Managing SELinux in SUSE®
Linux Enterprise Server 12
TUT 7986

Sander van Vugt
Author, Technical Trainer and Consultant
Sandervanvugt.nl
Agenda

What is SELinux?

SELinux State in SUSE® Linux Enterprise Server 12

Understanding SELinux Modes

The SELinux Policy

SELinux Access Control

Managing SELinux

Troubleshooting SELinux

Customizing SELinux Policy

Creating Custom Rules

Using SELinux Users and Roles
Why do you need Kernel Level Security?
What is SELinux?
What is SELinux?

• All syscalls are denied by default, unless specifically enabled

• All objects (files, ports, processes) are provided with a security label (the context)
  – User, role and type part in the context
  – Type part is the most important

• The SELinux policy contains rules where you can see which source context has access to which target context
SELinux Components

• The policy is the key component

• It defines security contexts for type enforcements
  – That means: if you use `ls -Z` on a directory, you'll find a context user, role and type and the policy knows what to do with it.

• It all comes down to a line like the following that is defined in the policy:
  – allow user_t bin_t file {read execute getattr};
    – Which states that user_t (the source) has allow to bin_t (the target) on the object class file with the permissions read execute and getattr.

• The SELinux Policy contains thousands of rules
SELinux versus AppArmor

• Both are supported in SUSE Linux Enterprise Server
• SELinux is becoming the de facto standard
  – wide support
  – relatively difficult configuration
  – starts from an all denied situation
• AppArmor also offered
  – default profiles available
  – relatively easy to create new configurations
SELinux State in SUSE Linux Enterprise Server 12
SELinux State in SUSE Linux Enterprise Server 12

- Full support for binaries and kernel support

- No SUSE Linux Enterprise Server policy yet

- Policy expected for SUSE Linux Enterprise Server 12 SP1

- Currently, the OpenSUSE 13.1 policy works quite good
Enabling SELinux in SUSE Linux Enterprise Server 12

- zypper in selinux-tools selinux-policy

- download selinux-policy-targeted from opensuse and install
  - ensure version matches that of selinux-policy
  - install both policy packages

- run `selinux-ready`; it will tell you what you need to do to enable SELinux
  - Add `security=selinux selinux=1` to the kernel boot parameters and don't forget to update grub using `grub2-mkconfig /boot/grub2/grub.cfg`.
  - Run `pam-config --a --selinux`

- reboot
Finalizing SUSE Linux Enterprise Server 12 SELinux Configuration

- Relabel the entire filesystem: `restorecon -R /
- Start the auditing service for messages in `/var/log/audit/audit.log`
  - `systemctl start auditd`
  - `systemctl enable auditd`
- `reboot`
- Type `sestatus` to verify current status
Understanding SELinux Modes
SELinux States and Modes

• Enabled
  - Enforcing: fully functional
  - Permissive: not blocking anything, but logging
    - SELinux support is available in the kernel, so applications will load with all SELinux libraries and behave differently
    - Expect unexpected behavior on some occasions.
    - Log files are in /var/log/audit.log
      - Understand timestamps in audit.log using `date –d @timestamp` (e.g.: `date –d @1413359626`)

• Disabled
  - SELinux support is not available in the kernel, so applications will load differently
The SELinux Policy
Refpolicy: the Mother of All Policies

• The refpolicy is a generic fully functional policy that is managed as a free software project

• Application developers provide code for the refpolicy, where after peer review it can be included

• Refpolicy is a common base for distributions that only need to make modifications to it

• Because of differences that in some cases are big, making changes can be hard.
Policy Features

- Targeted is the normal policy, which works with context labels only
- Multi Level Security (MLS) is used to give every object a security clearance label
- Multi Category Security (MCS) is like MLS but less detailed
- Support of features is indicated by policy version – current version is 28
  - use **sestatus** to find out
- Several options need to be compiled in if desired
  - Handling of unknown permissions
  - Support for unconfined domains
SELinux Access Control
SELinux Access Control

• Type Enforcement is important in the targeted policy
  - Find out using –Z option on several commands
    - netstat –Ztulpen
    - ps –Zaux
    - ls –Z

• Access is allowed between similar source and target types
  - This prevents services from accessing user files
  - User processes are typically running as unconfined
Managing SELinux
Managing SELinux Means Applying Context

- Context on files are set in the policy and applied to the filesystem
  - `semanage fcontext -a -t http_sys_content_t "/web(/.*)?"`
  - `restorecon -Rv`

- Do not use chon
  - chcon is evil
  - context applied to the inode, not to the policy
  - it won't survive an autorelabel
Finding the Right Context

- Easiest: check default objects
- Use `semanage fcontext –l` to get a list of all context settings in the policy
- Get the information from the policy, using `seinfo –t`
- Where available, use `_selinux` manpages (`man –k _selinux`)
Applying Port Security

- Applications (**ps Zaux**) and ports have context labels also.

- Managing port context can be required
  
  ```
  semanage port –a –t ssh_port_t –p tcp 2022
  semanage port –m –t ssh_port_t –p tcp 443
  ```
  
  -m is necessary to relabel a port that has a current label applied already.
Using Booleans

- Booleans provide an easy interface to change settings in the policy

- Use `semanage boolean --l` for an overview of available booleans

- Use `getsebool --a` alternatively

- Set booleans using `setsebool --P yourboolean [0|1]`
Troubleshooting SELinux
Using sealert

- Created to make human readable reports based on /var/log/audit/audit.log
  - Displays analysis as well as recommended action
- Matches against an event database and gives every solution a probability score
- When applied to /var/log/messages, **sealert** is used with a UUID on specific events
- Or use **sealert -a /var/log/audit/audit.log** to generate messages for all events that have happened
- Use **sealert -b** for a graphical interface
Reading Boolean Content

• `sesearch -b allow_ftpd_anon_write –ACT`
  - First character in output shows state in the policy (Disabled or Enabled)
  - Second character shows if the displayed rule is enabled or disabled (True or False)

• Search rules with a specific permission
  - `sesearch -b allow_ftpd_anon_write –p read -AC`
Reading SELinux Labels

- `sesearch –s httpd_t –t user_home_t –p read –AC`
  - shows all allow rules where httpd_t gets access to user_home_t
Finding Out What an Application Can Do

- First, find the source context type set to your applications
  - `ps Zaux | grep http` would give `httpd_t`

- Use `sesearch –A | grep httpd_t` to see all allow rules and to what specifically access is allowed.
  - These are *existing* rules, not just *effective* rules!

- Use `sesearch –AC | grep httpd_t` instead.
  - This shows the boolean needed to allow the rule displayed, and its current state. (ET / DF = Enabled, True / Disabled / False)
SELinux and Unsupported Applications

- Most applications are not SELinux aware
  - In general, they could be integrated with SELinux

- Some applications are and they make active calls to SELinux libraries
  - The applications behave differently if SELinux code is active
  - Use `ldd` to find out if SELinux libraries are used
  - On SUSE SELinux native applications are rare
Disabling SELinux for Specific Context Types

- `semanage permissive –a somelabel_t`

- Switch off using `semanage permissive –d somelabel_t`

- Use `semanage permissive –l` for an overview of domains that are currently set to permissive

- Don't use this too much because it makes your system more vulnerable!
Customizing SELinux Policy
Customizing SELinux Policy

- Booleans provide an easy interface to customize policy
- Modules can be used to provide basic support for SELinux functionality
  - `semodule -l` to list
  - `semodule -f` to shut off all rules for a part of the system
  - `audit2allow` is provided as easy-to-use interface to compile your own
Understanding Policy Modules

- *.te files contain all rules that are compiled into the policy (these files are key)

- *.if files define how other policy modules get access to this policy

- *.fc files contain labeling instructions

- *.pp files are the binary policy modules
Understanding Policy Modules

• After using `audit2allow` a *.te file and a *.pp file are created

• If repolicy is installed, find these files in `/etc/selinux/refpolicy/modules/services`

• Manual changes are allowed but not recommended
  
  – After making modifications, run `make && make install && make load`
Using audit2allow

• audit2allow is dangerous if you don't know what you're doing!

• Example: `grep http /var/log/audit/audit/log | audit2allow –M mypolicy`
  - Creates mypolicy.te and mypolicy.pp
  - Read mypolicy.te to see what it is doing
  - Use `semodule –i modulename.pp` to run the newly created policy module
Creating Custom Rules
Creating Custom Rules

- In a modular policy, the source files of the policy modules are where you want to apply modifications.
- In particular, look at the *.te files that contain what exactly has to be done.
- **audit2allow** is a reactive interface to generate some *.te files.
- Use policy sources for full access.
Understanding Custom Rules

- Default rule syntax:
  - allow <source> <destination> : <class> <permissions> ;
  - See audit.log for examples

- Source is always a domain

- Destination can be anything

- <class> is the thing that is accessed in the target
  - file, directory, socket, capability, etc
  - use seinfo –c for a complete overview

- Each class has specific permissions associated to it
  - Use seinfo –c<class> -x to show
  - As in seinfo –cfile -x
Translating Audit Message to Custom Rules

• Consider this message in audit.log

  type=AVC msg=audit(1413357425.988:1060): avc: denied { name_bind } for pid=29198 comm="sshd" src=443 scontext=unconfined_u:system_r:sshd_t:s0-s0:c0.c1023 tcontext=system_u:object_r:http_port_t:s0 tclass=tcp_socket

• Which translates into the following rule if you want to allow: `allow sshd_t http_port_t : tcp_socket { name_bind };`

• Don't do this manually, use `audit2allow` instead!

  - `grep ssh /var/log/audit/audit.log | audit2allow -M mypolicy`
Manually Adding Policy Files (1 of 2)

- Start by creating a .te file (~/.sander.te)

```plaintext
module sanderpolicy 1.0;
require {
    type sshd_t;
    type http_port_t;
    class tcp_socket { name_bind };
}
allow sshd_t http_port_t:tcp_socket { name_bind };
```
Manually Adding Policy Files (2 of 2)

- Create the policy module:
  - `checkmodule -M -m -o sander.mod sander.te`
  - `semodule_package -o sander.pp -m sander.mod`

- Run the policy module
  - `semodule -i sander.pp`

- Enable the policy module
  - `semodule -e sander.pp`
Using SELinux Users and Roles
Managing SELinux Users

• By default, users are logged in as unconfined_u

• Some default user roles exist:
  – user_u: regular restricted users
  – staff_u: operators
  – sysadm_u: system admins
  – custom users can be created

• An SELinux user defines the roles that a user can switch to

• Use `semanage user -l` for an overview
Understanding Unconfined

• By default, unconfined_u users have access to items running in unconfined domains.
  - Use `seinfo –aselinux_unconfined_type –x` to find out what exactly those are

• Linux users on login by default are all mapped to the unconfined_u user
  - `semanage login –l` shows available mappings
  - `__default__` is mapped to unconfined_u, as is the root user
  - `_system_u` is for all system processes
Creating SELinux Users

- `semanage login --a --s user_u linda`
  - Creates a user linda that is mapped to the SELinux user_u user role
  - When applying changes to existing users, you must relabel the homedirectory: `restorecon -RF /home/linda`

- `semanage login --a --s sysadm_u "%admins"`
  - Maps all members of the admins group to the SELinux sysadm_u user

- Use `semanage user --l` for an overview of current SELinux users

- Use `seinfo --adomain --r` for an overview of roles
Using roles

• A role is a collection of tasks that is allowed.
  - Use `seinfo --adomain --r` to show currently existing roles

• Users can enter a new role using the `newrole --r` command, as in `newrole --r sysadm_r`

• Services can be started in a specific role using `run_init /etc/init.d/myservice start`. This will start services in the `system_r` role, instead of the role of the current user
  - This happens as a default on most Linux distributions
  - No longer needed on systemd systems
    - From systemd perspective, services are started by systemd, not by the user
Specific SELinux Cases
Resetting the Root Password Using rd.break

• Understanding rd.break root password recovery
  – Root password is reset with SELinux disabled
  – When using passwd, a new temporary file is created. This file has no context label.
  – While booting, SELinux gets enabled and finds /etc/shadow without context labels, so shuts down all access to it
    – `ls -Z` will show `????` or unlabeled_t
  – To prevent, use `touch /.autorelabel` which relables all

• You can also consider using `load_policy -i` right after entering rd.break mode. This loads the policy and makes sure file labels are set correctly.
More information

Refpolicy project: http://oss.resys.com/projects/refpolicy

Dan Walsh blog: http://danwalsh.liveblog.com

Sven Vermeulen, SELinux System Administration
(ISBN 978-1-78328-317-0)
Check it out yourself!

QUESTIONS?

Thank you.
Unpublished Work of SUSE. All Rights Reserved.
This work is an unpublished work and contains confidential, proprietary, and trade secret information of SUSE. Access to this work is restricted to SUSE employees who have a need to know to perform tasks within the scope of their assignments. No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of SUSE. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.

General Disclaimer
This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. SUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for SUSE products remains at the sole discretion of SUSE. Further, SUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All SUSE marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.