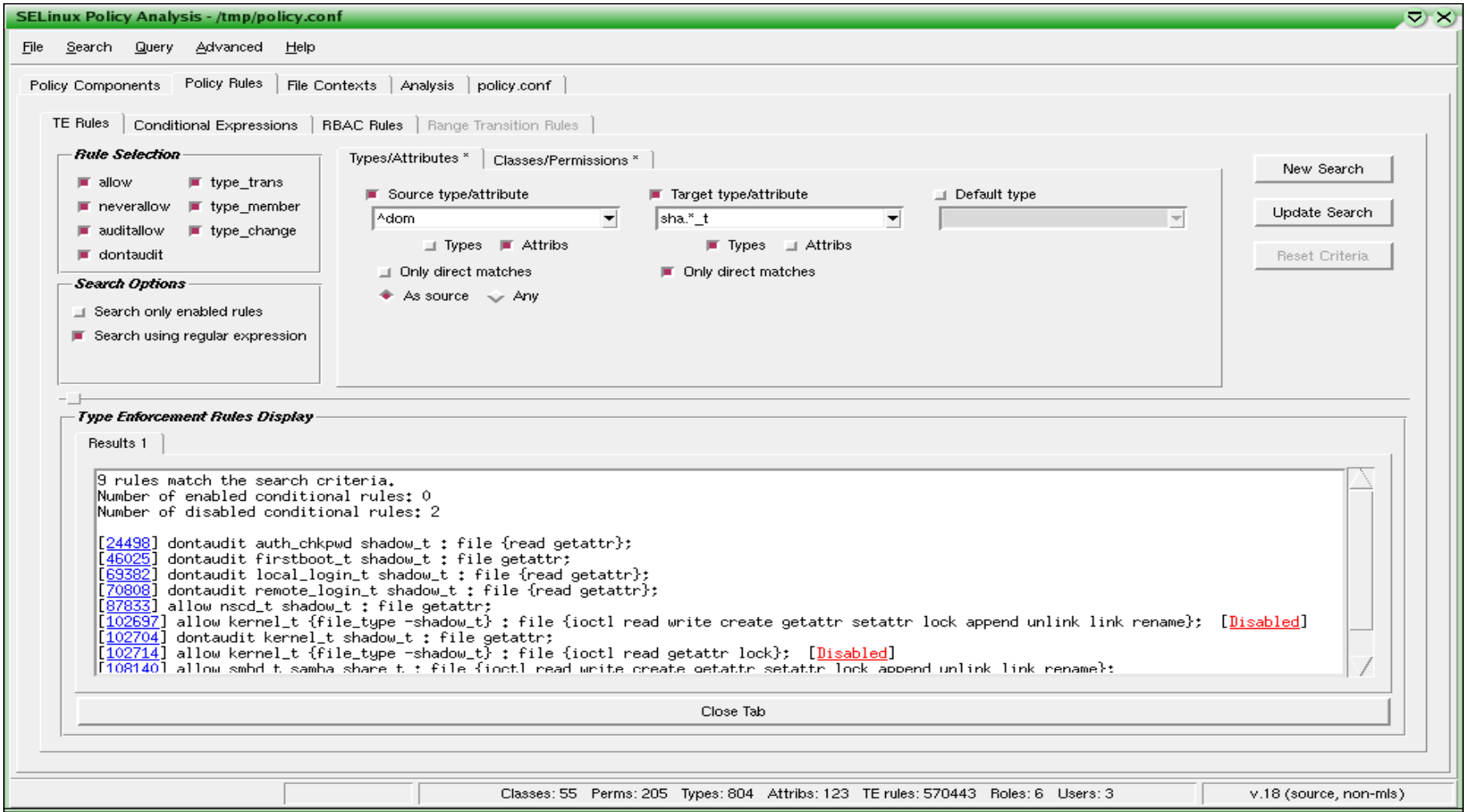
System Administrator Perspective

- **sediffx**



System Administrator Perspective

- apol



SELinux Usage

(Hints & Tips)

System Administrator Perspective

- **semanage**

  Configure elements of SELinux policy without modification/recompilation of policy sources

  . . . . aka on the fly

Example: Dynamically Allowing Apache to listen on port 1234

```
# semanage port -a -t httpd_port_t -p tcp 1234
```

System Administrator Perspective

- **semanage** (more examples)

Example: Allow `shawn` to join "`webadmin_u`" group

```
# semanage login -a -s webadmin_u shawn
```

Example: Relabel files for access by Apache

```
# semanage fcontext -a -t \
  httpd_sys_content_t "/data/webpages(/.*)?"
```

System Administrator Perspective

- **semanage** (most important example)

You don't need to disable SELinux to fix a single error!

```
type=SYSCALL msg=audit(1204719775.306:738): arch=40000003 syscall=54
success=no exit=-19 a0=4 a1=8933 a2=bfcec1bc a3=bfcec1bc items=0
ppid=3900 pid=5003 auid=501 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0
sgid=0 fsgid=0 tty=(none) comm="ip" exe="/sbin/ip"
subj=user_u:system_r:ifconfig_t:s0 key=(null)
```

The Fix:

```
# semanage permissive -a ifconfig_t
```

System Administrator Perspective

- **audit2allow**

  Allows generation of SELinux policy rules from logs of denied operations

Example: Fix all the errors on the system (completely not a good idea on a real system)

```
# cat /var/log/audit/audit.log | audit2allow -M FixAll
Generating type enforcment file: FixAll.te
Compiling policy: checkmodule -M -m -o FixAll.mod FixAll.te
Building package: semodule_package -o FixAll.pp -m FixAll.mod

# semodule -i FixAll.pp
```

Scenarios

Scenario: Fixing the RHT corporate VPN “update”

- Red Hat has a Corporate Standard Build (CSB) for desktop environments
- Red Hat pushes updates to said CSB
- I “tweak” my configuration files
- When RHT pushed a CSB update, it broke my VPN settings

Scenario: Fixing the RHT corporate VPN “update”

/var/log/messages:

```
type=SYSCALL msg=audit(1204719775.306:738): arch=40000003 syscall=54
success=no exit=-19 a0=4 a1=8933 a2=bfcec1bc a3=bfcec1bc items=0
ppid=3900 pid=5003 auid=501 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0
sgid=0 fsgid=0 tty=(none) comm="ip" exe="/sbin/ip"
subj=user_u:system_r:ifconfig_t:s0 key=(null)
```

Now what?

Scenario: Fixing the RHT corporate VPN “update”

```
type=SYSCALL msg=audit(1204719775.306:738): arch=40000003 syscall=54
success=no exit=-19 a0=4 a1=8933 a2=bfcec1bc a3=bfcec1bc items=0
ppid=3900 pid=5003 auid=501 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0
sgid=0 fsgid=0 tty=(none) comm="ip" exe="/sbin/ip"
subj=user_u:system_r:ifconfig_t:s0 key=(null)
```

What I Know:

1) AVC Event ID 738
2) syscall=54 (I'd have to google this)
3) root (or an application on its behalf) was running /sbin/ip
4) context = user_u:system_r:ifconfig_t:s0

Scenario: Fixing the RHT corporate VPN “update”

```
type=SYSCALL msg=audit(1204719775.306:738): arch=40000003 syscall=54
success=no exit=-19 a0=4 a1=8933 a2=bfcec1bc a3=bfcec1bc items=0
ppid=3900 pid=5003 auid=501 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0
sgid=0 fsgid=0 tty=(none) comm="ip" exe="/sbin/ip"
subj=user_u:system_r:ifconfig_t:s0 key=(null)
```

My Options:

1) Create a SELinux Policy Module

```
# ausearch -x “/sbin/ip” | audit2allow -M MyVPNFix
```

Scenario: Fixing the RHT corporate VPN “update”

```
type=SYSCALL msg=audit(1204719775.306:738): arch=40000003 syscall=54
success=no exit=-19 a0=4 a1=8933 a2=bfcec1bc a3=bfcec1bc items=0
ppid=3900 pid=5003 auid=501 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0
sgid=0 fsgid=0 tty=(none) comm="ip" exe="/sbin/ip"
subj=user_u:system_r:ifconfig_t:s0 key=(null)
```

My Options:

1) Create a SELinux Policy Module

```
# ausearch -x “/sbin/ip” | audit2allow -M MyVPNFix
# semodule -i MyVPNFix.pp
```

Scenario: Fixing the RHT corporate VPN “update”

```
type=SYSCALL msg=audit(1204719775.306:738): arch=40000003 syscall=54
success=no exit=-19 a0=4 a1=8933 a2=bfcec1bc a3=bfcec1bc items=0
ppid=3900 pid=5003 auid=501 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0
sgid=0 fsgid=0 tty=(none) comm="ip" exe="/sbin/ip"
subj=user_u:system_r:ifconfig_t:s0 key=(null)
```

My Options:

2) Disable enforcement of ifconfig_t (**there is no need to turn SELinux completely off!**)

```
# semanage permissive -a ifconfig_t
```

Questions