What is SELinux trying to tell me?

The 4 key causes of SELinux errors.

Daniel J Walsh
Principal Software Engineer
Lead SELinux developer
Red Hat

I have just completed a project to secure the Fedora Infrastructure, and plan on writing up what we did in a future Red Hat Magazine Article. After many years of looking at SELinux error messages I have found that almost all SELinux problems fall into one of the following categories.

- Labeling Problems
- A confined process is configured in a way different then the default SELinux expected.
- Bug in Policy or an Application.
- Your machine has been compromised.

I use this list to try to figure out what the problem is, when I am confronted with SELinux problem,

Is there a labeling problem?

SELinux == LABELS

When using SELinux the most important thing to understand is SELinux is all about labels. Every process, file, directory, device on an SELinux system has a label. If these labels are wrong for some reason SELinux will not function properly.

If a file is mislabeled a confined application might not be allowed access to the mislabeled file. If an executable is mislabeled, it may not transition to the correct label when executing, causing access violations and potentially causing it to mislabel files it creates. Processes and objects on the machines have labels. If the labeling is correct everything should work.

Sometimes an admin or software developer decides to change the the location of files used by a confined domain. For example if an administrator wants to store web pages in a unusual location, /srv/myweb. The administrator needs to tell the SELinux system that these files need to be accessible to the web server process. He does this by setting the labeling correctly in the system. So since the httpd process is allowed to access files labeled with the httpd_sys_content_t type, he needs to set the label. One mechanism for this is to use the chcon command.

```
# chcon -R -t httpd_sys_content_t /srv/myweb
```

This will set the labels correct, however you have not told the SELinux system to permanently label these files/directories with this label. In some circumstances a system Relabel could change these labels back to the default. You use the semanage command to make permanent changes to the SELinux system.

```
# semanage fcontext -a -t httpd_sys_content_t '/srv/myweb(/.*)?
```

This command tells the SELinux data store that the /src/myweb directory and all files under it should be labeled httpd_sys_content_t. Tools like restorecon and rpm read this data store when they are labeling or relabeling files. Note, however that the semanage command will not change the actual labels on files on your machine. You still need to execute restorecon to fix the labels.

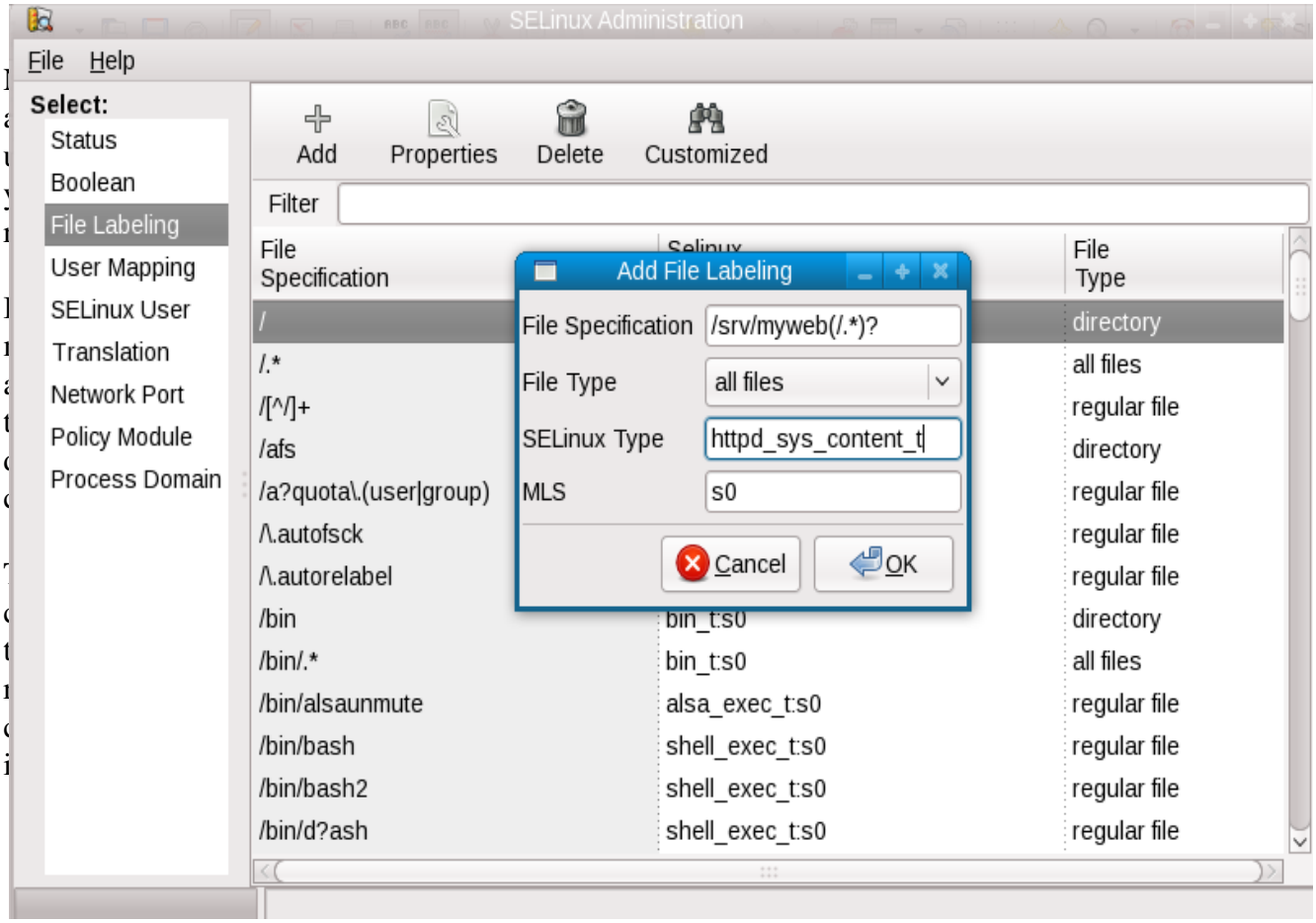
```
# restorecon -R /srv/myweb
```

restorecon reads the SELinux data store to determine how files under /srv/myweb should be labeled and then fixes them.

```
# matchpathcon /srv/myweb
```

The matchpathcon command reads the SELinux file context files and prints the default label for the specified path.

You can also use system-config-selinux to setup you SELinux labeling



Are my confined applications setup differently then the default?

SELinux has got to know????

A confined process/application can be run in many different ways. You need to tell SELinux how you have configured the application to run, and then SELinux will allow it the proper access. SELinux does not do this automatically.

SELinux policy has built-in if/then/else rules called booleans that allow you to tweak the predefined rules to allow different access. Out of the box SELinux policy for httpd does not allow it to send mail. We do this to prevent a compromised web site from becoming a spam box. But some administrators might want their httpd to send mail legitimately.
So you can set the httpd_can_sendmail boolean to tell SELinux it is ok to connect to the mail port or to execute one of the commands used to send mail.

```
# setsebool -P httpd_can_sendmail 1
```

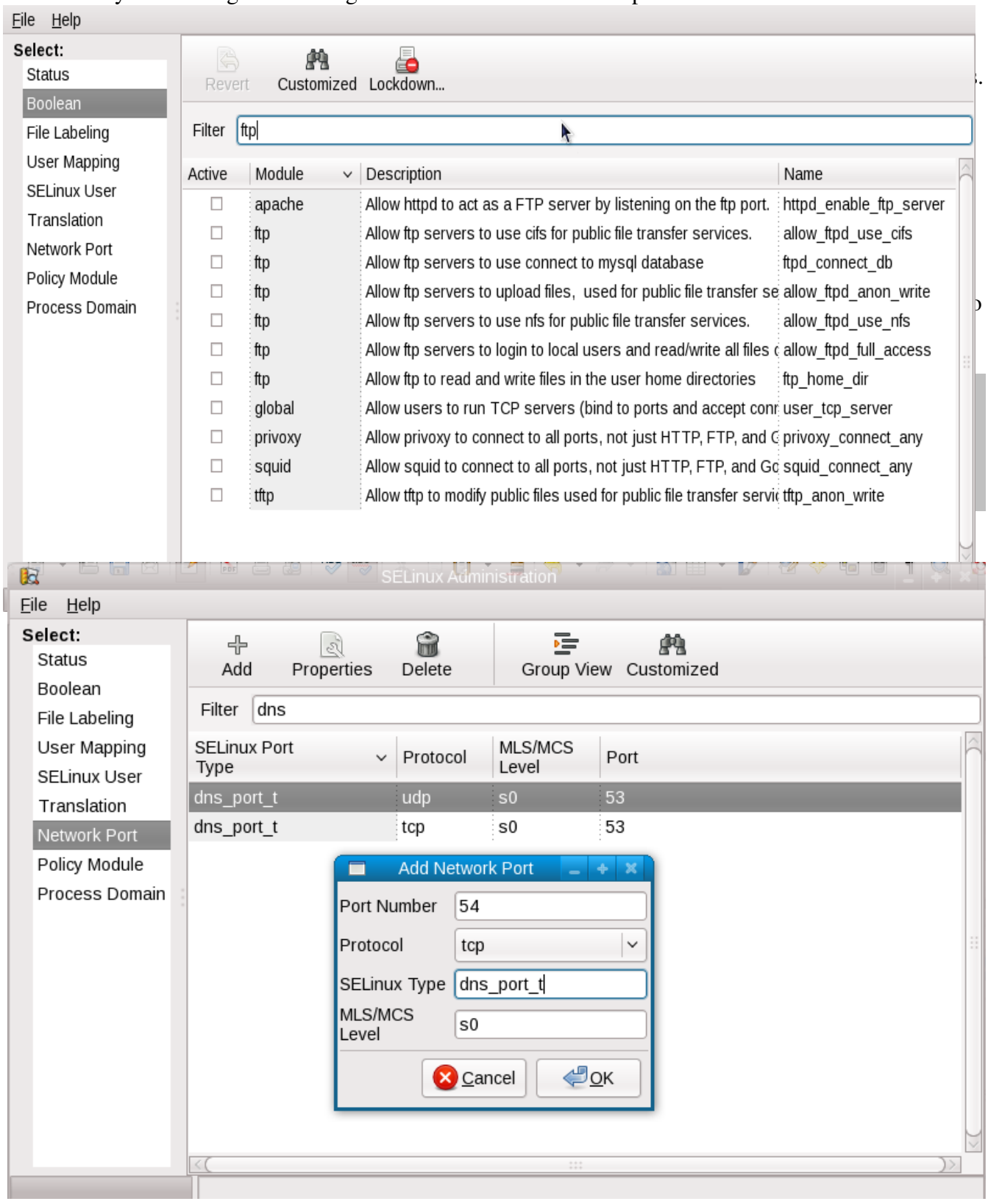
This permanently changes SELinux policy to allow httpd to send mail.

To view all booleans for http you can execute

```
# semanage boolean -l | grep http
httpd_can_network_relay        -> off   Allow httpd to act as a relay
httpd_can_network_connect_db   -> off   Allow HTTPD scripts and modules to connect to databases
over the network.
httpd_enable_cgi               -> on    Allow httpd cgi support
httpd_use_cifs                 -> off   Allow httpd to access cifs file systems
allow_httpd_mod_auth_pam       -> off   Allow Apache to use mod_auth_pam
allow_httpd_anon_write         -> off   Allow Apache to modify public files used for public file transfer
services. Directories/Files must be labeled public_content_rw_t.
httpd_enable_homedirs          -> on    Allow httpd to read home directories
allow_httpd_sys_script_anon_write -> off   Allow apache scripts to write to public content.
Directories/Files must be labeled public_content_rw_t.
httpd_dbus_avahi               -> on    Allow Apache to communicate with avahi service via dbus
httpd_can_sendmail             -> off   Allow http daemon to send mail
httpd_unified                  -> on    Unify HTTPD handling of all content files.
httpd_can_network_connect      -> off   Allow HTTPD scripts and modules to connect to the network
using TCP.
allow_httpd_mod_auth_ntlm_winbind -> off   Allow Apache to use mod_auth_pam
httpd_tty_comm                 -> on    Unify HTTPD to communicate with the terminal. Needed for
entering the passphrase for certificates at the terminal.
httpd_use_nfs                  -> off   Allow httpd to access nfs file systems
httpd_execmem                  -> off   Allow httpd scripts and modules execmem/execstack
httpd_built-in_scripting       -> on    Allow httpd to use built in scripting (usually php)
httpd_ssi_exec                 -> off   Allow HTTPD to run SSI executables in the same domain as system
```

CGI scripts.
httpd_enable_ftp_server -> off Allow httpd to act as a FTP server by listening on the ftp port.

Tools like system-config-selinux or getsebool -a will list all of the possible booleans.



Is there a bug in SELinux Policy or in a confined application?

SELinux policy and/or applications bugs are out there.

We write SELinux policy for a confined domain by looking at what application does and then putting the application or system into permissive mode, collecting the AVC messages and then update the policy using these messages. Sometimes an confined application is run with a code path that the policy writer did not know about, so the policy denies the access even though it should be allowed. While the application is working correctly, SELinux denies the access. After reporting this type of problem to support or as a bugzilla, you can add custom policy to your system simply by piping the SELinux error messages through audit2allow. Say a new version of postgresql comes out that SELinux is mistakenly denying access to a resource which it should be allowed to access. You can use audit2allow to build a custom policy module that can be installed on your system to allow the access.

```
# grep postgresql /var/log/audit/audit.log | audit2allow -R -M mypostgresql
```

This command will generate a local policy module which will allow all accesses that are currently being denied. It generates a file called mypostgresql.te which contains all of the allow rules. You should examine these before installing them to attempt to make sure it is safe to install the rules. You can always use the SELinux developers mail list or the fedora-selinux mail list for help understanding if it is safe to add the rules.

```
# semodule -i mypostgresql.pp
```

This command installs the local policy modifications permanently to your system. You probably want to report the SELinux errors to bugzilla or a mailing list so your local modifications can be added to the distribution's policy or upstream.

In many cases we see bugs in applications, leaked file descriptors, apps attempting access that they really do not need, listing every device in /dev for example. Apps making assumptions about root being all powerful. You can use the custom policy modules described above to stop the errors, but you really need to report the problem.

Has your machine been compromised?

SELinux confines compromises...

SELinux is not an intrusion detection system, and our tools do not currently do a great job of distinguishing between an intrusion and a general configuration, labeling or SELinux policy error. There are several tools available to do intrusion detection and some use the SELinux logs to watch for intrusions.

SELinux will trigger lots of AVC's if an app is actually compromised and tries to do something it is not designed to do. SELinux development is planning modifying setroubleshooter to look for compromised applications signatures. Currently Setroubleshoot reports that all AVC's the same way. One idea is to change Setroubleshoot to put up a Red Star and a stern warning when certain signatures show up.

SELinux has been around for a while now and the policy is pretty good. If an application needs major security privileges, SELinux policy probably already allows it. If you see AVC's that do not make **sense** or seem to be an application trying to change security settings, your application might be compromised.

Potential signatures of a compromised confined application:

- Confined application should never try to change SELinux enforcement.
  - This includes changing the enforcement mode or try to write to /etc/selinux
  - Setting booleans would also be a very strange thing for a confined application to do.
- Confined application should not try to modify the kernel
  - load kernel modules
  - Write to kernel directories
  - Write to boot loader or image directories
- Confined should not be attempting to write to files labeled etc_t (/etc), Since a confined domain that can write to etc_t would be able to overwrite passwd_t.
- Confined applications should not be attempting to write to “security configuration files”, For example certificates, kerberos files, most configuration data.
- Most confined applications should not be attempting to write to shadow_t or in most cases reading shadow_t.
- Confined applications should not be trying to overwrite log files. Especially if the

log file has nothing to do with the application.

- Most confined applications should not be trying to read files in the users home directories (user_home_t). This is where the good stuff is
- Confined application should not suddenly attempt to connect to random network ports that do not make sense. Spam Bots will try to connect to the mail port.
- Confined application trying to execute mail programs or connect to the mail ports, when they were not setup to send mail. Most crackers try to grab machine to set them up as Spam Bots, so no root access required.

These might be a bug, but the damage that could be done is two significant to ignore. If you see something like the above, get help in diagnosing the AVC messages. Use open source mail list like the selinux list of the fedora-selinux list to ask for help.