Technical Report OUCS-2014-02

Practical use of SELinux for enhancing the security of web applications

Authors:

Lech Szymanski and David Eyers
Department of Computer Science, University of Otago, New Zealand

Practical use of SELinux for enhancing the security of web applications

Part 1: Using Type Enforcement Security

Lech Szymanski and David Eyers

August 14, 2014

Acknowledgement

The authors gratefully acknowledge the research grant from the University of Otago that funded the project that facilitated the writing of this document.

Contents

1 Introduction

The Security-Enhanced Linux (SELinux) module [1] has been available in the mainline kernel for many years (since 2003), and is included as part of a growing number of popular Linux distributions, such as Fedora. However adopting its new and powerful security capabilities has been daunting to many. On forums, the ubiquitous advice on many SELinux related issues is: "Disable SELinux". Though this might be a valid choice for a casual desktop user, it makes little sense to disable your security system if you intend to run any type of a server on your machine.

At its core, SELinux is an elegant, effective, and very flexible framework for providing Mandatory Access Control (MAC), yet it takes dedication and time to get used to. This is mostly because the security of a multi-service system is an extremely complicated matter, which often requires a thorough understanding of the inner workings of that system. There is an intricate interplay of processes, sockets, files and inter-process communication on a Linux server. The complexity in SELinux comes from the necessity to create a enormous number of rules, conveyed by the policy, that spell out the way processes and resources are allowed to interact with each other. Mastering SELinux is not just about getting to know the fundamental concepts of the way it implements MAC, but how the rules of SELinux policy permeate and affect interactions of various subsystems of your Linux server.

This tutorial came about as a result of an investigation into the viability of using SELinux to secure multi-tier web systems that process sensitive data, as inspired by the SafeWeb project [2]. The aim of this document is to showcase SELinux and the targeted policy—the default SELinux policy on Fedora—in action through case studies and examples of SELinux-aware web services. Although a brief introduction to SELinux concepts is included, it might be a good idea first to go through a general introduction to SELinux [3], [4].

This document is the first part of a two-part tutorial and it consists of the following chapters:

- Chapter 2 gives an overview of SELinux fundamentals and demonstrates the tools that are useful for querying the policy;
- Chapter 3 demonstrates how to work with existing policy using the process of setting up a secured DokuWiki[1] web server as an example;

[1] `https://www.dokuwiki.org`

- Chapter 4 continues with the DokuWiki example introducing the basics of policy writing in order to utilise SELinux as a second ring of security around the security that the DokuWiki service provides itself.

It's important to keep in mind is that this tutorial is a guide through a set of scenarios that (hopefully) will aid in understanding of how SELinux works. For clarity of the presentation, some liberties have been taken with various aspects of system security, which most likely would be not appropriate for production systems.

2 SELinux

The fundamental concepts of SELinux are relatively straightforward. However, in order to use SELinux, one needs to work within the framework of the default policy, which comes with an overwhelming number of labels, rules, macros and definitions. In this chapter, as we explain the principles of SELinux, we also demonstrate how to probe the active policy and find examples relevant to the covered theory.

2.1 Installation

All the exercises in this tutorial were done on Fedora 20 Desktop Edition running on VirtualBox. All the installation instructions relevant for the tutorial, except for how to install Fedora Linux itself, are provided – they are scattered throughout this document introducing the required components as the need arises. For the tutorial we created user `setest` with sudo privileges.

2.1.1 SELinux Tools

For this chapter we only need a set of the standard SELinux tools that provide a means of exploring the policy. From the terminal, issue the following commands to install the aforementioned tools:

```
$ sudo yum install policycoreutils-gui policycoreutils-newrole
$ sudo yum install setools setools-libs
```

2.2 SELinux architecture

The packages we have just installed are scripts that fish out information about the current SELinux configuration. SELinux itself didn't need to be installed, because Fedora comes with SELinux enabled by default. This can verified by issuing the following command from the terminal:

```
$ sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      28
```

Unless the default settings have been changed, the SELinux status should come up as *enabled*. Let's briefly go over the elements of the SELinux framework.

2.2.1 LSM

SELinux rides on top of the Linux Security Modules (LSM) framework [5], [6], which is a generic framework for Mandatory Access Control (MAC) for Linux. In contrast, the standard security approach of Linux filesystems is a type of Discretionary Access Control (DAC), because permissions for access to resources (files or directories) are at the discretion of their owners. For example, using the `chmod` command, one can choose to give access to their home directory to anyone on the system. MAC security does not allow users to make such decisions. Instead, all the decisions on what's permissible and what is not are conveyed through the policy. While it is possible to set up SELinux so that an arbitrary user can modify the policy, this would largely defeat the purpose of MAC, which is to centralise the decision making about access control.

LSM is a service that advises the kernel on what's permissible and what is not. It does not replace the existing DAC system. SELinux and the DAC system operate independently (which sometimes may be frustrating, because both must be configured properly in order to get a service working). Linux kernels since version 2.6 have been fitted with hooks that call LSM's API at various places before performing certain actions (such as opening a file). This API supplies the LSM with information about the security contexts of the participating process and the resource being targetted, as well as the code (i.e. a symbollic identifier) for the object class and the action that is about to be performed. The codes for the action and the class of targeted object for a given LSM query are statically embedded into the API – a given hook in the kernel corresponds to a specific action on a certain class of object. LSM returns bearing information on whether, according to the current policy, the process is allowed this action. It's then up to the kernel whether or not to heed this advice. Whether it will, depends on what 'mode' SELinux has been configured for:

- in the **enforcing** mode, the kernel will heed the advice of the LSM and abort any operation that is not permitted;
- in the **permissive** mode, the kernel ignores the LSM's advice, but still makes a log entry for any denials reported – this is very handy for debugging;

- when SELinux is set to **disabled**, the kernel does not query the LSM at all.

Check the current mode by issuing the `sestatus` command:

```
$ sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      28
```

To switch to permissive mode, use the `setenforce` command:

```
$ sudo setenforce 0
$ sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   permissive
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      28
```

To get back to enforcing mode type:

```
$ sudo setenforce 1
$ sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      28
```

The effect of the `setenforce` command is not persistent across reboots. To configure the SELinux mode at boot, one would modify the SELINUX variable in the `/etc/selinux/config` file (this is also where SELinux can be disabled completely).

```
$ cat /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

For now leave the SELINUX variable set to “enforcing”.

The LSM system is not limited to work solely for the kernel. Any application can use its API to ask questions about what is permissible and what is not. In essence, LSM is a security advisor that leaves the enforcement of its policy responses to the kernel or the application making the query.

2.2.2 Security labels

Security labels are strings that describe a security context for a process or a system resource – they are analogous to security badges. From an SELinux standpoint, the operating system consists of a set of agents operating on objects. The agents are the running processes – they are the *sources* of actions. The objects are the directories and files, even other processes, sockets, etc. – they are the *targets* of those actions. Every process and every object on the system is tagged with a security label – the kernel facilitates the tagging of processes, whereas the file system implementations facilitate the tagging of files and directories.

The format of SELinux security label is as follows:
<`seuser`>:<`role`>:<`type`>:<`sensitivity`>:<`categories`>.

The meaning of different parts of the label will be discussed in detail later on. For now let's look at the current labels within part of the filesystem:

```
$ ls -Z
drwxr-xr-x. setest setest unconfined_u:object_r:user_home_t:s0 Desktop
drwxr-xr-x. setest setest unconfined_u:object_r:user_home_t:s0 Documents
drwxr-xr-x. setest setest unconfined_u:object_r:user_home_t:s0 Downloads
drwxr-xr-x. setest setest unconfined_u:object_r:audio_home_t:s0 Music
drwxr-xr-x. setest setest unconfined_u:object_r:user_home_t:s0 Pictures
drwxr-xr-x. setest setest unconfined_u:object_r:user_home_t:s0 Public
drwxr-xr-x. setest setest unconfined_u:object_r:user_home_t:s0 rpmbuild
drwxr-xr-x. setest setest unconfined_u:object_r:user_home_t:s0 Templates
drwxr-xr-x. setest setest unconfined_u:object_r:user_home_t:s0 Videos
```

The `-Z` switch of the `ls` command shows the SELinux labels of the contents of the current directory. In the example above, all but one of the directories are labelled as "unconfined_u:object_r:user_home_t:s0". The security context specified by this label is: `seuser`="unconfined_u", `role`="object_r", `type`="user_home_t", `sensitivity`="s0", with `categories` being undefined. Notice that the DAC flags are still there as well.

To see the context of the running processes, do:

```
$ ps -eZ
LABEL                             PID TTY          TIME CMD
system_u:system_r:init_t:s0         1 ?        00:00:02 systemd
system_u:system_r:kernel_t:s0       2 ?        00:00:00 kthreadd
system_u:system_r:kernel_t:s0       3 ?        00:00:00 ksoftirqd/0
system_u:system_r:kernel_t:s0       5 ?        00:00:00 kworker/0:0H
system_u:system_r:kernel_t:s0       7 ?        00:00:00 kworker/u:0H
.
.
.
```

The context labels are different, but follow the same format.

To see the context of your shell (or the process that runs your terminal) use the `id` command with `-Z` option:

```
$ id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

New directories inherent the labels of their parent directories. Similarly, processes inherit the labels of the process that spawns them. It is possible to change the security labels on files and directories, as well as on a running process, but only if the policy allows it.

2.2.3 Policy

The policy, to which we have referred a number of times already, is the dictionary that defines valid security labels, actions and all the rules dictating what is permissible. It specifies the 'allow' rules only – anything that is not explicitly allowed is not permitted. The policy is is written using the m4 macro language and gets compiled into a binary – it's much faster for the LSM to query the policy in a binary format rather than parsing the text-based representation.

Fedora 20 ships with a precompiled policy called the **targeted policy**. It comes in three flavours: *targeted*, *minimum* and *mls* – in this part of the tutorial we will focus on the one that is enabled by default, which is the *targeted* flavour. The binaries and various support files for the policy reside in the `/etc/selinux/targeted` directory. The policy binary is `/etc/selinux/targeted/policy/policy.<integer>`. The factory reset labels for the entire filesystem are listed in `/etc/selinux/targeted/contexts/files/file_contexts` – this is where the labels come from when SELinux is enabled for the first time.

Policy can be monolithic or modular – depending on how it was written. Modular design makes it easy to augment the policy at runtime with additional rules (modules). Monolithic policy is more strict, and more secure, as any change necessitates recompilation of the entire policy. We will not look at the policy source code at this point – it can be quite overwhelming for a SELinux novice. Note however, that the **targeted policy** is modular. A module is a self-contained sub-policy – it is dependent on the base module and may or may not be dependent on other modules. Some aspects of the policy can only be defined in the base module, which is the first module to loaded. Other modules can be activated or deactivated at will – even at runtime, if there are no interdependencies with other active modules. To see the currently active modules, you can issue the `semodule` command as shown:

```
$ sudo semodule -l
abrt    1.2.0
accountsd       1.0.6
acct    1.5.1
afs     1.8.2
aiccu   1.0.2
aide    1.6.1
ajaxterm        1.0.0
.
.
.
```

In Chapter 4 of this tutorial we will write our own policy module.

2.3 Domains

The essence of the SELinux framework is the division of the Linux system into a set of domains. This is akin to virtualisation, where a number of services get separated and confined to independent virtual machines. The goal of isolation is to provide assurances that a security hole in one service cannot be exploited to gain access to another. While virtualisation is generally highly effective, it is also frequently heavyweight and somewhat rigid, because it isolates the services completely. Isolation of processes and resources into domains with SELinux is achieved by using Type Enforcement (TE), which is flexible and customisable through the policy. Although there are other important aspects of SELinux security (and we will go over them shortly), the domain isolation is the most fundamental.

The policy defines a dictionary of types, object classes, and actions, and a set of rules governing whether agents of certain types are permitted to perform actions on objects of other types. A given type label can tag either the agent or the object of the action. It is the syntax of the allow rule that identifies one type as the source and another as the object of an action. It is even possible to have the same type functioning as both agent and object – for instance, a process attempting to change its SELinux context is the agent and the object of that action.

The `type` is specified by the third token of a security label:
<`seuser`>:<`role`>:<**`type`**>:<`sensitivity`>:<`categories`>.
The label "unconfined_u:object_r:user_home_t:s0" specifies `type`="user_home_t"; the label "system_u:system_r:kernel_t:s0" specifies `type`="kernel_t". To see all the types defined by the currently loaded policy, issue the following command:

```
$ seinfo --type

Types: 4054
   bluetooth_conf_t
   cmirrord_exec_t
   colord_exec_t
   foghorn_exec_t
   jacorb_port_t
   pki_ra_exec_t
   pki_ra_lock_t
   sosreport_t
   etc_runtime_t
   fenced_tmp_t
   git_session_t
   .
   .
   .
```

It's a long list, created with intent of serving a wide range of services commonly offered on Linux distributions. The policy also defines a set of object classes – to see the ones defined by the current policy, issue the following command:

```
$ seinfo --class
Object classes: 83
   netlink_audit_socket
   tcp_socket
   msgq
   x_property
   db_procedure
   dir
   .
   .
   .
```

The "dir" class corresponds to a directory object. The policy also defines a set of actions, but the actions are only meaningful with respect to a given class of an object. To view the valid actions for the "dir" class of objects, issue the following command:

```
$ seinfo --class=dir -x
   dir
      append
      create
      execute
      write
      relabelfrom
      link
      unlink
      ioctl
      getattr
      setattr
      read
      rename
      lock
      relabelto
      mounton
      quotaon
      swapon
      rmdir
      audit_access
      remove_name
      add_name
      reparent
      execmod
      search
      open
```

The allow statements in the policy follow this format:
`allow <source type> <target type>:<object class> {<action>, <action>, ... };`

Recall that any defined type can be the source or the target type. The `allow` statement can be read as follows: a process labelled with the `source type` is allowed to perform the listed `actions` on an `object class` labelled with the `target type`. The curly brackets can be omitted if only one action is specified.

The `source type` is analogous with the concept of a domain, since it is always associated with a process that is attempting to perform some action. For instance, the process that runs your shell, labeled as—

```
$ id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

—can be said to be running in the "unconfined_t" domain. And so the kernel allows the *ls* command to list contest of your home directory (which is of the "user_home_t" type) because there is a rule in the policy to:[1]

```
allow unconfined_t user_home_t: dir { getattr };
```

Hence, using the following allow rules as an example—

```
allow httpd_t httpd_sys_content_t : dir { ioctl read getattr lock search open } ;
allow httpd_t httpd_sys_content_t : file { ioctl read getattr lock open } ;
```

—we can say that a process running in the "httpd_t" domain (that is, a process tagged with the "httpd_t" type) is allowed to read files and directories of "httpd_sys_content_t" type. The default labelling settings of the **targeted policy** make use of this by tagging the *httpd* process with the "httpd_t" type, and in so doing confine it to the "httpd_t" domain. Everything under `/var/www/html` is tagged with the "httpd_sys_content_t" type. As a result, in the default configuration, the web server is capable of reading, but is not allowed to write to the files that provide HTTP content.

To see all the allow rules of the current policy for a specific source type, issue the following command:

```
$ sesearch --allow -d -s httpd_t
Found 765 semantic av rules:
   allow httpd_t jetty_var_lib_t : sock_file { ioctl read write create getattr setattr lock
       relabelfrom relabelto append unlink link rename open } ;
   allow httpd_t jetty_var_lib_t : fifo_file { ioctl read write create getattr setattr lock
       relabelfrom relabelto append unlink link rename open } ;
   allow httpd_t jetty_var_run_t : lnk_file { ioctl read write create getattr setattr lock
       relabelfrom relabelto append unlink link rename } ;
   allow httpd_t jetty_var_run_t : sock_file { ioctl read write create getattr setattr lock
       relabelfrom relabelto append unlink link rename open } ;
   allow httpd_t jetty_var_run_t : fifo_file { ioctl read write create getattr setattr lock
       relabelfrom relabelto append unlink link rename open } ;
   allow httpd_t logrotate_t : process sigchld ;
   allow httpd_t port_type : tcp_socket { recv_msg send_msg } ;
   allow httpd_t port_type : udp_socket { recv_msg send_msg } ;
   .
   .
   .
```

Again, the number of rules is at first rather overwhelming, but from among them, we pick out the following ones:

[1]That rule is specified indirectly through an attribute, but we won't discuss attributes in this chapter.

```
allow httpd_t httpd_sys_rw_content_t : dir { ioctl read write create getattr setattr lock
    unlink link rename add_name remove_name reparent search rmdir open } ;
allow httpd_t httpd_sys_rw_content_t : file { ioctl read write create getattr setattr lock
    append unlink link rename open } ;
```

Thus, any processes running in the "httpd_t" domain is allowed to write to files and directories of "httpd_sys_rw_content_t" type – in the next chapter we will go through the exercise of relabelling selected subdirectories of `/var/www/html` in order to allow the web server to write to them.

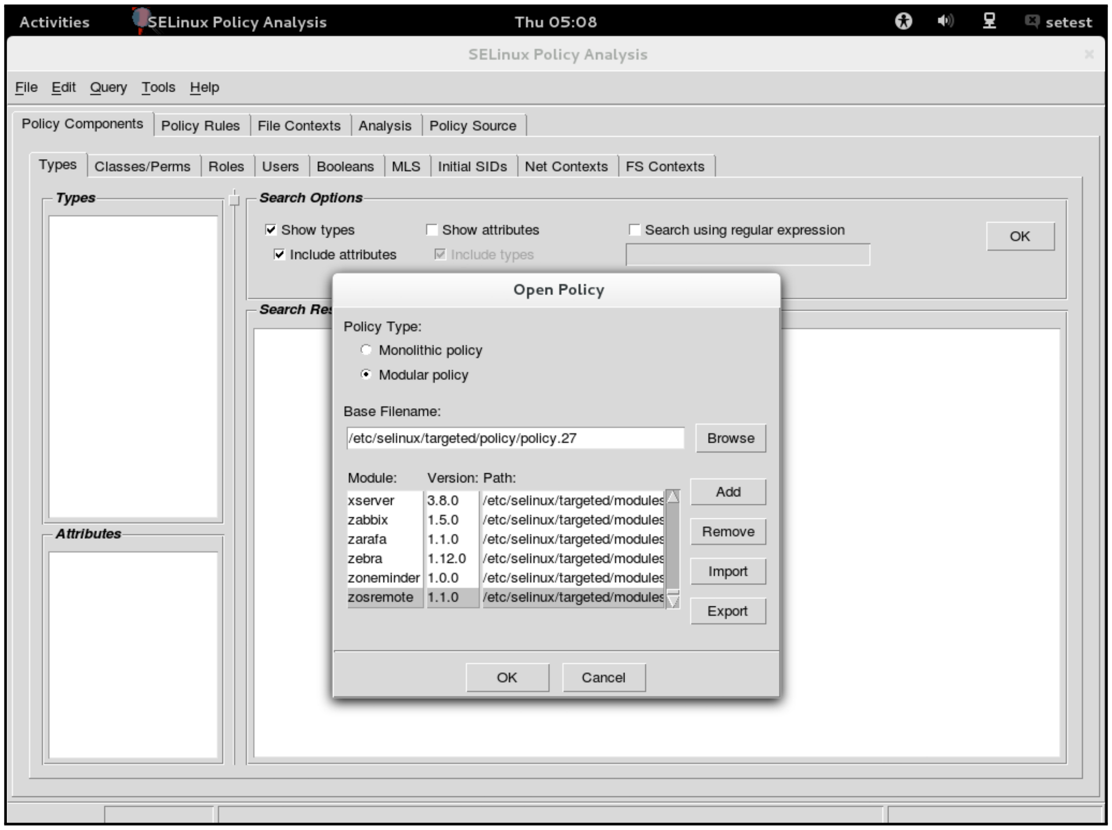
To see all the allow rules with specific source and target types and an object class, issue the following command:

```
$ sesearch --allow -s httpd_t -t httpd_sys_rw_content_t -c dir
Found 5 semantic av rules:
   allow httpd_t httpd_content_type : dir { getattr search open } ;
   allow httpd_t httpdcontent : dir { ioctl read write create getattr setattr lock unlink
       link rename add_name remove_name reparent search rmdir open } ;
   allow httpd_t httpd_content_type : dir { ioctl read getattr lock search open } ;
   allow httpd_t httpd_sys_rw_content_t : dir { ioctl read write create getattr setattr lock
        unlink link rename add_name remove_name reparent search rmdir open } ;
   allow httpd_t httpd_sys_rw_content_t : dir { ioctl read write create getattr setattr lock
        unlink link rename add_name remove_name reparent search rmdir open } ;
```

There is a tool that essentially provides a GUI for the `sesearch` command that can be started by running:

```
$ sudo apol
```

The tool seems to require superuser privileges to access the directory that contains the policy modules. In contrast to `sesearch`, which queries the currently running policy when no policy is specified, the `apol` tool needs to load the policy from the stored binary policy files. Go to File→Open and select the “Modular policy” option. For the base file name browse and select `/etc/selinux/targeted/policy/policy.27`. Then add all (or as many as desired) modules – they are the `*.pp` files in `/etc/selinux/targeted/modules/active/modules`.



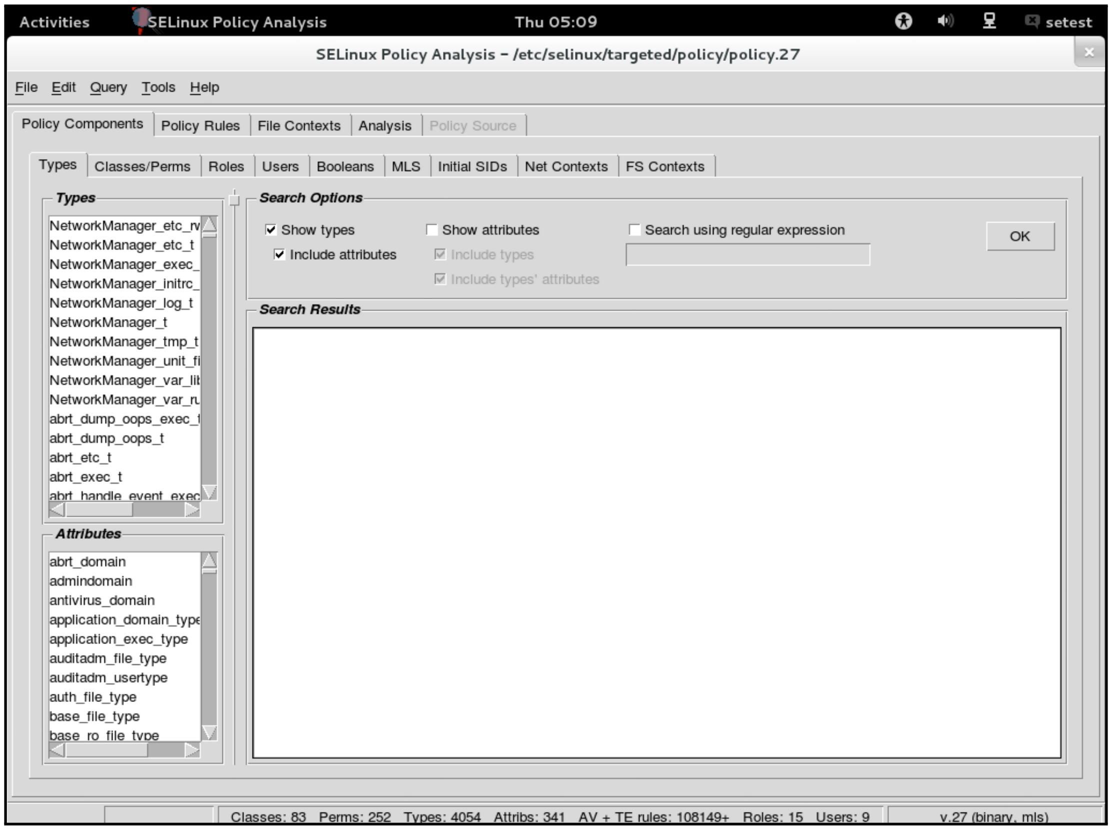
[✓] Show types
[✓] Include attributes
[ ] Show attributes
[✓] Include types
[ ] Search using regular expression

Policy Type:
Monolithic policy
[✓ Modular policy]

Base Filename:
/etc/selinux/targeted/policy/policy.27

Module:	Version:	Path:
xserver	3.8.0	/etc/selinux/targeted/modules
zabbix	1.5.0	/etc/selinux/targeted/modules
zarafa	1.1.0	/etc/selinux/targeted/modules
zebra	1.12.0	/etc/selinux/targeted/modules
zoneminder	1.0.0	/etc/selinux/targeted/modules
zosremote	1.1.0	/etc/selinux/targeted/modules

Press “OK” and the policy should be loaded, so that you can search through and analyse it.



Types:
NetworkManager_etc_rw
NetworkManager_etc_t
NetworkManager_exec_
NetworkManager_initrc_
NetworkManager_log_t
NetworkManager_t
NetworkManager_tmp_t
NetworkManager_unit_fi
NetworkManager_var_lit
NetworkManager_var_ru
abrt_dump_oops_exec_t
abrt_dump_oops_t
abrt_etc_t
abrt_exec_t
abrt_handle_event_exec

Attributes:
abrt_domain
admindomain
antivirus_domain
application_domain_type
application_exec_type
auditadm_file_type
auditadm_usertype
auth_file_type
base_file_type
base_ro_file_type

Search Options:
[✓] Show types
[✓] Include attributes
[ ] Show attributes
[✓] Include types
[✓] Include types' attributes
[ ] Search using regular expression

Classes: 83 Perms: 252 Types: 4054 Attribs: 341 AV + TE rules: 108149+ Roles: 15 Users: 9 | v.27 (binary, mls)

2.3.1 Transitions

Changing the the type in the security label of a process is equivalent to making a transition into another domain. If the new type happened to have exactly the same 'allow' rules in the policy as the old type, the change would not affect anything security-wise: although the process would be technically running in a different domain, it would retain exactly same permissions. However, the purpose of making such label changes is usually to transition into a domain with extended, reduced, or even sometimes altogether different permissions.

When a process spawns another process, the context of the child, by default, is taken from its parent. So, if you run an executable file labelled with the "bin_t" type from a shell, the new process will not be labelled with the "bin_t" type, but instead with the type that your shell is running under. In most cases this is the desired behaviour: that child processes are confined to the domain of their parents. However there are also situations when the child needs to transition in a different domain. For instance, when system starts up, the init process, that spawns various services, is itself confined to the "initrc_t" domain. If all of the services that get started were confined to the parent's domain, this would be suboptimal: the services being started are supposed to be independent, so from a security standpoint it would be risky for them to share a domain – even if they only did so briefly before transitioning to a different domain. Thus, there is a need to be able to specify that certain domain transitions must occur immediately when a process is spawned, for which the policy provides special syntax:

`type_transition` <`source type`> <`target type`> : `process` <`transition type`>;

The rule states that when a process running in the `source type` domain spawns a process from an executable of a `target type`, the new process is to immediately transition into the `transition type`. Let's look at an example.

The binary file `/usr/sbin/crond` is tagged with the following label:

```
$ ls -Z /usr/sbin/crond
-rwxr-xr-x. root root system_u:object_r:crond_exec_t:s0 /usr/sbin/crond
```

That file gives rise to the *crond* processes, which is spawned during the boot sequence by an init script, which is itself confined to the "initrc_t" domain. Let's search the policy for `type_transition` rules with "initrc_t" as the source type and "crond_exec_t" as the target type:

```
$ sesearch -T -s initrc_t -t crond_exec_t
Found 1 semantic te rules:
   type_transition initrc_t crond_exec_t : process crond_t;
```

Note the `-T` switch in the above statement – it restricts the search to only examine `type_-transition` rules. According to the rule that is retrieved above, any process spawned by a binary of "crond_exec_t" type when the parent is in the "initrc_t" domain will transition to the "crond_t" domain. Let's take a look at the label of the *crond* process:

```
$ ps -eZ | grep crond
system_u:system_r:crond_t:s0-s0:c0.c1023    508 ?        00:00:00 ate
system_u:system_r:crond_t:s0-s0:c0.c1023    516 ?        00:00:00 crond
```

Indeed, *crond* is running in the "crond_t" domain.

You are leaving the initrc_t domain
STOP
crond_exec_t
crond_t domain
crond
KS.

At first glance, domain transitions may seem kind of unsecure. However, recall that policy must allow for a specific transition. Typically the domain that a process transitions to is not allowed to transition to any other domain. When policy is set up that way, then transitions are a bit like a process having a one-time use, one-way pass through into a secured area. It's best practice to transition processes on spawn – this way they only ever run in one domain. It is also possible to provision for a transition at an arbitrary point in the process execution. But this more risky, because the process will then spend time, and get to operate within, two different domains.

For more on transitions refer to [7], [8].

2.3.2 Unconfined domain

The processes tagged with the "unconfined_t" type are said to be running in the *unconfined domain*. The philosophy of the **targeted policy** is to put restrictions only on selected services (such as *httpd*). Processes that have to do with direct user access, such as the shell, or X-windows, run in the unconfined domain, which is allowed (in the policy) to do almost everything. Hence the SELinux policy only **targets** certain services, while allowing casual users to utilise their Fedora system for conventional desktop tasks, often without them even realising that SELinux is there, carefully controlling the behaviour of a set of system services.

2.4 Role Based Access Control

SELinux also implements a Role Based Access Control (RBAC) security model, where permissions can vary for different users and the roles that they are able to acquire. Note that

SELinux users and roles are not the same as Linux users and groups. To avoid confusion, in this document we will refer to SELinux users as 'seusers'.

The RBAC model sits on top of TE. The policy defines a set of `seuser`s and `role`s, specifies the `role`s that a given `seuser` can take on, and sets the domains (types) that a given `role` can enter. Figuratively, this means that a given `seuser` is restricted to certain domains. In a literal sense, the policy specifies acceptable combinations of `seuser`, `role` and `type` for valid security labels. Recall that the security label format is:
<**seuser**>:<**role**>:<`type`>:<`sensitivity`>:<`categories`>.
To see the list of `seuser`s defined by the current policy, issue the following command:

```
$ seinfo --user

Users: 9
   sysadm_u
   system_u
   xguest_u
   root
   guest_u
   staff_u
   user_u
   unconfined_u
   git_shell_u
```

To see the list of SELinux `roles` defined by the current policy, issue the following command:

```
$ seinfo --role

Roles: 15
   auditadm_r
   dbadm_r
   guest_r
   staff_r
   user_r
   git_shell_r
   logadm_r
   object_r
   secadm_r
   sysadm_r
   system_r
   webadm_r
   xguest_r
   nx_server_r
   unconfined_r
```

To get the list of `roles` that a given `seuser` is permitted to take on, type:

```
$ seinfo -x --user=sysadm_u
   sysadm_u
      default level: s0
      range: s0 - s0:c0.c1023
      roles:
         object_r
         sysadm_r
```

From the preceeding output, we can see that "sysadm_u" can take on either the "object_r" or "sysadm_r" role. Ignore the range part for now. To see which domains an `seuser` in a given role is allowed to enter, issue the following command:

```
$ seinfo -x --role=sysadm_r
   sysadm_r
      Dominated Roles:
         sysadm_r
      Types:
         git_session_t
         bootloader_t
         netutils_t
         sandbox_x_client_t
         git_user_content_t
         .
         .
         .
```

The "object_r" role is a special role – all system resources take that role – meaning, their security labels specify `role`="object_r". It might seem strange to assign an `seuser` and a `role` to a resource, such as a file, which constitutes a target in some operation. However, SELinux has only one syntax for security labels, and therefore the label syntax must serve the needs of both the acting agents and the objects being acted upon. Thus, the `seuser` and `role` must be specified in the security contexts for both.

Another way to access the `seuser` information, is through the `semanage` command:

```
$ sudo semanage user -l

                Labeling   MLS/       MLS/
SELinux User    Prefix     MCS Level  MCS Range          SELinux Roles

git_shell_u     user       s0         s0                 git_shell_r
guest_u         user       s0         s0                 guest_r
root            user       s0         s0-s0:c0.c1023     staff_r sysadm_r system_r
    unconfined_r
staff_u         user       s0         s0-s0:c0.c1023     staff_r sysadm_r system_r
    unconfined_r
sysadm_u        user       s0         s0-s0:c0.c1023     sysadm_r
system_u        user       s0         s0-s0:c0.c1023     system_r unconfined_r
unconfined_u    user       s0         s0-s0:c0.c1023     system_r unconfined_r
user_u          user       s0         s0                 user_r
xguest_u        user       s0         s0                 xguest_r
```

For the most part, Linux users and SELinux seusers have nothing to do with each other. However, there is a scenario where it does make sense to derive the `seuser` from the system user, namely when a user logs into a terminal or an X-windows display manager. The login shell and X-windows login interface, as is the case for all processes, are labelled with SELinux contexts. It makes sense that a shell running on behalf of a Linux user would derive its security label based on who the user is. Thus for this purpose, there is a mapping between users and `seusers`. To see that mapping, issue the following command:

```
$ sudo semanage login -l

Login Name           SELinux User         MLS/MCS Range        Service

__default__          unconfined_u         s0-s0:c0.c1023       *
root                 unconfined_u         s0-s0:c0.c1023       *
system_u             system_u             s0-s0:c0.c1023       *
```

The default mapping specifies only two system users: `root` and `system_u`, that are mapped to "unconfined_u" and "system_u" "seuser"s respectively. Any other user will have the __*default*__ mapping applied to them, which maps to the "unconfined_u" `seuser`.

2.5 Multi Level Security

Multi-Level Security (MLS) is a third type of security supported by SELinux. Whereas enforcement based on TE and RBAC is always active, MLS enforcement is optional – it can be enabled or disabled at compile time of the policy. MLS security applies a more global type of policy, in that it generally applies the same rules across all the domains. However, it can be fine-tuned to some degree, by incorporating exceptions into the policy that are tied to certain domain types.

MLS in SELinux introduces sensitivity levels and categories to the security context:

- Sensitivity levels can be thought of as levels of security clearance, which can be assigned to processes and resources. These levels are hierarchical, from low to high clearance. A process labelled with a certain sensitivity level will only have read rights to files labelled with the same or lower sensitivity level. However, the same process will not be allowed to write to files of a lower sensitivity: this will ensure that an entity with a high security clearance cannot sharing information through resources that can be read by an entity with a lower clearance. This prevents the unintentional (and intentional) leakage of confidential data through the system.
- Categories, are a convenient way of fine-tuning available permissions while still running in the same domain. Occasionally it is necessary to give different permissions to different instances of the same process. For instance, *https* might need to be ruled by different read/write permissions depending on who is using the website (determined by login credentials, source IP, etc.). It can be a bit of a hassle to need to create different domains for this sort of purpose (although this is exactly what we're going to do in this tutorial). By assigning a set of categories to processes and resources, the read/write access can be controlled based on whether the categories match. Access to a resource is granted only if the categories of that object are a subset of the categories assigned to the process attempting to perform an action on that object.

The targeted flavour of the **targeted policy** does come with MLS enabled (and so does the mls flavour). To see the list of defined sensitivities in the current policy issue the `seinfo` command as follows:

```
$ seinfo --sensitivity

Sensitivities: 1
   s0
```

To list all the `categories`, type:

```
$ seinfo --category

Categories: 1024
   c0
   c1
   .
   .
   .
   c1023
```

Thus, there is only one sensitivity level, "s0", but there are 1024 categories – "c0" to "c1023". The reason why there is only one sensitivity level is that once MLS is enabled, sensitivity must be specified in the security label, despite the fact that targeted flavour of the **targeted policy** is really just interested in using the categories that MLS provides.

When MLS is disabled, the label does not specify `sensitivity` nor `categories`, and so the label format is as follows:

<`seuser`>:<`role`>:<`type`>.

When MLS is enabled, the format is—

<`seuser`>:<`role`>:<`type`>:<`sensitivity`>:<`categories`>,

—but the <`categories`> part can still be omitted (meaning, the labelled process or object in not a member of any category).

The <`sensitivity`> part of the label specifies the lowest and the highest clearance level of the process or object. The policy defines the level hierarchy. When only one sensitivity level is defined, the highest and the lowest clearance are the same and so `sensitivity`="s0-s0". For example, we can have:

```
system_u:system_r:crond_t:s0-s0
```

When the lowest and highest clearance levels are the same, a simpler style of specification is allowed: `sensitivity`="s0". Hence, the label below specifies exactly same security context as the one above:

```
system_u:system_r:crond_t:s0
```

For the categories, an arbitrary combination of categories can be specified, or none at all (as in the examples above). To specify only one category, `categories`="c0". For example:

```
system_u:system_r:crond_t:s0-s0:c0
```

Multiple categories are comma separated, such as `categories`="c0,c3". This example label specifies two categories, "c0" and "c3":

```
system_u:system_r:crond_t:s0-s0:c0,c3
```

A range of categories is specified as follows: `categories`="c2.c5" – this example specifies 4 categories, equivalent to `categories`="c2,c3,c4,c5". We have already seen labels that specify all categories, for example:

```
system_u:system_r:crond_t:s0-s0:c0.c1023
```

Note the difference between a security context that doesn't have a category and the one that specifies all categories: the former will be permitted to work with objects that also do not have a category, whereas the latter can work with any combination of categories (including when none are specified).

2.6 Summary

SELinux is a security framework that brings MAC to Linux. This framework labels processes and the filesystem with labels that are used to identify the security context of the agents and targets of various actions. The permissions for processes of a given security context to operate on resources within another security context are conveyed through the policy. The system is not forced on the kernel, but functions as a consulting service and is also available to applications through the SELinux API. It supports three types of security enforcement: TE, RBAC and MLS – with MLS enforcement being optional.

In programming it is often the case that mastering a programming language requires becoming proficient with use of its standard library in addition to learning the language syntax. Similarly with SELinux, one needs a good grasp of the default policy in order to effectively utilise the SELinux framework. In the next two chapters of this tutorial, we will demonstrate how to work with the existing policy, and how to modify it in order to take advantage of the TE security.

3 Case study: DokuWiki on SELinux

In this chapter we will go through the process of installing and securing DokuWiki (`https://www.dokuwiki.org`), which is a wiki web server that uses files for its back-end document storage. This is a good starting point for dealing with the existing SELinux policy, because DokuWiki is relatively simple, yet still requires additional privileges to that of a static website. The objective of this exercise is not just to provide the correct SELinux configuration to make the service work, but also to guide the reader through the process of troubleshooting the problems that may result from denials caused by misconfigured SELinux security.

3.1 Installation

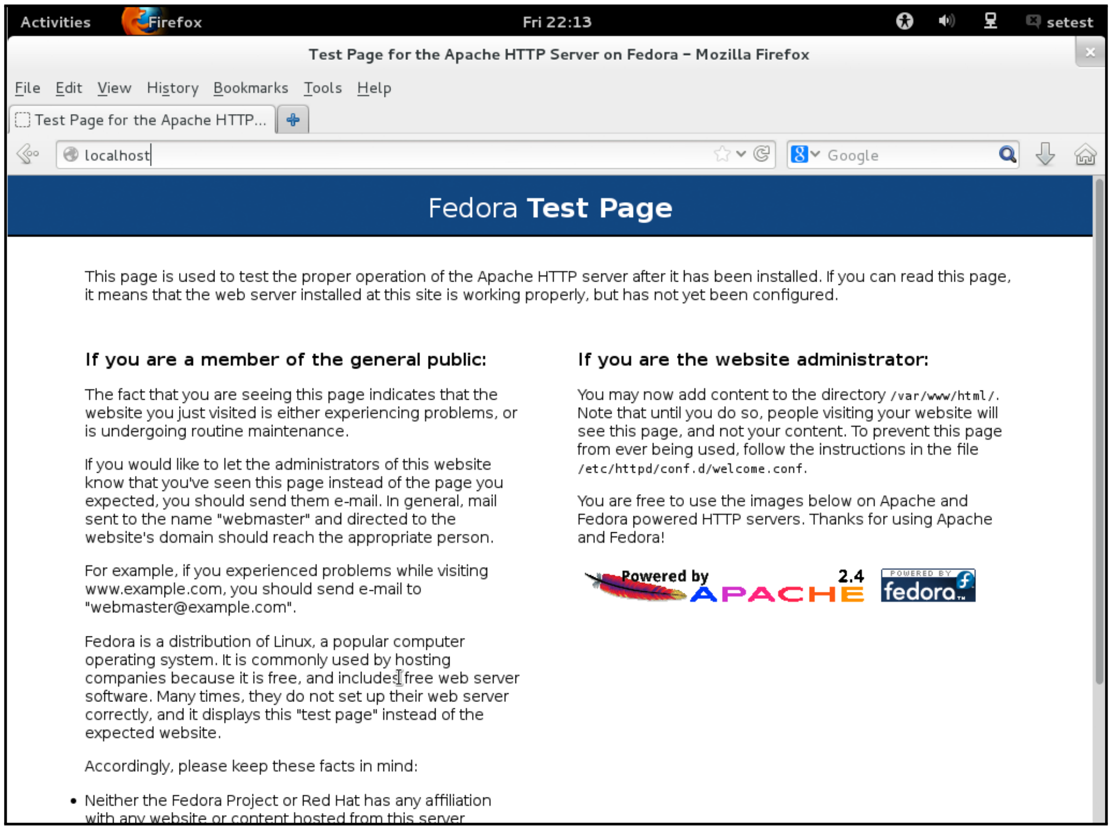
For this exercise we need to install the Apache server with PHP support:

```
$ sudo yum install httpd php
```

Enable and start the *httpd* service:

```
$ sudo systemctl start httpd.service
$ sudo systemctl enable httpd.service
```

Test whether Apache is running – from your web browser of choice, try to access *localhost*. You should see the Apache test page:


Activities
Firefox
Fri 22:13
setest
Test Page for the Apache HTTP Server on Fedora – Mozilla Firefox
File Edit View History Bookmarks Tools Help
Test Page for the Apache HTTP...
localhost
Google

Fedora **Test Page**

This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page, it means that the web server installed at this site is working properly, but has not yet been configured.

If you are a member of the general public:

The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

For example, if you experienced problems while visiting www.example.com, you should send e-mail to "webmaster@example.com".

Fedora is a distribution of Linux, a popular computer operating system. It is commonly used by hosting companies because it is free, and includes free web server software. Many times, they do not set up their web server correctly, and it displays this "test page" instead of the expected website.

Accordingly, please keep these facts in mind:

- Neither the Fedora Project or Red Hat has any affiliation with any website or content hosted from this server.

If you are the website administrator:

You may now add content to the directory `/var/www/html/`. Note that until you do so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the file `/etc/httpd/conf.d/welcome.conf`.

You are free to use the images below on Apache and Fedora powered HTTP servers. Thanks for using Apache and Fedora!

This is optional, but if you want the web pages to be accessible from other machines, you need to open port 80 in the Linux firewall:

```
$ sudo firewall-cmd --zone=public --add-port=80/tcp
$ sudo firewall-cmd --zone=public --permanent --add-port=80/tcp
```

3.2 DokuWiki setup

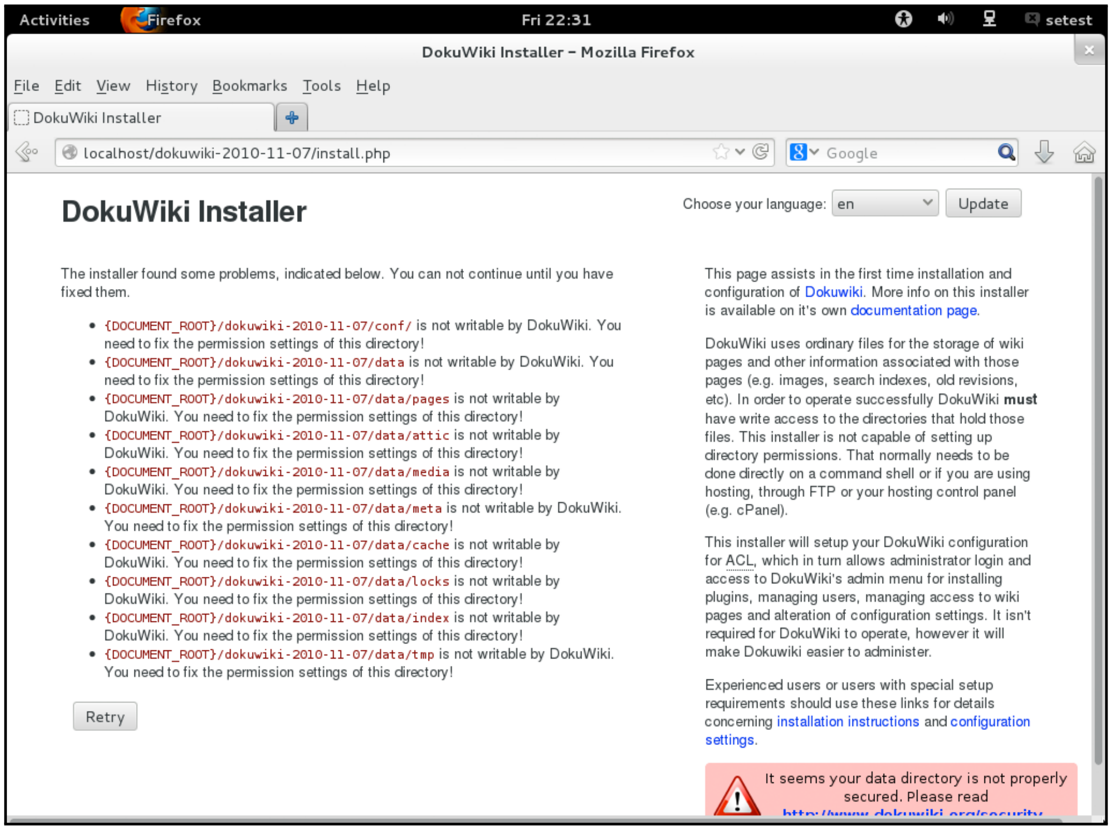
Download DokuWiki Anteater – we want this specific (quite old now) version, because it has some vulnerabilities which will be later used to showcase SELinux in action.

```
$ wget http://www.splitbrain.org/_media/projects/dokuwiki/dokuwiki-2010-11-07.tgz
```

Unpack it into the Apache's root folder for the HTML and PHP content:

```
$ sudo tar -zxvf dokuwiki-2010-11-07.tgz -C /var/www/html
```

The remainder of the install needs to be done from the web browser. Go to `http://localhost/dokuwiki-2010-11-07/install.php` in your browser. You should see the following page:


Activities
Firefox
Fri 22:31
setest
DokuWiki Installer – Mozilla Firefox
File Edit View History Bookmarks Tools Help
DokuWiki Installer
localhost/dokuwiki-2010-11-07/install.php
Google
Choose your language: en
Update
Retry

DokuWiki Installer

The installer found some problems, indicated below. You can not continue until you have fixed them.

- `{DOCUMENT_ROOT}/dokuwiki-2010-11-07/conf/` is not writable by DokuWiki. You need to fix the permission settings of this directory!
- `{DOCUMENT_ROOT}/dokuwiki-2010-11-07/data` is not writable by DokuWiki. You need to fix the permission settings of this directory!
- `{DOCUMENT_ROOT}/dokuwiki-2010-11-07/data/pages` is not writable by DokuWiki. You need to fix the permission settings of this directory!
- `{DOCUMENT_ROOT}/dokuwiki-2010-11-07/data/attic` is not writable by DokuWiki. You need to fix the permission settings of this directory!
- `{DOCUMENT_ROOT}/dokuwiki-2010-11-07/data/media` is not writable by DokuWiki. You need to fix the permission settings of this directory!
- `{DOCUMENT_ROOT}/dokuwiki-2010-11-07/data/meta` is not writable by DokuWiki. You need to fix the permission settings of this directory!
- `{DOCUMENT_ROOT}/dokuwiki-2010-11-07/data/cache` is not writable by DokuWiki. You need to fix the permission settings of this directory!
- `{DOCUMENT_ROOT}/dokuwiki-2010-11-07/data/locks` is not writable by DokuWiki. You need to fix the permission settings of this directory!
- `{DOCUMENT_ROOT}/dokuwiki-2010-11-07/data/index` is not writable by DokuWiki. You need to fix the permission settings of this directory!
- `{DOCUMENT_ROOT}/dokuwiki-2010-11-07/data/tmp` is not writable by DokuWiki. You need to fix the permission settings of this directory!

This page assists in the first time installation and configuration of Dokuwiki. More info on this installer is available on it's own documentation page.

DokuWiki uses ordinary files for the storage of wiki pages and other information associated with those pages (e.g. images, search indexes, old revisions, etc). In order to operate successfully DokuWiki **must** have write access to the directories that hold those files. This installer is not capable of setting up directory permissions. That normally needs to be done directly on a command shell or if you are using hosting, through FTP or your hosting control panel (e.g. cPanel).

This installer will setup your DokuWiki configuration for ACL, which in turn allows administrator login and access to DokuWiki's admin menu for installing plugins, managing users, managing access to wiki pages and alteration of configuration settings. It isn't required for DokuWiki to operate, however it will make Dokuwiki easier to administer.

Experienced users or users with special setup requirements should use these links for details concerning installation instructions and configuration settings.

It seems your data directory is not properly secured. Please read

There's a number of errors, all of which relate to the fact that by default, the files and subdirectories inside `/var/www/html` are read only. DokuWiki requires write permissions to a couple of subdirectories. Recall that there are two security systems that have to be taken into account: the normal filesystem DAC permissions and SELinux. Usually, it's best to sort them out one at a time, and so let's begin by switching SELinux into permissive mode. In permissive mode SELinux is still there, being consulted about permissions, but the kernel does not heed the LSM's advice.

```
$ sudo setenforce 0
```

If you refresh the page the errors should be still there. Since SELinux is being ignored, this means that DAC permissions are not in order.

3.2.1 DAC permissions

Let us take a look at the DokuWiki directory in detail:

```
$ ls -l /var/www/html/
total 4
drwxr-xr-x. 7 setest 102 4096 Nov  7  2010 dokuwiki-2010-11-07
```

User `setest` is the username under which DokuWiki was installed on our system. We'll first change DAC permissions to assign `dokuwiki-2010-11-07` to the `apache` group – a group that *httpd* is a member of.

```
$ sudo chown -R setest:apache /var/www/html/dokuwiki-2010-11-07
$ ls -l /var/www/html/
total 4
drwxr-xr-x. 7 setest apache 4096 Nov  7  2010 dokuwiki-2010-11-07
```

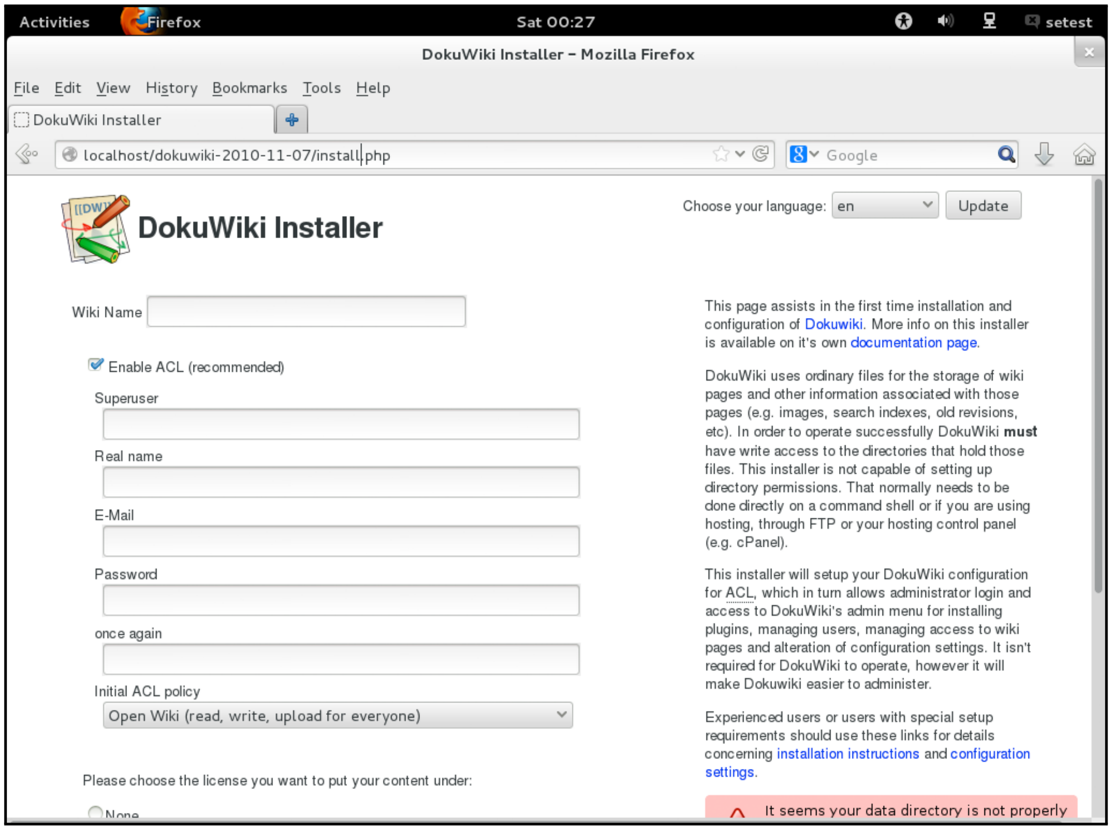
From the error messages in the browser it looks like *httpd* also requires write permissions to the `conf` and `data` subdirectories. Let's examine the current permissions:

```
$ ls -l /var/www/html/dokuwiki-2010-11-07/
total 88
drwxr-xr-x.  2 setest apache  4096 Nov  7  2010 bin
drwxr-xr-x.  2 setest apache  4096 Nov  7  2010 conf
-rw-r--r--.  1 setest apache 17992 Nov  7  2010 COPYING
drwxr-xr-x. 10 setest apache  4096 Nov  7  2010 data
-rw-r--r--.  1 setest apache  2185 Nov  7  2010 doku.php
-rw-r--r--.  1 setest apache 11730 Nov  7  2010 feed.php
drwxr-xr-x.  6 setest apache  4096 Nov  7  2010 inc
-rw-r--r--.  1 setest apache   182 Nov  7  2010 index.php
-rw-r--r--.  1 setest apache 17428 Nov  7  2010 install.php
drwxr-xr-x.  9 setest apache  4096 Nov  7  2010 lib
-rw-r--r--.  1 setest apache   306 Nov  7  2010 README
-rw-r--r--.  1 setest apache    22 Nov  7  2010 VERSION
```

At this point DAC allows only the owner, `setest`, to write to any of these directories. Let's change the group permissions so that members of the `apache` group are allowed to write as well:

```
$ chmod -R g+w /var/www/html/dokuwiki-2010-11-07/conf
$ chmod -R g+w /var/www/html/dokuwiki-2010-11-07/data
$ ls -l /var/www/html/dokuwiki-2010-11-07/
total 88
drwxr-xr-x.  2 setest apache  4096 Nov  7  2010 bin
drwxrwxr-x.  2 setest apache  4096 Nov  7  2010 conf
-rw-r--r--.  1 setest apache 17992 Nov  7  2010 COPYING
drwxrwxr-x. 10 setest apache  4096 Nov  7  2010 data
-rw-r--r--.  1 setest apache  2185 Nov  7  2010 doku.php
-rw-r--r--.  1 setest apache 11730 Nov  7  2010 feed.php
drwxr-xr-x.  6 setest apache  4096 Nov  7  2010 inc
-rw-r--r--.  1 setest apache   182 Nov  7  2010 index.php
-rw-r--r--.  1 setest apache 17428 Nov  7  2010 install.php
drwxr-xr-x.  9 setest apache  4096 Nov  7  2010 lib
-rw-r--r--.  1 setest apache   306 Nov  7  2010 README
-rw-r--r--.  1 setest apache    22 Nov  7  2010 VERSION
```

Refresh the page `http://localhost/dokuwiki-2010-11-07/install.php`, and you should get:


Activities
Firefox
Sat 00:27
setest
DokuWiki Installer – Mozilla Firefox
File Edit View History Bookmarks Tools Help
DokuWiki Installer
localhost/dokuwiki-2010-11-07/install.php
Google
Choose your language: en
Update

DokuWiki Installer

Wiki Name

✓ Enable ACL (recommended)

Superuser

Real name

E-Mail

Password

once again

Initial ACL policy

Open Wiki (read, write, upload for everyone)

Please choose the license you want to put your content under:

None

This page assists in the first time installation and configuration of Dokuwiki. More info on this installer is available on it's own documentation page.

DokuWiki uses ordinary files for the storage of wiki pages and other information associated with those pages (e.g. images, search indexes, old revisions, etc). In order to operate successfully DokuWiki **must** have write access to the directories that hold those files. This installer is not capable of setting up directory permissions. That normally needs to be done directly on a command shell or if you are using hosting, through FTP or your hosting control panel (e.g. cPanel).

This installer will setup your DokuWiki configuration for ACL, which in turn allows administrator login and access to DokuWiki's admin menu for installing plugins, managing users, managing access to wiki pages and alteration of configuration settings. It isn't required for DokuWiki to operate, however it will make Dokuwiki easier to administer.

Experienced users or users with special setup requirements should use these links for details concerning installation instructions and configuration settings.

It seems your data directory is not properly

There is a warning about the data directory not being properly secured, but other than that, it seems that this is going to be sufficient to get DokuWiki going. That warning is about remote users' ability to directly access the contents of DokuWiki's directories through Apache. This can be secured by appropriate .htaccess configuration, but for this tutorial we will not worry about this.

3.2.2 SELinux permissions

Now that the DAC permission are in order, let's turn our attention to SELinux. We're not going to change the SELinux mode just yet. First, refresh DokuWiki's install page in the browser and take a look at /var/log/audit/audit.log. Near the end of the file there should be an AVC message like this (there might be other AVC messages as well):

```
type=AVC msg=audit(1367648942.891:501): avc:  denied  { write } for  pid=4380 comm="httpd"
    name="conf" dev="dm-1" ino=164380 scontext=system_u:system_r:httpd_t:s0 tcontext=
    unconfined_u:object_r:httpd_sys_content_t:s0 tclass=dir
```

AVC messages record SELinux denials. In permissive mode, these denials don't impact the system's operation, but still get logged. Let's turn enforcing mode back on just to confirm that these messages have to do with access through DokuWiki.

```
$ sudo setenforce 1
```

Refresh the install page one more time. The errors, about the `conf` and `data` directories not being writable, should be back. In fact, the AVC messages spells out what the problem is – the source context of type "httpd_t" has no write permissions to directory of type "httpd_sys_content_t". The AVC message also indicates that the problem has to do with *httpd* process, and we can confirm this, by listing all of the processes that are running in the "httpd_t" domain:

```
$ ps -eZ | grep httpd_t
system_u:system_r:httpd_t:s0     2115 ?        00:00:06 httpd
system_u:system_r:httpd_t:s0     2116 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0     2117 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0     2118 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0     2119 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0     2120 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0     2189 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0     2193 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0     2194 ?        00:00:00 httpd
system_u:system_r:httpd_t:s0     2379 ?        00:00:00 httpd
```

We can also take a look at the SELinux labels on the files in DokuWiki's root directory:

```
$ ls -Z /var/www/html/dokuwiki-2010-11-07
drwxr-xr-x. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 bin
drwxrwxr-x. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 conf
-rw-r--r--. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 COPYING
drwxrwxr-x. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 data
-rw-r--r--. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 doku.php
-rw-r--r--. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 feed.php
drwxr-xr-x. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 inc
-rw-r--r--. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 index.php
-rw-r--r--. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 install.php
drwxr-xr-x. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 lib
-rw-r--r--. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 README
-rw-r--r--. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 VERSION
```

The type for all the files and subdirectories is "httpd_sys_content_t". Just out of curiosity, let's search the policy for rules that involve "httpd_t" as the source type and "httpd_sys_-content_t" as the target type:

```
$ sesearch -d --allow -s httpd_t -t httpd_sys_content_t -C
Found 4 semantic av rules:
   allow httpd_t httpd_sys_content_t : file { ioctl read getattr lock open } ;
   allow httpd_t httpd_sys_content_t : dir { ioctl read getattr lock search open } ;
   allow httpd_t httpd_sys_content_t : lnk_file { read getattr } ;
DT allow httpd_t httpd_sys_content_t : dir { ioctl read write getattr lock add_name
    remove_name search open } ; [ httpd_enable_cgi httpd_unified && httpd_builtin_scripting
    && ]
```

The `-C` option adds information about which of the rules are conditional. Conditional rules, marked with "DT", can be turned on and off using specially provided boolean variables (the variables involved and the condition for the rule to turn on follow after the rule itself). The "D" in "DT" stands for "disabled". The reason why we chose to reveal the conditional rules is to point out that despite there being a rule that does "allow httpd_t httpd_sys_content_t : dir write ", such a rule may not be active. As it turns out, the default SELinux settings mirror the DAC permissions in that the policy does not allow the "httpd_t" domain (to which *httpd* is confined) to write to `/var/www/html`, or any of its subdirectories. Enabling that conditional rule to solve our configuration problem is not a good idea at this point – after all, we do not want to allow "httpd_t" to write to all DokuWiki files and subdirectories. Instead, we will relabel the two subdirectories that must be able to be written to, with the type that "httpd_t" has the write permissions for.

The policy already defines a type intended for *httpd* writable content – the "httpd_sys_-rw_content_t" type.

```
$ sesearch -d --allow -s httpd_t -t httpd_sys_rw_content_t -C
Found 7 semantic av rules:
ET allow httpd_t httpd_sys_rw_content_t : file { ioctl read write create getattr setattr
    lock append unlink link rename open } ; [ httpd_builtin_scripting ]
DT allow httpd_t httpd_sys_rw_content_t : file { ioctl read write create getattr setattr
    lock append unlink link rename open } ; [ httpd_enable_cgi httpd_unified &&
    httpd_builtin_scripting && ]
ET allow httpd_t httpd_sys_rw_content_t : dir { ioctl read write create getattr setattr lock
     unlink link rename add_name remove_name reparent search rmdir open } ; [
    httpd_builtin_scripting ]
DT allow httpd_t httpd_sys_rw_content_t : dir { ioctl read write create getattr setattr lock
     unlink link rename add_name remove_name reparent search rmdir open } ; [
    httpd_enable_cgi httpd_unified && httpd_builtin_scripting && ]
ET allow httpd_t httpd_sys_rw_content_t : lnk_file { ioctl read write create getattr setattr
     lock append unlink link rename } ; [ httpd_builtin_scripting ]
DT allow httpd_t httpd_sys_rw_content_t : lnk_file { ioctl read write create getattr setattr
     lock append unlink link rename } ; [ httpd_enable_cgi httpd_unified &&
    httpd_builtin_scripting && ]
ET allow httpd_t httpd_sys_rw_content_t : sock_file { read write getattr append open } ; [
    httpd_builtin_scripting ]
```

All of these rules are conditional, but the "E" in the "ET" prefix stands for "enabled", and so, to make this a bit more clear, let us filter out all but the enabled rules:

```
$ sesearch -d --allow -s httpd_t -t httpd_sys_rw_content_t -C | grep ET
ET allow httpd_t httpd_sys_rw_content_t : file { ioctl read write create getattr setattr
    lock append unlink link rename open } ; [ httpd_builtin_scripting ]
ET allow httpd_t httpd_sys_rw_content_t : dir { ioctl read write create getattr setattr lock
     unlink link rename add_name remove_name reparent search rmdir open } ; [
    httpd_builtin_scripting ]
ET allow httpd_t httpd_sys_rw_content_t : lnk_file { ioctl read write create getattr setattr
     lock append unlink link rename } ; [ httpd_builtin_scripting ]
ET allow httpd_t httpd_sys_rw_content_t : sock_file { read write getattr append open } ; [
    httpd_builtin_scripting ]
```

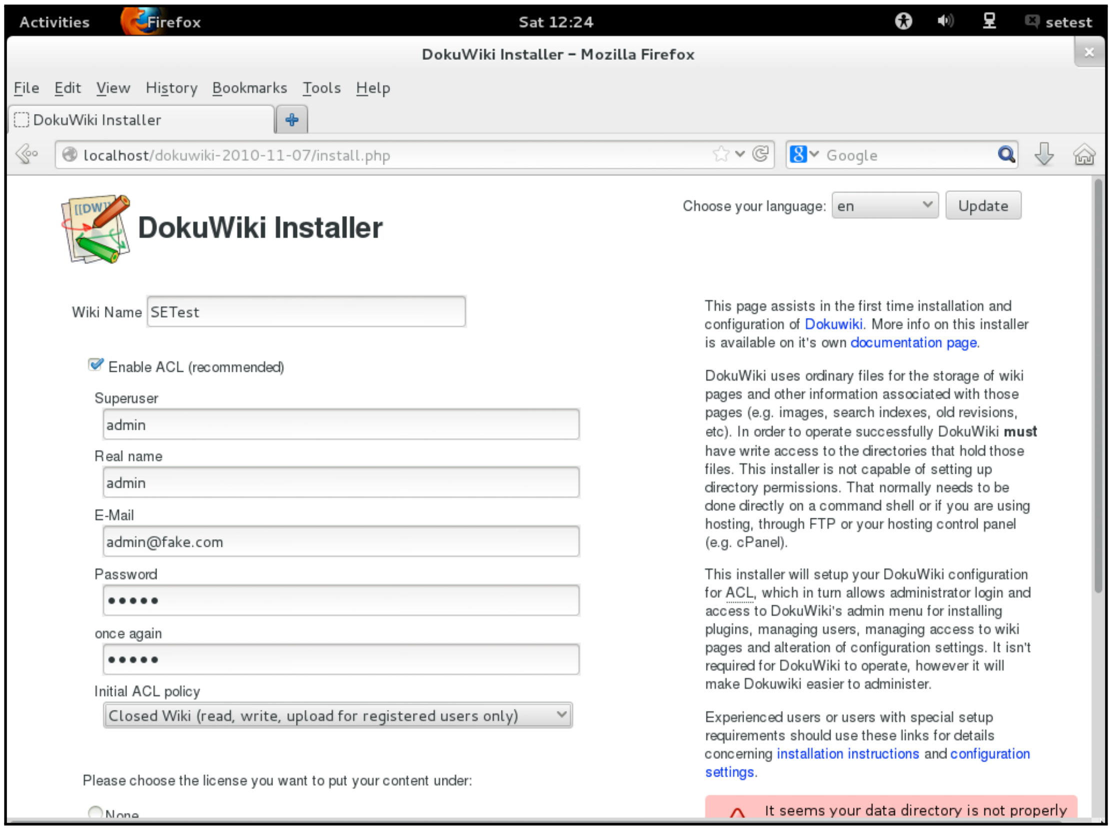
It should be clear from the above set of rules that the policy does allow writing from "httpd_-t" domain to files and directories of "httpd_sys_rw_context_t" type. To fix our configuration problem, we just need to relabel the two DokuWiki subdirectories with the"httpd_-sys_rw_context_t" type. Here's the command that will achieve this:

```
$ sudo chcon -t httpd_sys_rw_content_t /var/www/html/dokuwiki-2010-11-07/conf
$ sudo chcon -R -t httpd_sys_rw_content_t /var/www/html/dokuwiki-2010-11-07/data
```

The `-R` option does the relabelling recursively, so that all the internal contents of the directory get relabelled as well. Check that the new directories have been relabelled:

```
$ ls -Z /var/www/html/dokuwiki-2010-11-07/
drwxr-xr-x. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 bin
drwxrwxr-x. setest apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 conf
-rw-r--r--. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 COPYING
drwxrwxr-x. setest apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 data
-rw-r--r--. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 doku.php
-rw-r--r--. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 feed.php
drwxr-xr-x. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 inc
-rw-r--r--. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 index.php
-rw-r--r--. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 install.php
drwxr-xr-x. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 lib
-rw-r--r--. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 README
-rw-r--r--. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 VERSION
```

Refresh the install page in your web browser, and there should be no errors now. Proceed with the DokuWiki setup as instructed on the page. Give your wiki a name, enable ACL, create superuser account and make sure to select the "Closed Wiki" ACL policy. When you are finished, press the **Save** button.


Activities
Firefox
Sat 12:24
setest
DokuWiki Installer – Mozilla Firefox
File Edit View History Bookmarks Tools Help
DokuWiki Installer
localhost/dokuwiki-2010-11-07/install.php
Google
DokuWiki Installer
Choose your language: en
Update

Wiki Name SETest

[✓] Enable ACL (recommended)

Superuser: admin

Real name: admin

E-Mail: admin@fake.com

Password: •••••

once again: •••••

Initial ACL policy: Closed Wiki (read, write, upload for registered users only)

Please choose the license you want to put your content under:

None

This page assists in the first time installation and configuration of Dokuwiki. More info on this installer is available on it's own documentation page.

DokuWiki uses ordinary files for the storage of wiki pages and other information associated with those pages (e.g. images, search indexes, old revisions, etc). In order to operate successfully DokuWiki **must** have write access to the directories that hold those files. This installer is not capable of setting up directory permissions. That normally needs to be done directly on a command shell or if you are using hosting, through FTP or your hosting control panel (e.g. cPanel).

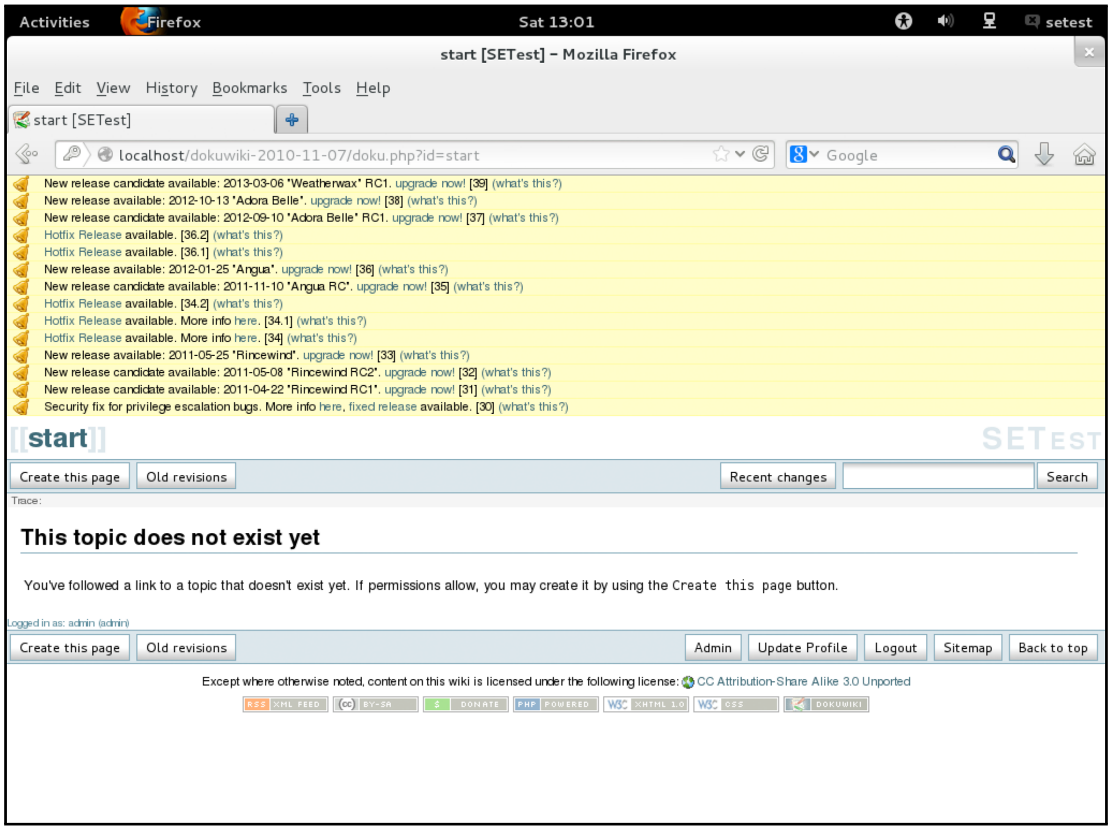
This installer will setup your DokuWiki configuration for ACL, which in turn allows administrator login and access to DokuWiki's admin menu for installing plugins, managing users, managing access to wiki pages and alteration of configuration settings. It isn't required for DokuWiki to operate, however it will make Dokuwiki easier to administer.

Experienced users or users with special setup requirements should use these links for details concerning installation instructions and configuration settings.

It seems your data directory is not properly

3.3 Working DokuWiki

You have now a functioning DokuWiki with both DAC and SELinux allowing *httpd* to write only to the `conf` and `data` subdirectories under `/var/www/html/dokuwiki-2010-11-07`. If you login to DokuWiki as an `admin` you will see a number of security warnings – this is to inform you that a newer, more secure, version of DokuWiki is available. Do not upgrade – we are using this vulnerable version of DokuWiki on purpose in order to demonstrate SELinux in action – we'll come to that demonstration in the next chapter.


Activities
Firefox
Sat 13:01
setest
start [SETest] – Mozilla Firefox
File Edit View History Bookmarks Tools Help
start [SETest]
localhost/dokuwiki-2010-11-07/doku.php?id=start
Google

New release candidate available: 2013-03-06 "Weatherwax" RC1. upgrade now! [39] (what's this?)
New release available: 2012-10-13 "Adora Belle". upgrade now! [38] (what's this?)
New release candidate available: 2012-09-10 "Adora Belle" RC1. upgrade now! [37] (what's this?)
Hotfix Release available. [36.2] (what's this?)
Hotfix Release available. [36.1] (what's this?)
New release available: 2012-01-25 "Angua". upgrade now! [36] (what's this?)
New release candidate available: 2011-11-10 "Angua RC". upgrade now! [35] (what's this?)
Hotfix Release available. [34.2] (what's this?)
Hotfix Release available. More info here. [34.1] (what's this?)
Hotfix Release available. More info here. [34] (what's this?)
New release available: 2011-05-25 "Rincewind". upgrade now! [33] (what's this?)
New release candidate available: 2011-05-08 "Rincewind RC2". upgrade now! [32] (what's this?)
New release candidate available: 2011-04-22 "Rincewind RC1". upgrade now! [31] (what's this?)
Security fix for privilege escalation bugs. More info here, fixed release available. [30] (what's this?)

This topic does not exist yet

You've followed a link to a topic that doesn't exist yet. If permissions allow, you may create it by using the `Create this page` button.

Except where otherwise noted, content on this wiki is licensed under the following license: CC Attribution-Share Alike 3.0 Unported

3.4 Changing the factory reset security contexts

In the previous section we have changed the security context of DokuWiki subdirectories using the `chcon` command. This method of relabelling is persistent and so the labels we changed will remain in their changed state even after a reboot. However, the policy specifies the *factory reset* contexts for all files and directories – changes made with `chcon` are not incorporated into that setting. The factory reset on SELinux labels is handy when SELinux is enabled for the first time, or the policy is switched (the new policy might have completely different labelling scheme), Hence, if one wished to incorporate the label changes into the factory reset labelling, here's how this can be done for the two subdirectories we have relabelled above:

```
$ sudo semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/dokuwiki-2010-11-07/
    conf'
$ sudo semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/dokuwiki-2010-11-07/
    data(/.*)?'
```

This command assures that the type in the security labels for the two directories is switched to "httpd_sys_rw_content_t" during factory reset context restoration. The syntax at the end of the second command is a regular expression that assure all the subdirectories are relabelled as well. The command won't work if you use an invalid combination of SELinux seuser, role and type. You can check whether the changes have made it to factory reset configuration as follows:

```
$ cat /etc/selinux/targeted/contexts/files/file_contexts.local
# This file is auto-generated by libsemanage
# Do not edit directly.

/var/www/html/dokuwiki-2010-11-07/conf     system_u:object_r:httpd_sys_rw_content_t:s0
/var/www/html/dokuwiki-2010-11-07/data(/.*)?    system_u:object_r:httpd_sys_rw_content_t:s0
```

The `semanage` command just changes the factory reset configuration, it doesn't actually relabel the directories. In our case the directories have been already relabelled with the `chcon` command, but after the changes with `semanage` it's probably best to apply the new labels by sourcing them from the factory reset settings. For the two directories in question, this can be done with the following commands:

```
$ restorecon -v '/var/www/html/dokuwiki-2010-11-07/conf'
$ restorecon -v -R '/var/www/html/dokuwiki-2010-11-07/data'
```

If after the `restorecon` the security contexts on those directories have switched back to `type`="httpd_sys_content_t", then the changes made to the factory reset settings have not worked as intended. If the labels list `type`="httpd_sys_rw_content_t", then the factory reset context settings have been successfully changed to work as expected.

We should also mention here, that if there is a need to restore factory reset context on all files, create an `.autorelabel` file in the system root directory and reboot:

```
$ touch /.autorelabel
$ reboot
```

On boot up, when `.autorelabel` is found in the root directory, everything gets relabelled with the factory reset contexts before Linux starts-up (`.autorelabel` gets deleted automatically afterwards, so there is not relabelling on the following reboot).

3.5 Discussion

Let's pause for a moment and think how seemingly easy it was to change the SELinux permissions by relabelling the files. It's worth noting that, although we did have to elevate Linux privileges to `sudo` in order to execute the `chcon` command, that was to overcome the system's DAC permissions, not the SELinux permissions for relabelling. However, behind the scenes, SELinux did in fact check whether we were allowed to relabel those contexts. To see why this all worked without a hitch let's chase down the relevant permissions within the policy.

When a process (our shell) attempts to change security label of a file, the kernel consults the LSM regarding whether the policy allows the "relabelfrom" action on the file's current type and "relabelto" action to the new type. Let's check what domain the shell is running in:

```
$ id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

The shell is running in the "unconfined_t" domain. Let's examine the policy whether "unconfined_t" is allowed to perform "relabelfrom" on a "file" that is of "httpd_sys_-content_t" type. While we're at it, let's also check for a similar rule, except with the "relabelto" action and "httpd_sys_rw_content_t" target type.

```
$ sesearch --allow -s unconfined_t -t httpd_sys_content_t -p relabelfrom -c file
Found 1 semantic av rules:
   allow files_unconfined_type file_type : file { ioctl read write create getattr setattr
       lock relabelfrom relabelto append unlink link rename execute swapon quotaon mounton
       execute_no_trans entrypoint open audit_access } ;

$ sesearch --allow -s unconfined_t -t httpd_sys_rw_content_t -p relabelto -c file
Found 1 semantic av rules:
   allow files_unconfined_type file_type : file { ioctl read write create getattr setattr
       lock relabelfrom relabelto append unlink link rename execute swapon quotaon mounton
       execute_no_trans entrypoint open audit_access } ;
```

Strangely, the search returns with the same rule in both cases showing different types to those specified in the query: "files_unconfined_type" for the source and "file_type" for the target type. These new types are in fact attributes. An attribute is a feature of SELinux policy that allows grouping of several types under one name. In cases where there are sets of types that need the same permissions, it's convenient to group them under an attribute and write one allow rule that will apply to all the types. To make things a bit easier for reading, by convention, types are named with a "_t" suffix and attributes with a "_type" suffix. The `sesearch` command resolves the relationship between attributes and types and displays the policy rule(s) that allow the operation that's been specified in the command arguments. It seems that "files_unconfined_type" is an attribute of "httpd_sys_content_-t" and "httpd_sys_rw_content_t", and "file_type" is an attribute of "unconfined_t". Let's take a look at the types grouped under "file_type" (there's a lot, so we narrow down the results with grep):

```
$ seinfo --attribute=file_type -x | grep httpd_sys
      httpd_sys_content_t
      httpd_sys_htaccess_t
      httpd_sys_ra_content_t
      httpd_sys_rw_content_t
      httpd_sys_script_exec_t
```

Indeed both "httpd_sys_content_t" and "httpd_sys_rw_content_t" are there. We can also check the attributes for a given type. Let's do that with the "unconfined_t" type (again, we restrict the output with grep):

```
$ seinfo --type=unconfined_t -x | grep files
      files_unconfined_type
      filesystem_unconfined_type
```

As expected, "unconfined_t" has an attribute "files_unconfined_type". If we were to check the allow rules against the "dir" class of objects, we would get rules similar to the ones we have observed for "file" object. The reason why we were able to make those context changes is because our shell is running in a domain that is allowed to do this particular relabelling. In fact, processes running in the "unconfined_t" domain are allowed to do quite a lot under the default policy. However, a process running in the "httpd_t" domain, would not be able to make such context changes. You can check that policy has no allow rules for:

```
$ sesearch --allow -s httpd_t -t httpd_sys_content_t -p relabelfrom -c file

$ sesearch --allow -s httpd_t -t httpd_sys_rw_content_t -p relabelto -c file
```

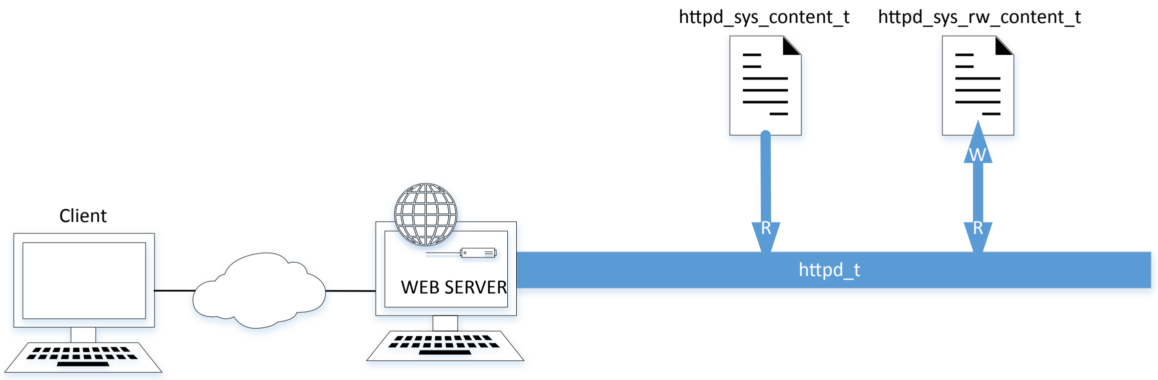


Figure 3.1: Apache server running in the "httpd_t" domain.

3.6 Summary

In this chapter we have set up a file-based DokuWiki service that required dynamic web content. Using the labels predefined in the default policy, we modified security contexts on selected directories in order to allow Apache to write to the directories that store DokuWiki's content. Figure 3.5 show a diagram of the current SELinux setup – the *httpd* process runs in the "httpd_t" domain, which has read-only permissions for files and directories of type "httpd_sys_content_t" and has read/write permissions to files and directories of type "httpd_sys_rw_content_t".

4 Case study: DokuWiki with double hull security

In the previous chapter we worked with existing SELinux policy to configure a DokuWiki web service. In this chapter we will augment the existing policy to create a security ring around the web service, which mirrors DokuWiki's own security system. This type of configuration is often referred to as *double hull* security, because it maintains two independent security rings around a service. The appeal of double hull security is that it is unlikely that the same vulnerability will be found in both systems, making it less likely that an attacker would be able to penetrate both layers of protection. This is especially relevant to DokuWiki, because it's an open source software system that allows users to extend standard services with custom-built plugins. It is possible for these plugins to interact with (and even bypass) the DokuWiki access control system, and so there is the potential for bugs to arise that will compromise security. We will demonstrate this scenario by using an old version of DokuWiki with an old version of a plugin that is known to create such a security hole.

4.1 Installation

In the current configuration SELinux Apache (that is the *httpd* process) always runs in "httpd_t" domain. For the double hull setup we will need to run an *httpd* fork for each client's request, potentially in different SELinux domains. We will also need to access the SELinux API from PHP.

4.1.1 SELinux module for Apache

Install the `mod_selinux` module, which provides SELinux support for Apache:

```
$ sudo yum install mod_selinux
```

Documentation on how the module operates can be found at [9]. This module ensures that each HTTP request is served by a fork of *httpd* with an SELinux context assigned according to the rules given in the configuration file `/etc/httpd/conf.d/mod_selinux.conf`. This allows for different clients' requests to be served, by worker processes that run in different SELinux domains. There are numerous options for configuring the SELinux contexts for clients: they can be based on requested URL, the client's IP or even HTTP authentication. In this tutorial we will trigger context transitions from PHP, and thus will leave the configuration file as it is, which should have only the following (not commented out) option:

```
selinuxServerDomain     *:s0
```

This option specifies that a fork of *httpd* stays in the same domain as the parent, setting the MLS part of the context `sensitivity`="s0".

4.1.2 SELinux API for PHP

To enable SELinux API calls from PHP we require a installation of an additional package from the PECL collection, found at `http://pecl.php.net/package/selinux`. Here's how to install the latest (at the time of writing) version of the package:

```
$ sudo yum install php-devel libselinux-devel gcc
$ wget http://pecl.php.net/get/selinux-0.3.1.tgz
$ tar -zxvf selinux-0.3.1.tgz
$ cd selinux-0.3.1
$ phpize
$ ./configure
$ make
$ sudo make install
$ echo "extension=selinux.so"| sudo tee /etc/php.d/selinux.ini
```

In order for the new changes to take effect, restart the *httpd* service:

```
$ sudo systemctl restart httpd.service
```

The list of SELinux functions available from PHP is listed at [10].

4.1.3 SELinux Template

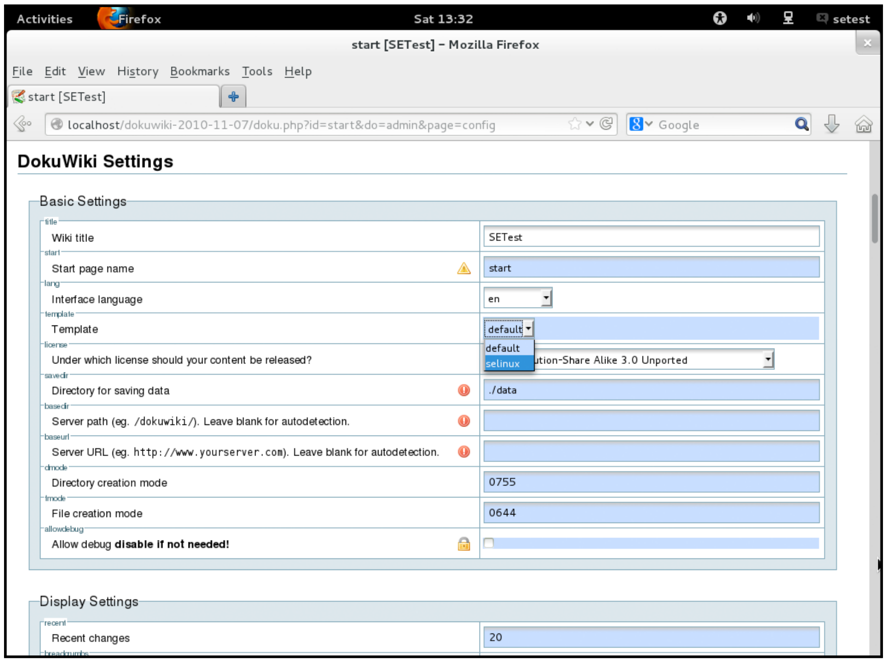
Having access to the SELinux API from PHP allows us to install a template that displays SELinux configuration, as well as the context of the serving process, directly into DokuWiki's pages. This will provide some convenient visibility into the SELinux state, and is very useful for this tutorial (though unlikely to be a good idea in a production deployment!). Download the file `dokuwiki_tpl_selinux.tgz` and unpack it into the DokuWiki's templates directory:

```
$ wget http://www.cs.otago.ac.nz/space/selinuxtutorial/archives/dokuwiki_tpl_selinux.tgz
$ tar -zxvf dokuwiki_tpl_selinux.tgz -C /var/www/html/dokuwiki-2010-11-07/lib/tpl
```

There should now be a `selinux` subdirectory in DokuWiki's templates directory:

```
$ ls -Z /var/www/html/dokuwiki-2010-11-07/lib/tpl/
drwxr-xr-x. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 default
-rw-r--r--. setest apache unconfined_u:object_r:httpd_sys_content_t:s0 index.php
drwxrwxr-x. setest setest unconfined_u:object_r:httpd_sys_content_t:s0 selinux
```

In your browser, when logged-in to DokuWiki as an `admin`, go to the *Admin* page and click on *Configuration Settings*. Under *Basic Settings* there should be now an option for an **selinux** *Template*. Select it and **Save**.


Activities
Firefox
Sat 13:32
setest
start [SETest] – Mozilla Firefox
File Edit View History Bookmarks Tools Help
start [SETest]
localhost/dokuwiki-2010-11-07/doku.php?id=start&do=admin&page=config
Google

DokuWiki Settings

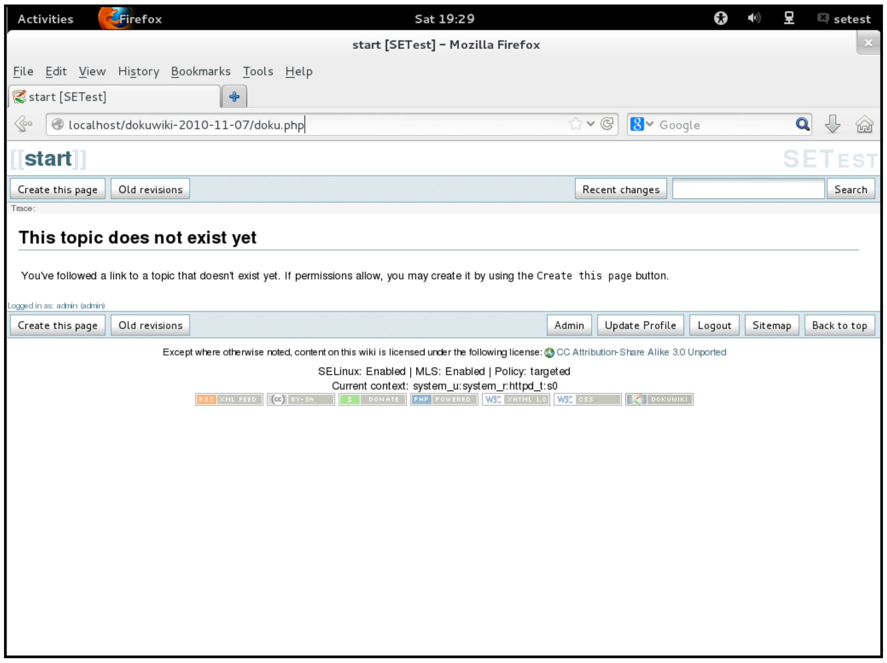
Basic Settings

title: Wiki title	SETest
start: Start page name	start
lang: Interface language	en
template: Template	default (default / selinux)
license: Under which license should your content be released?	ution-Share Alike 3.0 Unported
savedir: Directory for saving data	./data
basedir: Server path (eg. /dokuwiki/). Leave blank for autodetection.	
baseurl: Server URL (eg. http://www.yourserver.com). Leave blank for autodetection.	
dmode: Directory creation mode	0755
fmode: File creation mode	0644
allowdebug: Allow debug **disable if not needed!**	☐

Display Settings

recent: Recent changes	20

The new template will provide SELinux information in the footer of every DokuWiki page. Go to the main page (`http://localhost/dokuwiki-2010-11-07/doku.php`) and you should see the usual start page, but now augmented with the SELinux information footer like this:



This topic does not exist yet

You've followed a link to a topic that doesn't exist yet. If permissions allow, you may create it by using the `Create this page` button.

Except where otherwise noted, content on this wiki is licensed under the following license: CC Attribution-Share Alike 3.0 Unported

SELinux: Enabled | MLS: Enabled | Policy: targeted
Current context: system_u:system_r:httpd_t:s0

The information in the footer tells you whether SELinux is enabled or not, whether or not the optional MLS/MCS policy is enabled, the current policy name, and the SELinux context of the process that fetched this page (which at this point should be of `type`="httpd_t").

The template that we just installed is simply a copy of the default DokuWiki template with a few extra function calls added that fetch the required SELinux information. If you are curious about what the PHP code looks like that calls SELinux, you can examine the contents of the `/var/www/html/dokuwiki-2010-11-07/lib/tpl/selinux/footer.html` file. The function that gets the security information is listed below:

File : footer.html

```
function selinux_info() {
    echo "<div>SELinux: ";
    $selinuxen = selinux_is_enabled();
    if($selinuxen) {
        $selinuxenforce = selinux_getenforce();
        $selinuxerror = false;
        if($selinuxenforce==0) {
            echo "Permissive";
        } elseif($selinuxenforce==1) {
            echo "Enabled";
        } else {
           echo "Error";
           $selinuxerror = true;
        }
        if(!$selinuxerror) {
           echo " | MLS: ";
           if(selinux_mls_is_enabled()) {
               echo "Enabled";
           } else {
               echo "Disabled";
           }
           echo " | Policy: " . selinux_getpolicytype();
           echo "</div><div>Current context: " . selinux_getcon();
        }
    } else {
        echo "Disabled";
    }
    echo "</div>";
}
```

Note that the `selinux_getcon` function fetches the SELinux context of the process executing the PHP script.

4.2 Pages and namespaces

While still logged-in to DokuWiki as an `admin`, go to the main page, `http://localhost/dokuwiki-2010-11-07/doku.php`, and select **Create this page** (if you haven't created it yet). Add the following text:

```
Welcome to dokuwki SEtest.  Take a look at these links:

- Link to [[devel:pageX|pageReadWrite]] in devel namespace

- Link to [[dev.l:pageY|pageReadOnly]] in dev.l namespace
```

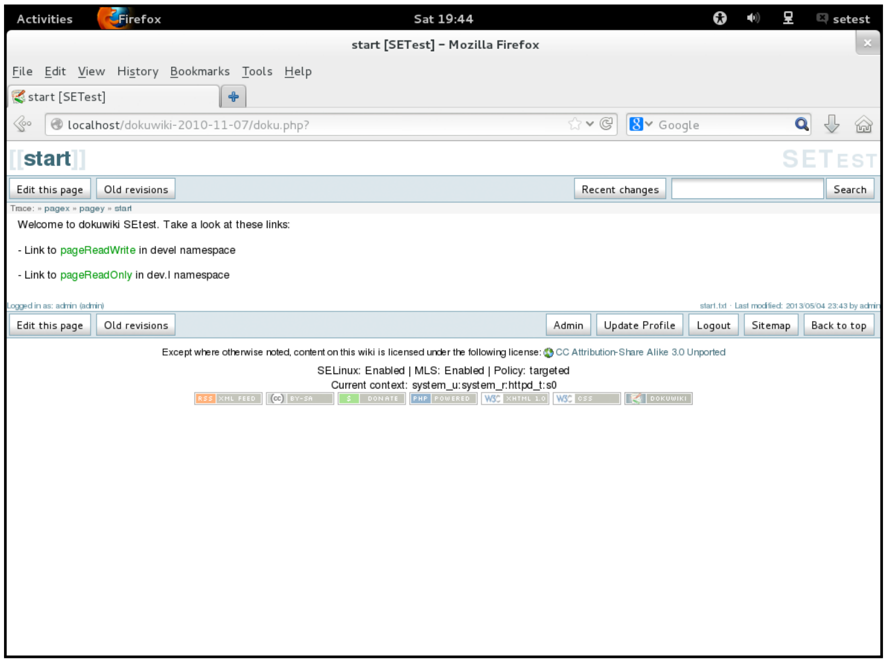
Press **Save**. The start page should now contain the above text, and include two links. Click on **pageReadWrite**. This will take you to *pageX* in DokuWiki's **devel** namespace, which does not exist yet. Click on **Create this page** and paste in the following text:

```
This page is in devel namespace.  It should be ReadWrite for developer group.
```

Press **Save**. Go back to `http://localhost/dokuwiki-2010-11-07/doku.php` and click the second link, **pageReadOnly**. This takes you to *pageY* in the **dev.l** namespace. Click **Create this page** and paste in the following text:

```
This page is in dev.l namespace.  It should be ReadOnly for developer group.
```

Press **Save**. If you go back to `http://localhost/dokuwiki-2010-11-07/doku.php` you should have a page that looks like this:


Activities
Firefox
Sat 19:44
setest
start [SETest] – Mozilla Firefox
File Edit View History Bookmarks Tools Help
start [SETest]
localhost/dokuwiki-2010-11-07/doku.php?
Google
[[start]]
SETEST
Edit this page
Old revisions
Recent changes
Search

Trace: » pagex » pagey » start

Welcome to dokuwiki SEtest. Take a look at these links:

- Link to pageReadWrite in devel namespace

- Link to pageReadOnly in dev.l namespace

Logged in as: admin (admin)

start.txt · Last modified: 2013/05/04 23:43 by admin

Except where otherwise noted, content on this wiki is licensed under the following license: CC Attribution-Share Alike 3.0 Unported

SELinux: Enabled | MLS: Enabled | Policy: targeted
Current context: system_u:system_r:httpd_t:s0

Now you have three pages, *start*, *PageX* and *PageY* in DokuWiki's *root*, *devel* and *dev.l* namespaces respectively.

Next, we will create a DokuWiki user that has read and write permissions in the *devel* namespace and read only permissions in the *dev.l* space.

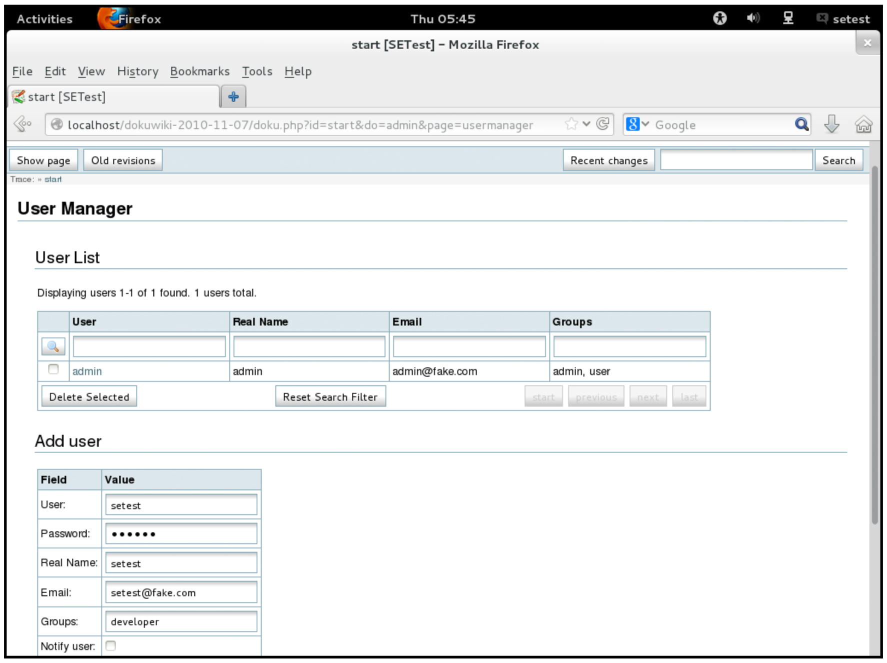
4.3 DokuWiki ACL

DokuWiki maintains pages organised into namespaces. On the server these namespaces correspond to different directories. DokuWiki's Access Control Lists (ACL) [11] assigns users to groups, and allows the setting of group permissions for different namespaces. The possible permissions are (listed from the least to most privileged):

- none—not allowed to access DokuWiki content (will only see the login page)
- read—allowed to read data
- edit—allowed to modify existing pages
- create—allowed to create new pages
- upload—allowed to upload media files
- delete—allowed to delete media files
- admin—allowed to administer the site

These permissions are hierarchical – a given level allows everything that all of the lower permission levels do; a group is assigned to only one permission level and the privileges from levels below it are implied.

Let's create a new DokuWiki user. While logged-in to DokuWiki as an **admin**, go to the **User Manager** section from the **Admin** page. Create a new user: `setest`. Fill in all the necessary information and write `developer` in the *Group* box – since that group doesn't exist yet, DokuWiki will create it.



start [SETest] – Mozilla Firefox

localhost/dokuwiki-2010-11-07/doku.php?id=start&do=admin&page=usermanager

User Manager

User List

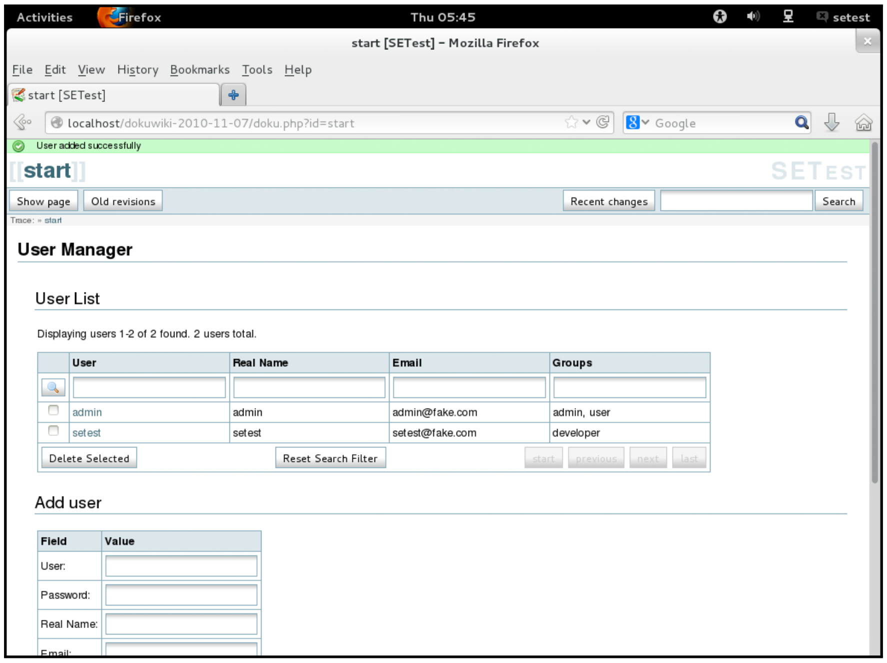
Displaying users 1-1 of 1 found. 1 users total.

User	Real Name	Email	Groups
admin	admin	admin@fake.com	admin, user

Add user

Field	Value
User:	setest
Password:	••••••
Real Name:	setest
Email:	setest@fake.com
Groups:	developer
Notify user:	☐

Press the **Add** button at the bottom of the page. The new user should now appear on the *User List*.



start [SETest] – Mozilla Firefox

localhost/dokuwiki-2010-11-07/doku.php?id=start

User added successfully

User Manager

User List

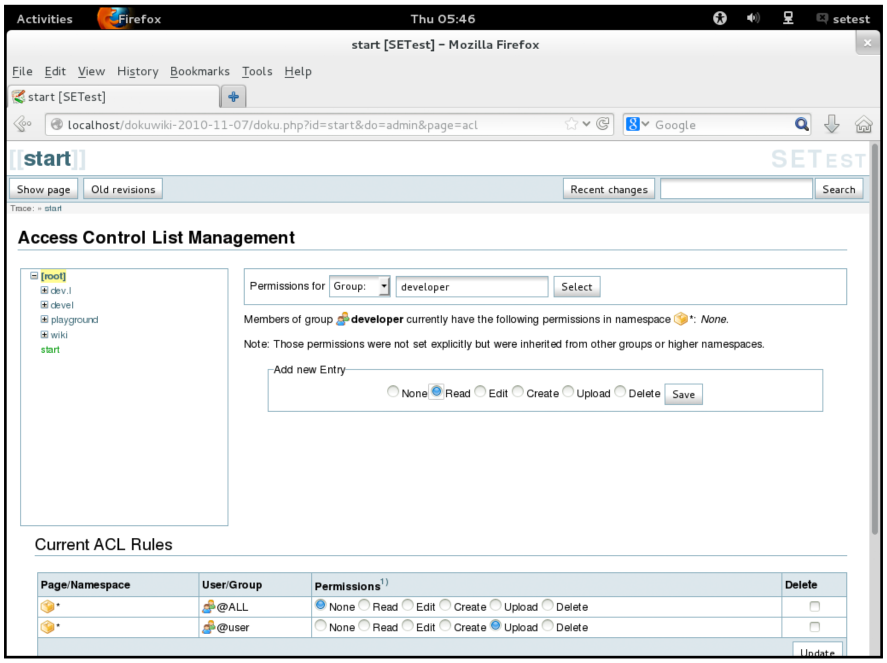
Displaying users 1-2 of 2 found. 2 users total.

User	Real Name	Email	Groups
admin	admin	admin@fake.com	admin, user
setest	setest	setest@fake.com	developer

Add user

Field	Value
User:	
Password:	
Real Name:	
Email:	

Next, we need to specify ACL permissions for the `developer` group. Go to the **Access Control List Management** section from the **Admin** page. Select `[root]` namespace on the lefthand side and in the **Permissions for Group:** type in `developer` and click Select. In the **Add new Entry** windows select **Read**.



Activities Firefox Thu 05:46 setest

start [SETest] - Mozilla Firefox

localhost/dokuwiki-2010-11-07/doku.php?id=start&do=admin&page=acl

Access Control List Management

[root] dev.l devel playground wiki start

Permissions for Group: developer Select

Members of group **developer** currently have the following permissions in namespace *: *None.*

Note: Those permissions were not set explicitly but were inherited from other groups or higher namespaces.

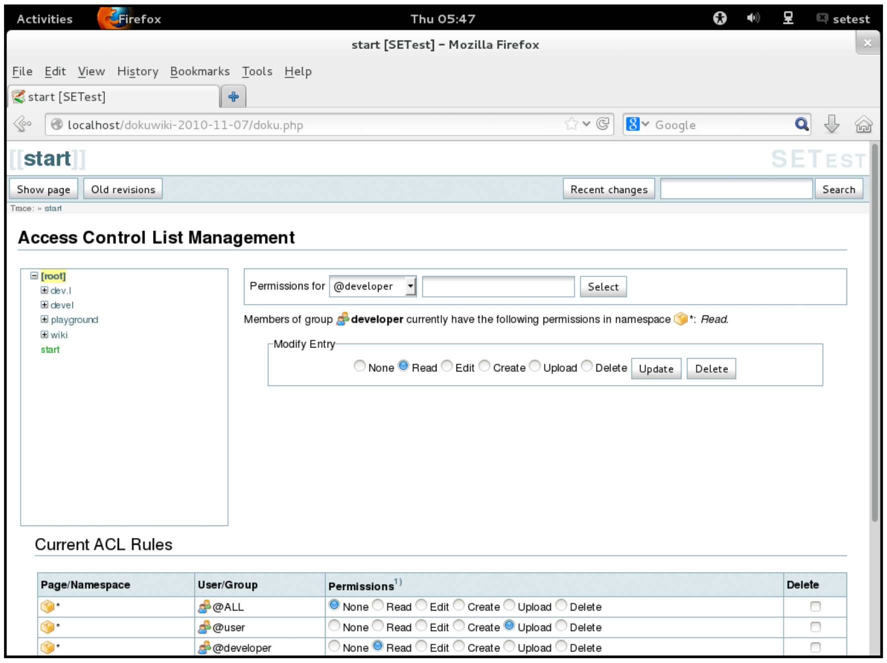
Add new Entry

None [✓ Read] Edit Create Upload Delete Save

Current ACL Rules

Page/Namespace	User/Group	Permissions[1]	Delete
*	@ALL	[✓ None] Read Edit Create Upload Delete	☐
*	@user	None Read Edit Create [✓ Upload] Delete	☐

Click **Save**. A new line should appear at the bottom of the **Current ACL Rules** specifying that `developer` group has read permissions to the *root* namespace (designated with the symbol *).



Activities Firefox Thu 05:47 setest

start [SETest] - Mozilla Firefox

localhost/dokuwiki-2010-11-07/doku.php

Access Control List Management

[root] dev.l devel playground wiki start

Permissions for @developer Select

Members of group **developer** currently have the following permissions in namespace *: *Read.*

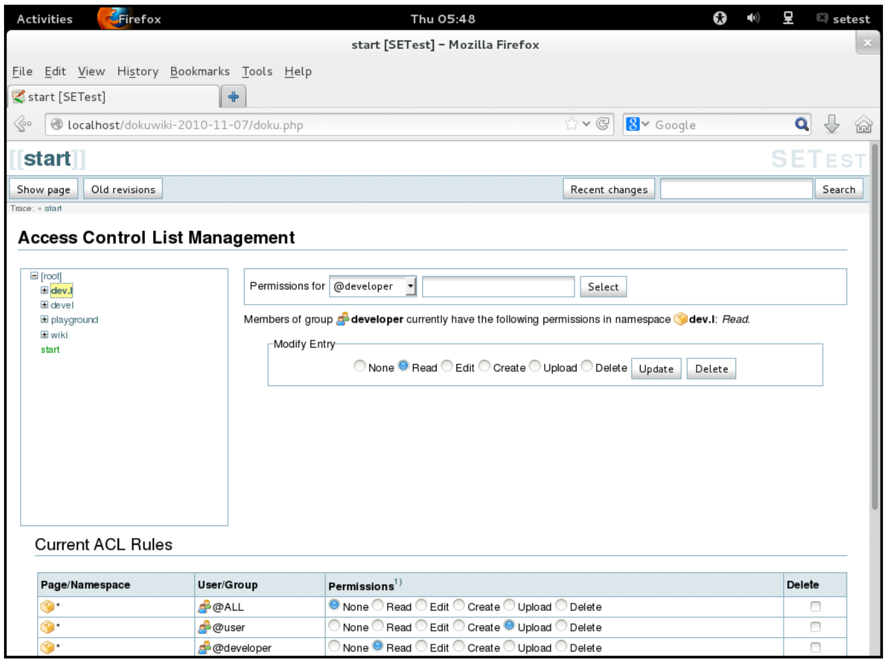
Modify Entry

None [✓ Read] Edit Create Upload Delete Update Delete

Current ACL Rules

Page/Namespace	User/Group	Permissions[1]	Delete
*	@ALL	[✓ None] Read Edit Create Upload Delete	☐
*	@user	None Read Edit Create [✓ Upload] Delete	☐
*	@developer	None [✓ Read] Edit Create Upload Delete	☐

Now select **dev.l** namespace in the left-hand window. The **Permissions for** should now specify `@developer` group, so no need to type anything there. Make sure that **Read** is selected in the **Add new Entry** window and press **Save**.



Access Control List Management

[root]
dev.l
devel
playground
wiki
start

Permissions for @developer Select

Members of group developer currently have the following permissions in namespace dev.l: *Read.*

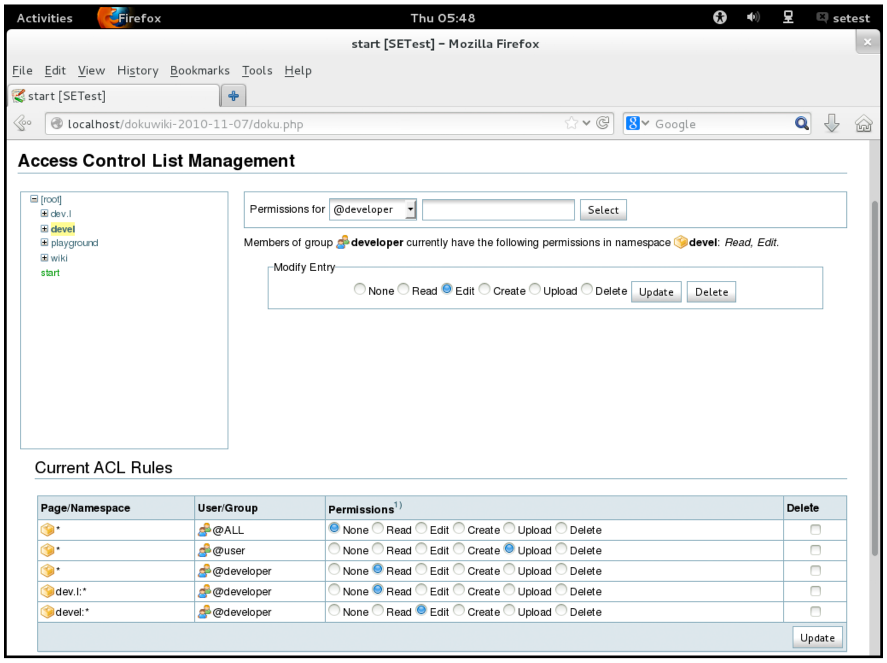
Modify Entry

None [✓ Read] Edit Create Upload Delete Update Delete

Current ACL Rules

Page/Namespace	User/Group	Permissions[1]	Delete
*	@ALL	[✓ None] Read Edit Create Upload Delete	☐
*	@user	None Read Edit Create [✓ Upload] Delete	☐
*	@developer	None [✓ Read] Edit Create Upload Delete	☐

Finally, select **devel** namespace in the left window, give the **developer** group **Edit** permissions. Press **Save**. The **Current ACL Rules** should look like the rules in the diagram below:


Activities
Firefox
Thu 05:48
setest
start [SETest] – Mozilla Firefox
File Edit View History Bookmarks Tools Help
start [SETest]
localhost/dokuwiki-2010-11-07/doku.php
Google

Access Control List Management

[root]
dev.l
devel
playground
wiki
start

Permissions for @developer Select

Members of group developer currently have the following permissions in namespace devel: *Read, Edit.*

Modify Entry

None Read [✓ Edit] Create Upload Delete Update Delete

Current ACL Rules

Page/Namespace	User/Group	Permissions[1]	Delete
*	@ALL	[✓ None] Read Edit Create Upload Delete	☐
*	@user	None Read Edit Create [✓ Upload] Delete	☐
*	@developer	None [✓ Read] Edit Create Upload Delete	☐
dev.l:*	@developer	None [✓ Read] Edit Create Upload Delete	☐
devel:*	@developer	None Read [✓ Edit] Create Upload Delete	☐

Update

Table 4.1 shows the permissions of the `developer` group for different namespaces:

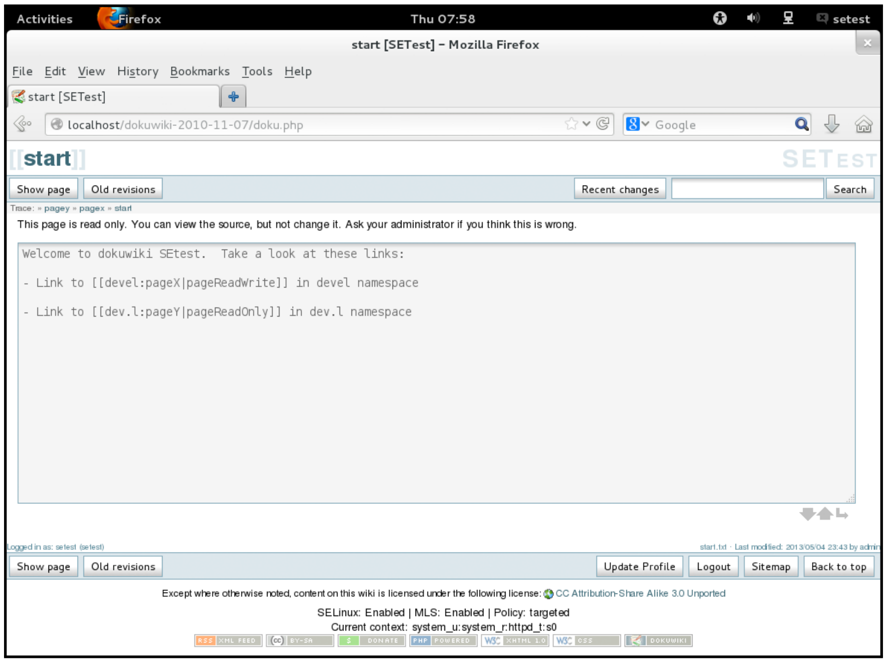
Table 4.1: Group permissions for DokuWiki namespaces

Namespace \ Group	`ALL`	`developer`	`admin`
root	None	Read	Upload
dev.l	None	Read	Upload
devel	None	Edit	Upload

Note that group `ALL` corresponds to users that are not logged in, and that the group `admin` is not shown in the **Current ACL Rules** window, but by default it has the highest privileges.

4.4 ACL vulnerability

Now, in your browser, logout from DokuWiki and login as `setest`. If you go to the start page, `http://localhost/dokuwiki-2010-11-07/doku.php?id=start`, which is in the root namespace, you'll notice that you don't have edit privileges for that page – instead of the **Edit this page** button there is a **Show pagesource** button. This is because the `developer` group, to which `setest` was assigned, has only read privileges in the `root` namespace. You can click on the **Show pagesource** button and you will see a message explaining that to your current user, the content is read-only.



Trace: » pagey » pagex » start

This page is read only. You can view the source, but not change it. Ask your administrator if you think this is wrong.

```
Welcome to dokuwiki SEtest.  Take a look at these links:

- Link to [[devel:pageX|pageReadWrite]] in devel namespace

- Link to [[dev.l:pageY|pageReadOnly]] in dev.l namespace
```

Logged in as: setest (setest)

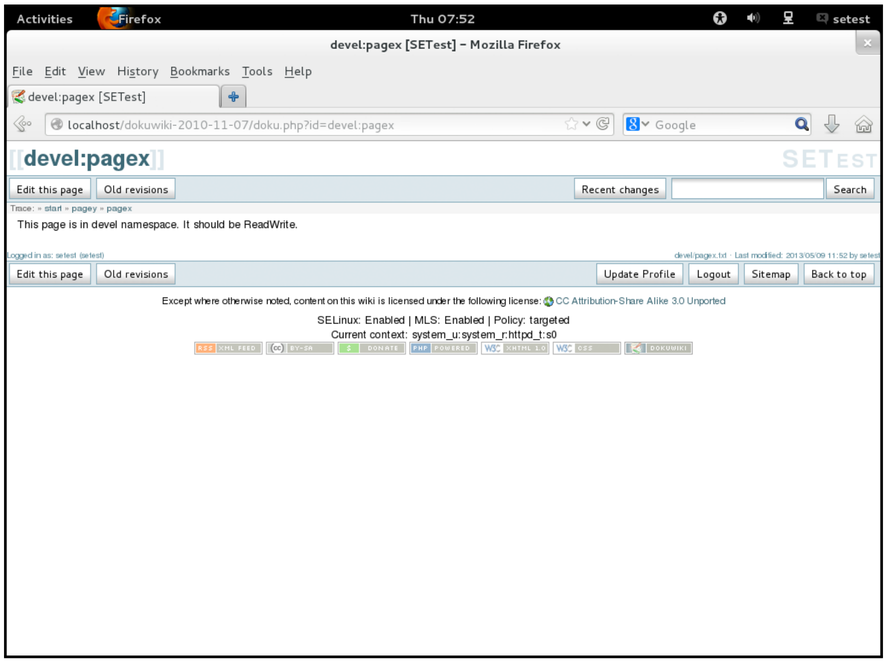
start.txt · Last modified: 2013/05/04 23:43 by admin

Except where otherwise noted, content on this wiki is licensed under the following license: CC Attribution-Share Alike 3.0 Unported

SELinux: Enabled | MLS: Enabled | Policy: targeted

Current context: system_u:system_r:httpd_t:s0

Now visit PageX, `http://localhost/dokuwiki-2010-11-07/doku.php?id=devel:pagex`. This page is in the `devel` namespace, to which `setest` has edit permissions. The **Edit this page** button is present, and thus the current user is free to edit the page content.



Trace: » start » pagey » pagex

This page is in devel namespace. It should be ReadWrite.

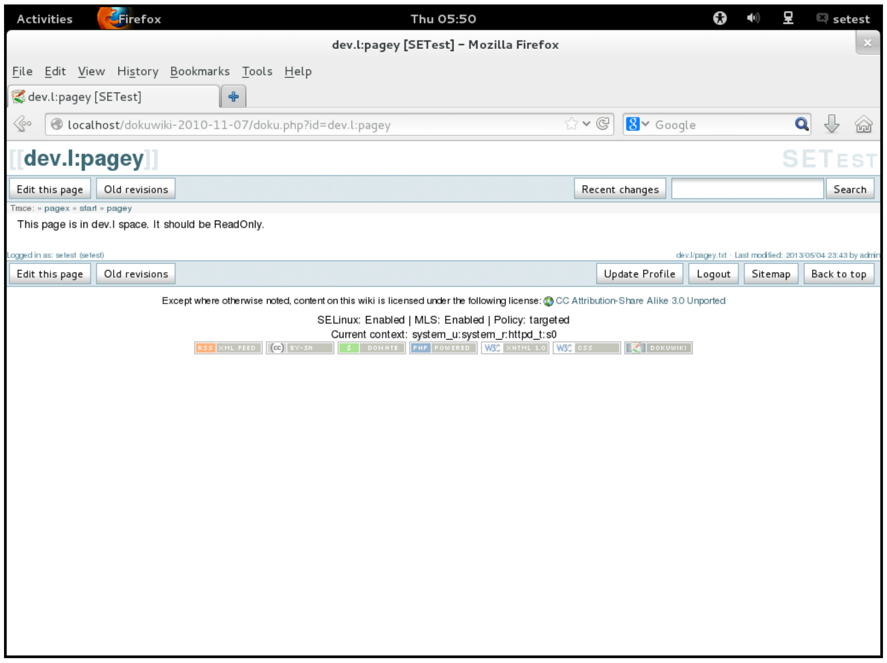
Logged in as: setest (setest)

devel/pagex.txt · Last modified: 2013/05/09 11:52 by setest

Except where otherwise noted, content on this wiki is licensed under the following license: CC Attribution-Share Alike 3.0 Unported

SELinux: Enabled | MLS: Enabled | Policy: targeted
Current context: system_u:system_r:httpd_t:s0

Now visit PageY, `http://localhost/dokuwiki-2010-11-07/doku.php?id=dev.l:pagey`. This page is in the `dev.l` namespace, to which the `setest` user is supposed to have read only permissions. Unexpectedly, however, the **Edit this page** button is still present. You can test it – if you click on it and edit the content, the page will actually get modified!



Trace: » pagex » start » pagey

This page is in dev.l space. It should be ReadOnly.

Logged in as: setest (setest)

dev.l/pagey.txt · Last modified: 2013/05/04 23:43 by admin

Except where otherwise noted, content on this wiki is licensed under the following license: CC Attribution-Share Alike 3.0 Unported

SELinux: Enabled | MLS: Enabled | Policy: targeted
Current context: system_u:system_r:httpd_t:s0

The ACL has failed – in fact, this is a known vulnerability for this version of DokuWiki (described here: `https://bugs.dokuwiki.org/index.php?do=details&task_id=2136`). It turns out that there is a bug in the ACL implementation that confuses namespaces that differ only by a letter replacement with the '.' character.

4.5 Double hull overview

At the moment, SELinux is not shielding us against this ACL vulnerability, because we haven not configured it yet to mirror DokuWiki's ACL permissions for different namespaces. There are a number of ways in which to configure SELinux to do this job. The approach in this tutorial is as follows: we will create a set of SELinux types, each labelling a directory on the server corresponding to a DokuWiki namespace (recall, that DokuWiki creates a separate directory for each namespace). We will also create a set of SELinux types corresponding to different DokuWiki groups. The *httpd* fork serving a client with credentials for a given group will transition into the domain of the corresponding SELinux source type. We will create `allow` rules in the policy such that for a given combination of source type and target type the actions for operations on "file" and "dir" objects will mirror the DokuWiki ACL permissions for the corresponding group and namespace. We can thus establish another independent ring of security around the DokuWiki system.

Table 4.2 shows the SELinux types and rules we intend to set up – note how the first three rows correspond to table 4.1 (SELinux does not allow periods in the label syntax, and so "dokuwiki_content_dev_l_t" corresponds to the `dev.l` namespace). The empty cells signify no allow rules. The last row of the table shows a domain created for media content, which DokuWiki stores in a separate directory – we need this type to provide double hull protection of the upload and delete permissions.

Table 4.2: SELinux types and allow rules that mirror DokuWiki's ACL

source `type` / target `type`	"dokuwiki_t"	"dokuwiki_developer_t"	"dokuwiki_admin_t"
"dokuwiki_content_t"		read	read,write,create,delete
"dokuwiki_content_devel_t"		read,write	read,write,create,delete
"dokuwiki_content_dev_l_t"		read	read,write,create,delete
"dokuwiki_content_media_t"		read	read,write,create,delete

4.6 Policy modules

In order to extend the policy to define new types and 'allow' rules we will write an SELinux *policy module*. A typical module contains rules pertaining to a specific service. For instance, the `apache` module specifies the rules regarding the domain that *httpd* is going to run in. It makes sense to put everything relating to our DokuWiki service into a custom module, and it is also a great way to get introduced to the art of SELinux policy writing.

The source code for a policy module is split into three files::

- `<module name>.te` – contains new policy definitions and permission rules
- `<module name>.if` – an interface file which contains macros – a typical macro consists of a set of rules with placeholders for parts of the security context passed in through tge arguments
- `<module name>.fc` – lists the configuration for SELinux labels on files

A module must have the `*.te` file, but the other two types of files are optional.

4.6.1 Build setup

In your home directory create a `modules` directory – that is where we will keep the source code for the new module. Within that directory, create the following `Makefile`.

File : Makefile

```
SELINUX_MAKE:= /usr/share/selinux/devel/Makefile

TE_MODULES:= $(wildcard *.te)
PP_MODULES:= $(subst , ,$(TE_MODULES:.te=.pp))

all:
	make -f $(SELINUX_MAKE)

install: $(PP_MODULES)
	@for module in $(PP_MODULES); do \
		echo "semodule -i $$module";\
		semodule -i $$module;\
	done

clean:
	make -f $(SELINUX_MAKE) clean
```

This `Makefile` contains instructions on compiling and installing a policy module. Note that the `make` and `clean` rules are just a redirect to `/usr/share/selinux/devel/Makefile`. This file got created when the `libselinux-devel` package was installed, and in addition to providing all the rules for policy compilation, it defines a number of useful macros (that speed up the process of policy writing).

4.7 The `dokuwiki` module

Our module will be called `dokuwiki`. At first it will define only one type, "dokukiwi_t", that is intended to tag the *httpd* fork that serves a client with lowest DokuWiki possible privileges (a non-logged in DokuWiki user that is allowed to access the login page only). We will add also a rule that allows a process running in the "httpd_t" domain to transition to the newly defined domain. In the `modules` directory create file `dokuwiki.te` that has the following contents:

File : dokuwiki.te

```
policy_module(dokuwiki, 1.0)

# Import existing types from the policy
require {
        type httpd_t;
        class process { dyntransition };
}

# Create new domain/type
type dokuwiki_t;

# Restrict dokuwiki_t permissions to be a
# subset of those of httpd_t
typebounds httpd_t dokuwiki_t;

# Allow transition from httpd_t to dokuwiki_t
allow httpd_t dokuwiki_t:process { dyntransition };
```

The first line specifies the name of the module and version – dokuwiki 1.0. Lines beginning with the `#` character are comments.

There `require` statement imports definitions from the existing policy – in this case a type "httpd_t" and object class "process" with an action "dyntransition".

Outside the `require` block, the `type` statements define new types. The module defines one new type – "dokuwiki_t". The `typebounds` statement creates a constraint, such that one domain cannot exceed permissions of the other. In our case, we limit the permissions of "dokuwiki_t" to never exceed those of the "httpd_t". This ensures that at runtime the permissions of "dokuwiki_t" are restricted to subset of permissions on "httpd_t". Type-bound restrictions are not checked at compile time. Hence, the policy will compile with allow rules on "dokuwiki_t" that exceed those of "httpd_t", except that at runtime, the extra permissions will be overruled by the `typebound` constraint. We can think of "dokuwiki_t" as being a subdomain of "httpd_t". This is a safety feature, which is enforced by Apache's `mod_selinux` module – the module refuses transition of an *httpd* fork to a type that is not type-bound to "httpd_t".

At runtime, whenever a change of type on a process is initiated, the kernel consults the LSM as to whether the "dyntransition" action is allowed on a "process" object to the new type. To allow this, at the end of `dokuwiki.te` there is an `allow` statement – it gives permission for the transition of a "process" object to the "dokuwikit_t" type while running in the "httpd_t" domain.

Compile this policy module—

```
$ make
make -f /usr/share/selinux/devel/Makefile
make[1]: Entering directory `/home/setest/modules'
Compiling targeted dokuwiki module
/usr/bin/checkmodule:  loading policy configuration from tmp/dokuwiki.tmp
/usr/bin/checkmodule:  policy configuration loaded
/usr/bin/checkmodule:  writing binary representation (version 15) to tmp/dokuwiki.mod
Creating targeted dokuwiki.pp policy package
rm tmp/dokuwiki.mod.fc tmp/dokuwiki.mod
make[1]: Leaving directory `/home/setest/modules'
```

—and install it:

```
$ sudo make install
semodule -i dokuwiki.pp
```

You can check if the module was loaded successfully along with the rest of the policy using the `semodule` command:

```
$ sudo semodule -l | grep dokuwiki
dokuwiki        1.0
```

4.8 Switching the domain

Having defined the policy, we can now consider when a forked *httpd* process, running on behalf of some client, will transition into the "dokuwiki_t" domain. Ideally this switch should happen at the time of the process forking. This type of transition can be provisioned by appropriate configuration of Apache's `mod_selinux` module. However, we will need to switch into a domain based on DokuWiki's ACL group, and that information is available some time later in the execution of the *httpd* fork – specifically, in the PHP code that verifies the credentials of the client.

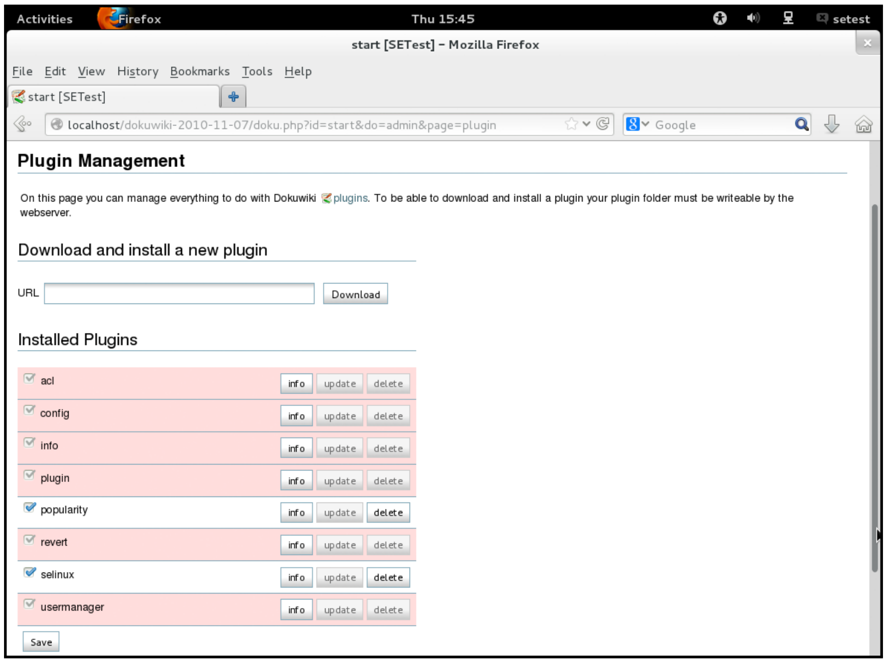
4.8.1 SELinux plugin

We thus need to trigger the domain transition in PHP, as soon as possible after the login credentials are available. By that time, the PHP script is being executed by an *httpd* fork, which was spawned specifically for a given client's HTTP request. The transition trigger can be provisioned through a DokuWiki plugin.

DokuWiki's source code defines a set of hooks, which will execute custom PHP code at various points of the execution of DokuWiki's scripts. One such point is right after the credentials of the connecting client have been checked. That is when we will trigger a switch on the SELinux context of the running process. Download the file `dokuwiki_plugin_selinux.tgz` and expand the `tar` file into DokuWiki's plugin directory

```
$ wget http://www.cs.otago.ac.nz/space/selinuxtutorial/archives/dokuwiki_plugin_selinux.tgz
$ tar -zxvf dokuwiki_plugin_selinux.tgz -C /var/www/html/dokuwiki-2010-11-07/lib/plugins/
```

The new plugin should be picked up automatically by DokuWiki. Just to be sure, in your browser, login to DokuWiki as superuser and go to the **Manage Plugins** page from the **Admin** section of the website, `http://localhost/dokuwiki-2010-11-07/doku.php?id=start&do=admin&page=plugin`, and confirm that `selinux` plugin is selected.


Activities
Firefox
Thu 15:45
setest
start [SETest] – Mozilla Firefox
File Edit View History Bookmarks Tools Help
start [SETest]
localhost/dokuwiki-2010-11-07/doku.php?id=start&do=admin&page=plugin
Google

Plugin Management

On this page you can manage everything to do with Dokuwiki plugins. To be able to download and install a plugin your plugin folder must be writeable by the webserver.

Download and install a new plugin

URL [ ] Download

Installed Plugins

✓ acl	info	update	delete
✓ config	info	update	delete
✓ info	info	update	delete
✓ plugin	info	update	delete
✓ popularity	info	update	delete
✓ revert	info	update	delete
✓ selinux	info	update	delete
✓ usermanager	info	update	delete

Save

The PHP code that triggers the transition can be found in the `/var/www/html/dokuwiki-2010-11-07/action.php` file; the contents of the file are listed below:

File : action.php

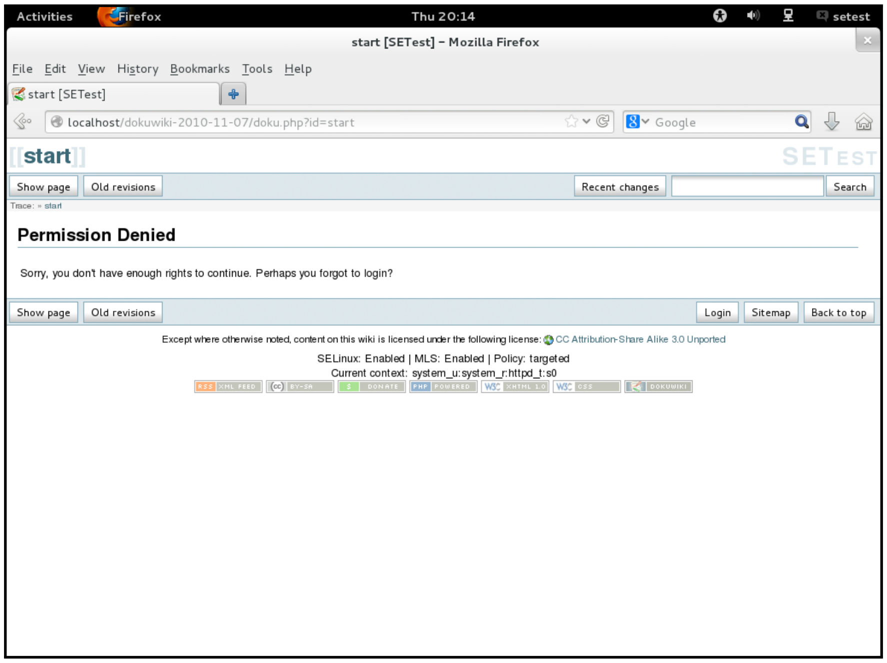
```
<?php
/**
 * DokuWiki Plugin selinux (Action Component)
 *
 * @license GPL 2 http://www.gnu.org/licenses/gpl-2.0.html
 * @author  Lech Szymanski <lechszym@cs.otago.ac.nz>
 */

// must be run within Dokuwiki
if (!defined('DOKU_INC')) die();

if (!defined('DOKU_LF')) define('DOKU_LF', "\n");
if (!defined('DOKU_TAB')) define('DOKU_TAB', "\t");
if (!defined('DOKU_PLUGIN')) define('DOKU_PLUGIN',DOKU_INC.'lib/plugins/');

require_once DOKU_PLUGIN.'action.php';

class action_plugin_selinux extends DokuWiki_Action_Plugin {
    public function register(Doku_Event_Handler &$controller) {

       $controller->register_hook('AUTH_LOGIN_CHECK', 'AFTER', $this, 'handle_domain_switch'
           );
    }

    public function handle_domain_switch(Doku_Event &$event, $param) {
       global $conf;
       global $USERINFO;

       if(!$conf['useacl']) {
              return;
       }

       $grp = $USERINFO['grps'][0];

       $domain = 'dokuwiki_t';
       if(!empty($grp)) {
          $domain = 'dokuwiki_' . $grp . '_t';
       }

       $currentCon = selinux_getcon();

       /* First token is for user */
       $tok = strtok($currentCon,':');
       $newCon = $tok;
       $count = 2;
       $tok = strtok(':');

       while(false !== $tok && $count<7) {
           /* $count == 2  is for role
            *        == 3  is for type
            *        == 4  is for sensitivity level
            */

           if($count == 3) {
               $tok = $domain;
           }
           $newCon .= ':' . $tok;
           $tok = strtok(':');
           $count++;
       }
       selinux_setcon($newCon);
    }
}
```

The script registers a hook for just after the login credentials check, which DokuWiki performs for every client request to verify the user's permissions, as picked up from the appropriate session cookie. The `selinux_*` functions are part of the SELinux API for PHP. The plugin code retrieves the ACL group of the current user, and then retrieves the SELinux label of the process executing the script. The label is parsed and the third token, corresponding to `type`, is replaced with a type derived from the ACL group. If the user is not logged in, the `$grp` variable will be empty and the new domain type will be "dokuwiki_t", otherwise the new label is created with the following format "dokuwiki_<$grp>_t". The last function call initiates the switch of the current process to the new context.

Since we have only defined the "dokuwiki_t" type, the types derived from ACL group name in the above plugin will not be valid and SELinux transition of the context will not work.

The plugin is running already, and it should be trying to make a transition to the "doku_-admin_t" domain (assuming you're still logged-in to DokuWiki a a superuser). Take a look at the footer of the current DokuWiki page – the type in the listed SELinux context should be "httpd_t". That means that the transition didn't work, and the reason is that the type "doku_admin_t" has not been define yet.

Logout from DokuWiki. Now, when accessing DokuWiki's login page, the plugin should attempting a transition from "httpd_t" to "dokuwiki_t" on the running process. The target type has been defined and the transition was allowed in our module. However, you'll notice that the context at the bottom of the DokuWiki login page still has `type`="httpd_t" – the process did not switch domains.



Permission Denied

Sorry, you don't have enough rights to continue. Perhaps you forgot to login?

Except where otherwise noted, content on this wiki is licensed under the following license: CC Attribution-Share Alike 3.0 Unported

SELinux: Enabled | MLS: Enabled | Policy: targeted
Current context: system_u:system_r:httpd_t:s0

If you check the log file `/var/log/audit/audit.log` there should be a message near the end similar to this one:

```
type=AVC msg=audit(1368145147.499:1103): avc:  denied  { setcurrent } for  pid=15438 comm="
    httpd" scontext=system_u:system_r:httpd_t:s0 tcontext=system_u:system_r:httpd_t:s0
    tclass=process
type=SYSCALL msg=audit(1368145147.499:1103): arch=40000003 syscall=4 success=no exit=-13 a0=
    c a1=b4f21448 a2=20 a3=b4f21448 items=0 ppid=14057 pid=15438 auid=4294967295 uid=48 gid
    =48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 ses=4294967295 tty=(none) comm="
    httpd" exe="/usr/sbin/httpd" subj=system_u:system_r:httpd_t:s0 key=(null)
```

It seems that SELinux is denying the transition from "httpd_t". The details state what the problem is – a process of the "httpd_t" type is not allowed to perform the "setcurent" action on a "process" of "httpd_t" type. Recall from the previous chapter, that in order to change a label on a directory two sets of permissions are needed: "labelto" (the new type) and "labelfrom" (the old type). Domain transitions for processes operate in a similar way. In our module, we have created a rule that allows transition of a process to "domain_t" while running in the "httpd_t" domain. That rule does not actually allow "httpd_t" to transition from its type to another. We need to add a separate rule for that. Edit the `dokuwikit.te` file to match the contents listed below:

File : dokuwiki.te

```
policy_module(dokuwiki , 1.1)

# Import existing types from the policy
require {
        type httpd_t;
        class process { dyntransition setcurrent };
}

# Create new domain/type
type dokuwiki_t;

# Restrict dokuwiki_t privileges to be a
# subset of those of httpd_t
typebounds httpd_t dokuwiki_t;

# Allow transition from httpd_t to dokuwiki_t
allow httpd_t dokuwiki_t:process { dyntransition };
allow httpd_t self:process { setcurrent };
```

Note that the "setcurrent" action has been added to the require block for the "process" class. The rule to allow the transition from "httpd_t" while running in the "httpd_t" domain is added at the bottom – it uses the keyword `self`, which derives the target type from the source type of the rule. Hence, the last statement is analogous to the statement `allow httpd_t httpd_t:process {setcurrent}`. Issue `make` and `sudo make install`. The module with the new rules will simply replace the previous one. Refresh the webpage.

If you refresh your DokuWiki login page you'll notice that the transition to "dokuwiki_t" still is not happening. Searching through the audit log again will not reveal a clear AVC message: this time there is no information from SELinux regarding what needs to be done next. A reasonable step to take at this point would be to set SELinux to permissive mode and test if the context switch occurs – usually this would verify that it's SELinux permissions on the transition that are the problem, and not something else. However, in this case, if you do switch to permissive mode, the context switch still will not occur. This is unfortunate, because despite what this might suggest, it *is* actually SELinux that is preventing the switch. This is probably the greatest source of frustration for developers wrestling with SELinux – sometimes, SELinux will not tell you what is wrong in its usual diagnostic logging. Sometimes these 'silent' errors occur regardless whether SELinux is in permissive or enforcing mode. Remember, in permissive mode the kernel still consults the LSM, and when the LSM API generates an error (rather than a denial), the kernel might abort the action.

In our case, the problem stems from the fact that the newly created type actually needs a set of attributes in order for the LSM to allow the type to tag a process. However, even that will not be enough for "dokuwiki_t" to operate properly. Another problem is that "dokuwi_t" is meant to tag the *httpd* process, and that process needs a large selection of permissions in order to serve a web page to a client. Take a look at the set of permissions that the targeted policy lists for the "httpd_t" type:

```
$ sesearch --allow -s httpd_t
Found 1175 semantic av rules:
   allow httpd_t pcscd_var_run_t : file { ioctl read getattr lock open } ;
   allow httpd_t pcscd_var_run_t : dir { getattr search open } ;
   allow httpd_t privfd : fd use ;
   .
   .
   .
```

There's a lot of them, and (as you would expect) they are not there without a reason. In order to allow the *httpd* process to function properly, all these permissions are necessary (remember, if there is no 'allow' rule for some action, it means that it will be denied by the LSM). We need to make sure that the "dokuwiki_t" type is set up with all of these permissions.

4.8.2 Cloning a domain

Unfortunately, there is no easy way to copy all of the permissions of one domain to another in SELinux. The `typebounds` statement restricts a domain to be a subset of another other, but it does not actually transfer the 'allow' rules from one domain to another. There is no statement in the policy that would allow a domain to take on all the properties of the other domain. One possible workaround is to create an attribute, to which we can assign all of the required permissions, and then assign the attribute to two (or more) types. However, the targeted policy does not create a single attribute that contains all of the permissions required for the *httpd* process.

We will use a particular workaround. We will use the `sesearch` command to list all the permissions for the "httpd_t" domain and use a script to create a policy template that assigns all those permissions to another domain. In the terminal, while in the `modules` directory, download and run the following script:

```
$ wget http://www.cs.otago.ac.nz/space/selinuxtutorial/scripts/clone_selinux_domain.sh
$ bash clone_selinux_domain.sh httpd_t > dokuwiki.if
```

The script takes a while to run – this is because there are a lot of rules to process, and Bash string manipulation (at least the way we wrote it in this script) is quite slow. The script accepts an argument for the domain type that you want to clone – in this case "httpd_t". The output of the macro is redirected to `dokuwiki.if` – the interface file for the `dokuwiki` module. Once the script is finished, the macro should look like this:

```
$ cat dokuwiki.if
interface(`clone_httpd_t',`
        gen_require(`
          type httpd_t;
          type lib_t;
          .
          .
          .
       ')

   typeattribute $1 kernel_system_state_reader;
   typeattribute $1 syslog_client_type;
   .
   .
   .

   allow $1 httpd_t : process { fork transition sigchld sigkill sigstop signull signal
       getsched setsched getsession getpgid setpgid getcap setcap shar\
e getattr setfscreate noatsecure siginh rlimitinh dyntransition setkeycreate setsockcreate
    ptrace_child } ;
   allow $1 httpd_t : capability { chown dac_override kill setgid setuid net_bind_service
       sys_nice sys_tty_config } ;
   .
   .
   .
')
```

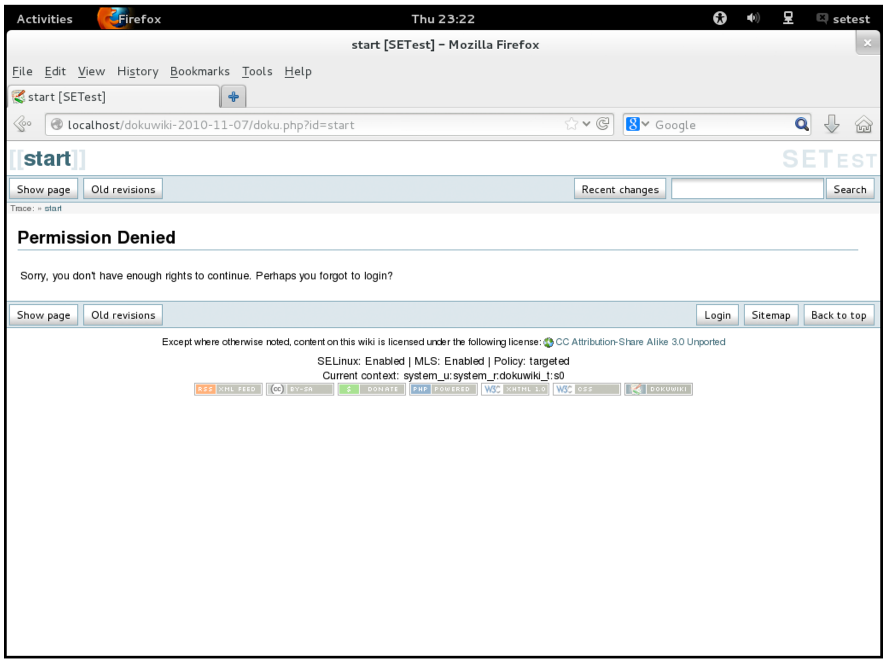
The name of the macro is `clone_httpd_t` and it takes one argument. It's somewhat similar to a bash script, in that it takes a list of arguments, which are referred to with the argument number preceded by the $ sign. Since this macro resides in the interface file for our module, `dokuwiki.if`, the `Makefile` will pick it up. All we need to do is to change `dokuwikit.te` to call the macro after creating the new domain, as shown below:

File : dokuwiki.te

```
policy_module(dokuwiki , 1.2)

# Import existing types from the policy
require {
        type httpd_t;
        class process { dyntransition setcurrent };
}

# Create new domain/type
type dokuwiki_t;
clone_httpd_t(dokuwiki_t);

# Restrict dokuwiki_t privileges to be a
# subset of those of httpd_t
typebounds httpd_t dokuwiki_t;

# Allow transition from httpd_t to dokuwiki_t
allow httpd_t dokuwiki_t:process { dyntransition };
allow httpd_t self:process { setcurrent };
```

Run `make` and `sudo make install`. In the browser, refresh the DokuWiki login page, and this time the transition to "dokuwikit_t" should be successful (as confirmed by the context in the footer of the page).

The `sesearch` command used in the clone script picks up conditional `allow` statements, regardless whether they are enabled or not. These rules go into the clone domain macro as unconditional rules. However, in our case, the clone is `typebound` by the original domain, which will prevent the new domain from doing something at runtime that the original one is not allowed to do.



Permission Denied

Sorry, you don't have enough rights to continue. Perhaps you forgot to login?

Except where otherwise noted, content on this wiki is licensed under the following license: CC Attribution-Share Alike 3.0 Unported

SELinux: Enabled | MLS: Enabled | Policy: targeted
Current context: system_u:system_r:dokuwiki_t:s0

The new domain is working, and the transition works too – now it's time to create SELinux domains for the respective ACL groups.

4.9 DokuWiki ACL group domains

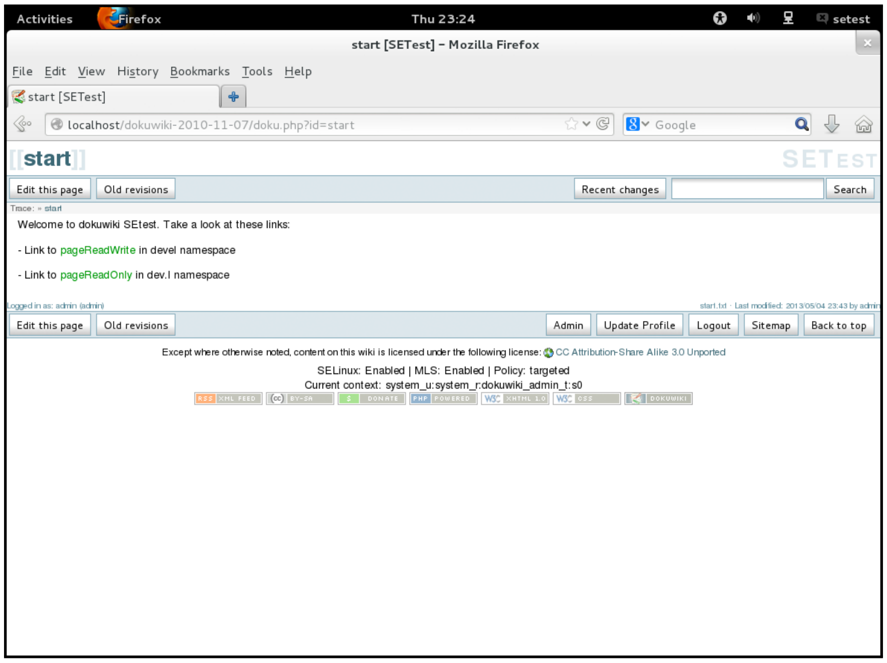
Recall that when a user logs into DokuWiki, the plugin we have installed attempts to transition the *httpd* process to domain "dokuwiki_<$grp>_t", where <$grp> is the ACL group that the DokuWiki user belongs to. The transition occurs once for every HTTP request. We need to create the following SELinux domains: "dokuwiki_admin_t" and "dokuwiki_developer_t" corresponding to DokuWiki's `admin` and `developer` groups respectively. Modify the `dokuwiki.te` file to match the following:

File : dokuwiki.te

```
policy_module(dokuwiki, 1.3)

# Import existing types from the policy
require {
        type httpd_t;
        class process { dyntransition setcurrent };
}

# Create new domain/type
type dokuwiki_t;
clone_httpd_t(dokuwiki_t);

# Restrict dokuwiki_t privileges to be a
# subset of those of httpd_t
typebounds httpd_t dokuwiki_t;

# Allow transition from httpd_t to dokuwiki_t
allow httpd_t dokuwiki_t:process { dyntransition };
allow httpd_t self:process { setcurrent };

# Create domain for admin group
type dokuwiki_admin_t;
clone_httpd_t(dokuwiki_admin_t);
typebounds httpd_t dokuwiki_admin_t;
allow httpd_t dokuwiki_admin_t:process { dyntransition };

# Create domain for developer group
type dokuwiki_developer_t;
clone_httpd_t(dokuwiki_developer_t);
typebounds httpd_t dokuwiki_developer_t;
allow httpd_t dokuwiki_developer_t:process { dyntransition };
```

Both new domains need to clone "httpd_t" and are type-bound by it. Also, transition to each of those domains from `httpd_t` is allowed.

Run `make` and `sudo make install`. Now, when you test DokuWiki, the plugin should switch to "dokuwiki_admin_t" when logged in as `admin`—



Welcome to dokuwiki SEtest. Take a look at these links:

- Link to pageReadWrite in devel namespace

- Link to pageReadOnly in dev.l namespace

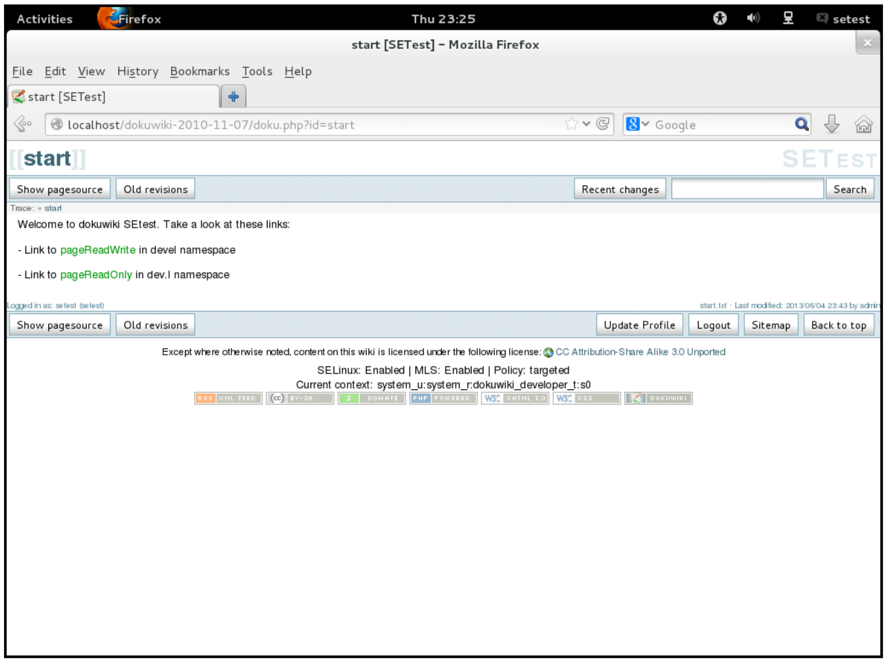
Logged in as: admin (admin)

start.txt · Last modified: 2013/05/04 23:43 by admin

Except where otherwise noted, content on this wiki is licensed under the following license: CC Attribution-Share Alike 3.0 Unported

SELinux: Enabled | MLS: Enabled | Policy: targeted
Current context: system_u:system_r:dokuwiki_admin_t:s0

—and to "dokuwiki_developer_t" when logged in to DokuWiki as `setest`:



Welcome to dokuwiki SEtest. Take a look at these links:

- Link to pageReadWrite in devel namespace

- Link to pageReadOnly in dev.l namespace

Logged in as: setest (setest)

start.txt · Last modified: 2013/05/04 23:43 by admin

Except where otherwise noted, content on this wiki is licensed under the following license: CC Attribution-Share Alike 3.0 Unported

SELinux: Enabled | MLS: Enabled | Policy: targeted
Current context: system_u:system_r:dokuwiki_developer_t:s0

Recall that it is the SELinux plugin that, using ACL credentials, figures out what domain to transition to.

Note that, at this point, the three new domains have the same permissions. Hence, we can simplify the code by creating a macro. Add the following code at the top of `dokuwiki.if` – but DO NOT overwrite the `clone_httpd_t` macro:

File : dokuwiki.if

```
interface(`create_dokuwiki_domain',`
        gen_require(`
          type httpd_t;
          class process { dyntransition };
        ')

        # Create new domain/type
        type $1;
        clone_httpd_t($1);

        # Restrict new domain privileges to be a
        # subset of those of httpd_t
        typebounds httpd_t $1;

        # Allow transition from httpd_t to the new domain
        allow httpd_t $1:process { dyntransition };
')
```

The `create_dokuwiki_domain` macro takes one argument, symbolised by `$1`. It generates a `require` statement that will import existing types, classes and actions, used by the macro. Then, the macro lists all the rules that will get assigned to the domain passed in the argument (including the call to the `clone_httpd_t` macro). Now, using the new macro, we can simplify the code in `dokuwiki.te` as follows:

File : dokuwiki.te

```
policy_module(dokuwiki, 1.4)

# Import existing types from the policy
require {
        type httpd_t;
        class process { setcurrent };
}

allow httpd_t self:process { setcurrent };

# Create domain for no permissions
create_dokuwiki_domain(dokuwiki_t);

# Create domain for admin group
create_dokuwiki_domain(dokuwiki_admin_t);

# Create domain for developer group
create_dokuwiki_domain(dokuwiki_developer_t);
```

Compile and install the policy (`make` and `sudo make install`). The SELinux plugin should function just like before, switching to the appropriate group domain for different users.

4.10 ACL namespace types

Although we have successfully caused the *httpd* process to transition to an SELinux domain corresponding to the DokuWiki user's group, at this point there are no rules that differentiate these domains. Since DokuWiki group permissions can vary for different namespaces, we need a different domain to label each namespace directory. Modify `dokuwiki.te` to the following:

File : dokuwiki.te

```
policy_module(dokuwiki, 1.5)

# Import existing types from the policy
require {
        type httpd_t;
        class process { setcurrent };
}

allow httpd_t self:process { setcurrent };

# Create domain for no permissions
create_dokuwiki_domain(dokuwiki_t);

# Create domain for admin group
create_dokuwiki_domain(dokuwiki_admin_t);

# Create domain for developer group
create_dokuwiki_domain(dokuwiki_developer_t);

# Create type for root namespace
type dokuwiki_content_t;
files_type(dokuwiki_content_t);

# Create type for devel namespace
type dokuwiki_content_devel_t;
files_type(dokuwiki_content_devel_t);

# Create type for dev.1 namespace
type dokuwiki_content_dev_1_t;
files_type(dokuwiki_content_dev_1_t);

# Create type for media files
type dokuwiki_content_media_t;
files_type(dokuwiki_content_media_t);
```

We have created four new types: "dokuwiki_content_t", "dokuwiki_content_devel_t", "dokuwiki_content_dev_1_t", and"dokuwki_content_media_t". The first three types correspond to DokuWiki's `root`, `devel` and `dev.1` namespaces respectively (policy doesn't like the period character in the type name). The last type is for upload and delete permissions on media files. Whereas the types created in the previous sections were meant to label a process, these new types are supposed to label files and directories. Once again, a long list of rules is needed in order for a type to function as a label of a file or a directory, but to help with this rule assignment the policy provides a `file_type` attribute that collects together all of the required permissions. The easiest way to assign the attribute to a new type is to use the `files_type` macro that is provided by the policy. This macro is defined in an interface file `/usr/share/selinux/devel/include/kernel/files.if`, which is picked up by our `Makefile`. If you wish to examine the macro, you will notice that it assigns a number of attributes to the type that is passed in as an argument.

Next, we need to label DokuWiki's directories with the appropriate namespace type. In the previous section we used the `chcon` command to relabel files and directories. This time we will take advantage of the `*.fc` file that gets compiled with our policy module. In the same directory where your `dokuwiki.te` and `dokuwiki.if` reside, create `dokuwiki.fc` with the following contents:

File : dokuwiki.fc

```
/var/www/html/dokuwiki-2010-11-07(/.*)?                    gen_context(unconfined_u:object_r
    :httpd_sys_content_t,s0)
/var/www/html/dokuwiki-2010-11-07/conf(/.*)?               gen_context(unconfined_u:object_r
    :httpd_sys_rw_content_t,s0)
/var/www/html/dokuwiki-2010-11-07/data(/.*)?               gen_context(unconfined_u:object_r
    :httpd_sys_rw_content_t,s0)
/var/www/html/dokuwiki-2010-11-07/data/media(/.*)?         gen_context(unconfined_u:object_r
    :dokuwiki_content_media_t,s0)
/var/www/html/dokuwiki-2010-11-07/data/pages/(.txt)?       gen_context(unconfined_u:object_r
    :dokuwiki_content_t,s0)
/var/www/html/dokuwiki-2010-11-07/data/pages/devel(/.*)?   gen_context(unconfined_u:object_r
    :dokuwiki_content_devel_t,s0)
/var/www/html/dokuwiki-2010-11-07/data/pages/dev.l(/.*)?   gen_context(unconfined_u:object_r
    :dokuwiki_content_dev_l_t,s0)
```

The `*.fc` file lists directories and their corresponding SELinux contexts. The term `(/.*)?` is a regular expression signifying inclusion of all subdirectories and their contents. The labelling rules we have are specified for the entire DokuWiki directory – the rules that come last overrule the previous ones. So, although the entire DokuWiki directory is labelled with "httpd_sys_content_t" (a readonly type for a process confined to the "httpd_t" domain), the rules that follow label the subdirectories differently.

Compile the policy module and install it. Notice that the contexts of the namespace directories have not been changed:

```
$ ls -Z /var/www/html/dokuwiki-2010-11-07/data/pages/
drwxr-xr-x. apache apache system_u:object_r:httpd_sys_rw_content_t:s0 devel
drwxr-xr-x. apache apache system_u:object_r:httpd_sys_rw_content_t:s0 dev.l
drwxrwxr-x. setest apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 playground
-rw-r--r--. apache apache system_u:object_r:httpd_sys_rw_content_t:s0 start.txt
drwxrwxr-x. setest apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 wiki
```

This is because the labels from the `*.fc` are not applied to the system when new module is loaded. Instead, these labelling instructions get added to the default labelling configuration. To verify that the default configuration settings have changed, take a look at the `/etc/selinux/targeted/contexts/file_context` file. It should contain the following entries:

```
$ cat /etc/selinux/targeted/contexts/file_contexts | grep dokuwiki
/var/lib/dokuwiki(/.*)? system_u:object_r:httpd_sys_rw_content_t:s0
/var/www/html/dokuwiki-2010-11-07(/.*)? unconfined_u:object_r:httpd_sys_content_t:s0
/var/www/html/dokuwiki-2010-11-07/conf(/.*)?    unconfined_u:object_r:httpd_sys_rw_content_t
    :s0
/var/www/html/dokuwiki-2010-11-07/data(/.*)?    unconfined_u:object_r:httpd_sys_rw_content_t
    :s0
/var/www/html/dokuwiki-2010-11-07/data/media(/.*)?      unconfined_u:object_r:
    dokuwiki_content_media_t:s0
/var/www/html/dokuwiki-2010-11-07/data/pages/(.txt)?    unconfined_u:object_r:
    dokuwiki_content_t:s0
/var/www/html/dokuwiki-2010-11-07/data/pages/dev.l(/.*)?        unconfined_u:object_r:
    dokuwiki_content_dev_l_t:s0
/var/www/html/dokuwiki-2010-11-07/data/pages/devel(/.*)?        unconfined_u:object_r:
    dokuwiki_content_devel_t:s0
```

To apply the new labelling, we can use the "restore default context" facility on DokuWiki's root directory:

```
$ sudo restorecon -R /var/www/html/dokuwiki-2010-11-07
```

The -R option makes the call recursive, so default contexts are also restored on all subdirectories. The new labels should look like this:

```
$ ls -Z /var/www/html/dokuwiki-2010-11-07/data
drwxrwxr-x. setest apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 attic
drwxrwxr-x. setest apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 cache
-rw-rw-r--. setest apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 _dummy
drwxrwxr-x. setest apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 index
drwxrwxr-x. setest apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 locks
drwxrwxr-x. setest apache unconfined_u:object_r:dokuwiki_content_media_type:s0 media
drwxrwxr-x. setest apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 meta
drwxrwxr-x. setest apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 pages
-rw-rw-r--. setest apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 security.png
-rw-rw-r--. setest apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 security.xcf
drwxrwxr-x. setest apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 tmp
```

If the labels are still not as shown above, it might be because of the changes we have made previously to the default context with the `semanage` command. Take a look at the contents of the `cat /etc/selinux/targeted/contexts/files/file_contexts.local` file:

```
$ cat /etc/selinux/targeted/contexts/files/file_contexts.local
# This file is auto-generated by libsemanage
# Do not edit directly.

/var/www/html/dokuwiki-2010-11-07/conf     system_u:object_r:httpd_sys_rw_content_t:s0
/var/www/html/dokuwiki-2010-11-07/data(/.*)?    system_u:object_r:httpd_sys_rw_content_t:s0
```

The `restorecon` command restores context from `file_contexts.local` after the contexts from `file_contexts`, and so if your `file_contexts.local` contains the two entries as shown above, they overwrite everything. Use the `semanage` command again to delete the local default context settings like so:

```
$ sudo semanage fcontext -d /var/www/html/dokuwiki-2010-11-07/conf
$ sudo semanage fcontext -d "/var/www/html/dokuwiki-2010-11-07/data(/.*)?"
```

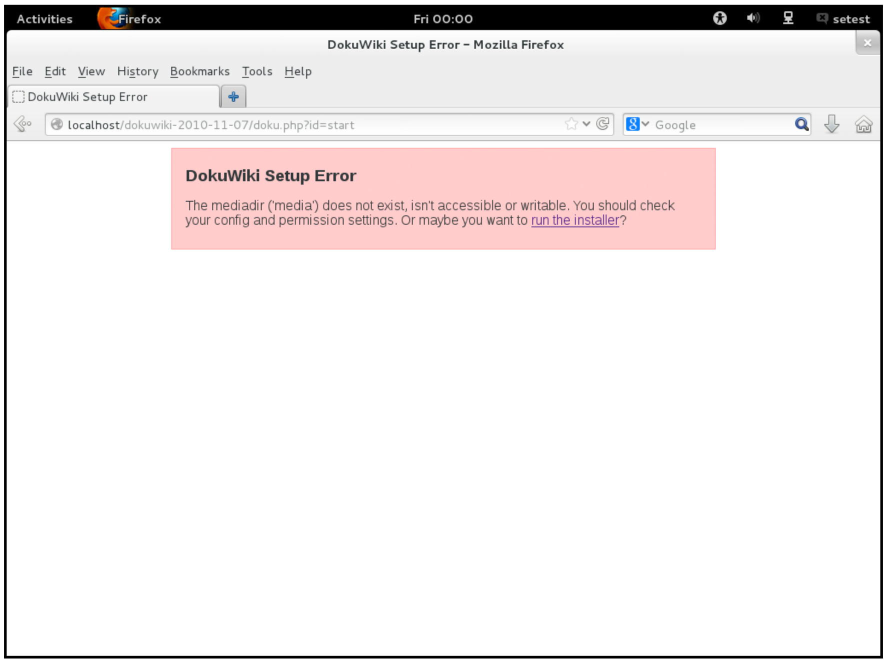
In the second command the quotes are necessary, because of the regular expression symbols at the end. Check that `file_contexts.local` does not list those default contexts anymore;

```
$ cat /etc/selinux/targeted/contexts/files/file_contexts.local
# This file is auto-generated by libsemanage
# Do not edit directly.
```

Try the `restorecon` command again and now DokuWiki directories should be labelled correctly. Note that the `media` directory is labelled with the "dokuwiki_content_media_t" type – this is a special type used for ACL upload and delete permissions.

```
$ ls -Z /var/www/html/dokuwiki-2010-11-07/data/pages
drwxr-xr-x. apache apache system_u:object_r:dokuwiki_content_devel_type:s0 devel
drwxr-xr-x. apache apache system_u:object_r:dokuwiki_content_dev_l_type:s0 dev.l
drwxrwxr-x. setest apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 playground
-rw-r--r--. apache apache system_u:object_r:httpd_sys_rw_content_t:s0 start.txt
drwxrwxr-x. setest apache unconfined_u:object_r:httpd_sys_rw_content_t:s0 wiki
```

Reload the DokuWiki website in your browser. You should get a message like this:


Activities
Firefox
Fri 00:00
setest
DokuWiki Setup Error – Mozilla Firefox
File Edit View History Bookmarks Tools Help
DokuWiki Setup Error
localhost/dokuwiki-2010-11-07/doku.php?id=start
Google

DokuWiki Setup Error

The mediadir ('media') does not exist, isn't accessible or writable. You should check your config and permission settings. Or maybe you want to run the installer?

This error occurs because we have relabelled various DokuWiki directories, but no permissions have been given to any of the process labelling types – or even the "httpd_t" domain – to perform any actions on the newly defined content types. We need to specify those permissions in our policy module, and we will do it to match the setup from Table 4.2.

4.11 DokuWiki double hull security setup

Since DokuWiki's ACL permissions are hierarchical, we will create a set of macros, each of which at its end will call the previous one, and thus inherit its permissions. Add the following macros at the top of your `dokuwiki.if`.

File : dokuwiki.if

```
interface(`dokuwiki_acl_read',`
        getattr_files_pattern($1,$2,$2);
        read_files_pattern($1,$2,$2);
        read_files_pattern($1,dokuwiki_content_media_t,dokuwiki_content_media_t);
')

interface(`dokuwiki_acl_edit',`
        dokuwiki_acl_read($1,$2);
        manage_files_pattern($1,$2,$2);
')

interface(`dokuwiki_acl_create',`
        dokuwiki_acl_edit($1,$2);
        create_files_pattern($1,$2,$2);
')

interface(`dokuwiki_acl_upload',`
        gen_require(`
                type dokuwiki_content_media_t;
        ')

        dokuwiki_acl_create($1,$2);
        manage_files_pattern($1,dokuwiki_content_media_t,dokuwiki_content_media_t);
')

interface(`dokuwiki_acl_delete',`
        dokuwiki_acl_upload($1,$2);
        delete_files_pattern($1,dokuwiki_content_media_t,dokuwiki_content_media_t);
')

interface(`dokuwiki_acl_admin',`
        dokuwiki_acl_delete($1,$2);
')
```

These six macros correspond to read, edit, create, upload, delete and admin privileges. Each macro takes two arguments: `$1` is the SELinux type corresponding to a DokuWiki group, `$2` is the SELinux type corresponding to a DokuWiki namespace. The macros treat `$1` and `$2` as source and target types respectively. Aside from calling the previously defined macro, each macro also relies on some pre-defined macros that are found in `/usr/share/selinux/devel/include/support/file_patterns.spt`. These macros are a convenient way to assign appropriate 'allow' rules to the source domain for read and write permissions to files and directories of a target type:

- `getattr_files_pattern` allows a process of type `$1` to fetch attributes of directories of type `$2` and files of type `$3`
- `read_files_pattern` allows a process of type `$1` to open and read directories of type

$2 and files of type $3

- manage_files_pattern allows a process of type $1 to open, read and write to directories of type $2 and files of type $3
- create_files_pattern allows a process of type $1 to create files inside directories of type $2
- create_files_pattern allows a process of type $1 to delete directories of type $2 and files of type $3

Note that the dokuwiki_acl_upload and dokuwiki_acl_delete permissions add read-/write and delete permissions for files in the "dokuwiki_content_media_type" domain. The dokuwiki_acl_admin macro gives the same permissions as the dokuwiki_acl_delete macro, hence in our setup SELinux double-hull security only encompasses the access to the DokuWiki content – permissions for administration of the website are handled solely by the ACL.

Below are the contents of the new version of dokuwiki.te, which, using the newly defined macros, adds the permissions for group domains on namespace types according to Table 4.2.

File : dokuwiki.te

```
policy_module(dokuwiki, 1.6)

# Import existing types from the policy
require {
        type httpd_t;
        class process { setcurrent };
}

allow httpd_t self:process { setcurrent };

# Create domain for no permissions
create_dokuwiki_domain(dokuwiki_t);

# Create domain for admin group
create_dokuwiki_domain(dokuwiki_admin_t);

# Create domain for developer group
create_dokuwiki_domain(dokuwiki_developer_t);

# Create type for root namespace
type dokuwiki_content_t;
files_type(dokuwiki_content_t);

# Create type for devel namespace
type dokuwiki_content_devel_t;
files_type(dokuwiki_content_devel_t);

# Create type for dev.1 namespace
type dokuwiki_content_dev_1_t;
files_type(dokuwiki_content_dev_1_t);

# Create type for media files
type dokuwiki_content_media_t;
files_type(dokuwiki_content_media_t);

# Admin permissions for httpd_t on all namespaces
dokuwiki_acl_admin(httpd_t,dokuwiki_content_t);
dokuwiki_acl_admin(httpd_t,dokuwiki_content_devel_t);
dokuwiki_acl_admin(httpd_t,dokuwiki_content_dev_1_t);

# Admin permissions for dokuwiki_admin_t on all namespaces
dokuwiki_acl_admin(dokuwiki_admin_t,dokuwiki_content_t);
dokuwiki_acl_admin(dokuwiki_admin_t,dokuwiki_content_devel_t);
dokuwiki_acl_admin(dokuwiki_admin_t,dokuwiki_content_dev_1_t);

# Permissions for dokuwiki_developer_t on all namespaces
dokuwiki_acl_read(dokuwiki_developer_t,dokuwiki_content_t);
dokuwiki_acl_edit(dokuwiki_developer_t,dokuwiki_content_devel_t);
dokuwiki_acl_read(dokuwiki_developer_t,dokuwiki_content_dev_1_t);
```

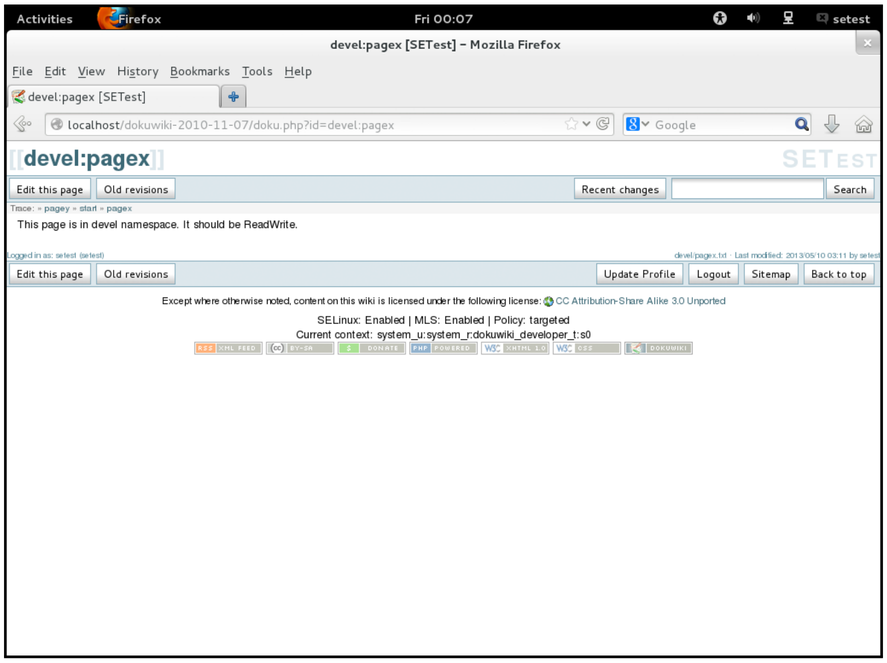
Note that the "httpd_t" domain requires all privileges on all namespaces, because all the dokuwiki domains are type-bound by it. The domain corresponding to the admin group, "dokuwiki_admin_t" is assigned admin privileges on all namespaces. The domain corresponding to the developer group, "dokuwiki_developer_t", gets read-only privileges on the "dokuwiki_content_t" and "dokuwiki_content_dev_l" types, and edit privileges on the "dokuwiki_devel" type. The "dokuwiki_t" type is not assigned any privileges because that's the type corresponding to a client that has not logged in (it still can read "httpd_sys_content_t" and write to "httpd_sys_rw_content_t – it gets those permissions by inheriting the permissions from the `clone_httpd_t` macro".

Compile and install the `dokuwiki` policy module.

The script that cloned the permissions for the "httpd_t" type was run before we allowed "httpd_t" admin access to all the namespace types. Therefore, neither "dokuwiki_t" nor any of the "dokuwiki_<$grp>_t" domains inherit those permissions through the `clone_-httpd_t` macro. However, if one was to run that script to generate the clone macro after the loading of the latest version of the `dokuwiki` module, the macro would include and assign admin permissions to all dokuwiki group types on all namespace types. This illustrates the potential dangers of domain cloning, and is probably why such a feature is not found in the SELinux policy syntax.

Now, in the browser, refresh the DokuWiki page, there should be no error. Login as `admin` – the SELinux context at the bottom of the page should state that the type of the process serving the page was "dokuwiki_admin_t". Logout – the SELinux context should have the type as "dokuwiki_t". Now login again as `setest` – the type should be `dokuwiki_-developer_t`.

Now, while logged in as `setest`, go to `http://localhost/dokuwiki-2010-11-07/doku.php?id=devel:pagex` in the web browser. As a member of the developer group, **setest** has edit privileges to that namespace (hence the **Edit this page** button being present).


Activities
Firefox
Fri 00:07
setest
devel:pagex [SETest] – Mozilla Firefox
File Edit View History Bookmarks Tools Help
devel:pagex [SETest]
localhost/dokuwiki-2010-11-07/doku.php?id=devel:pagex
Google
[[devel:pagex]]
SETEST
Edit this page
Old revisions
Recent changes
Search
Trace: » pagey » start » pagex

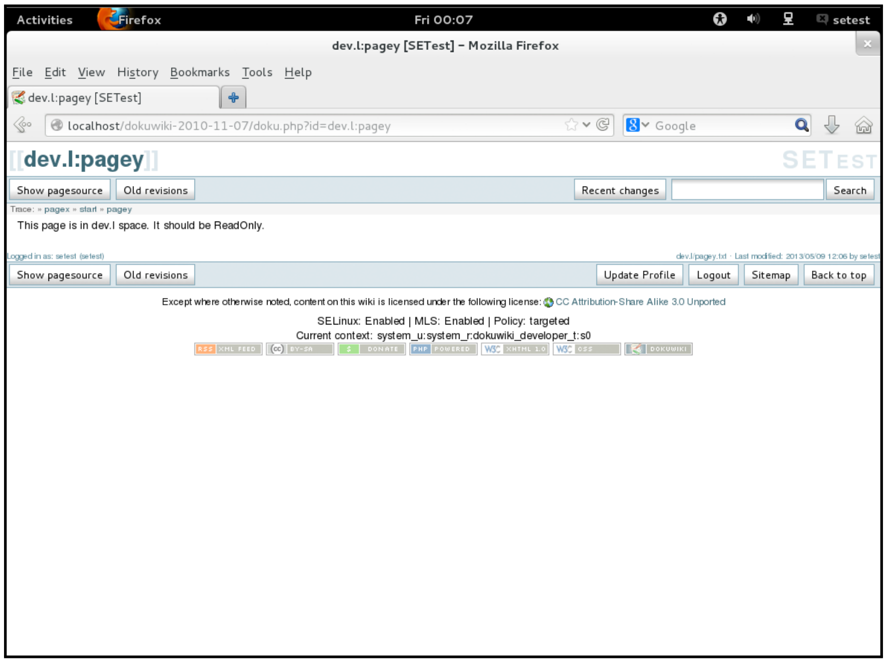
This page is in devel namespace. It should be ReadWrite.

Logged in as: setest (setest)
devel/pagex.txt · Last modified: 2013/05/10 03:11 by setest

Except where otherwise noted, content on this wiki is licensed under the following license: CC Attribution-Share Alike 3.0 Unported

SELinux: Enabled | MLS: Enabled | Policy: targeted
Current context: system_u:system_r:dokuwiki_developer_t:s0

Now, go to `http://localhost/dokuwiki-2010-11-07/doku.php?id=dev.l:pagey`. As a member of developer group, **setest** is supposed to have only read privileges to this namespace. Indeed the **Edit this page** button is not there present. Instead there is only the **Show pagesource** option.


Activities
Firefox
Fri 00:07
setest
dev.l:pagey [SETest] – Mozilla Firefox
File Edit View History Bookmarks Tools Help
dev.l:pagey [SETest]
localhost/dokuwiki-2010-11-07/doku.php?id=dev.l:pagey
Google
[[dev.l:pagey]]
SETEST
Show pagesource
Old revisions
Recent changes
Search
Trace: » pagex » start » pagey

This page is in dev.l space. It should be ReadOnly.

Logged in as: setest (setest)
dev.l/pagey.txt · Last modified: 2013/05/09 12:06 by setest
Show pagesource
Old revisions
Update Profile
Logout
Sitemap
Back to top

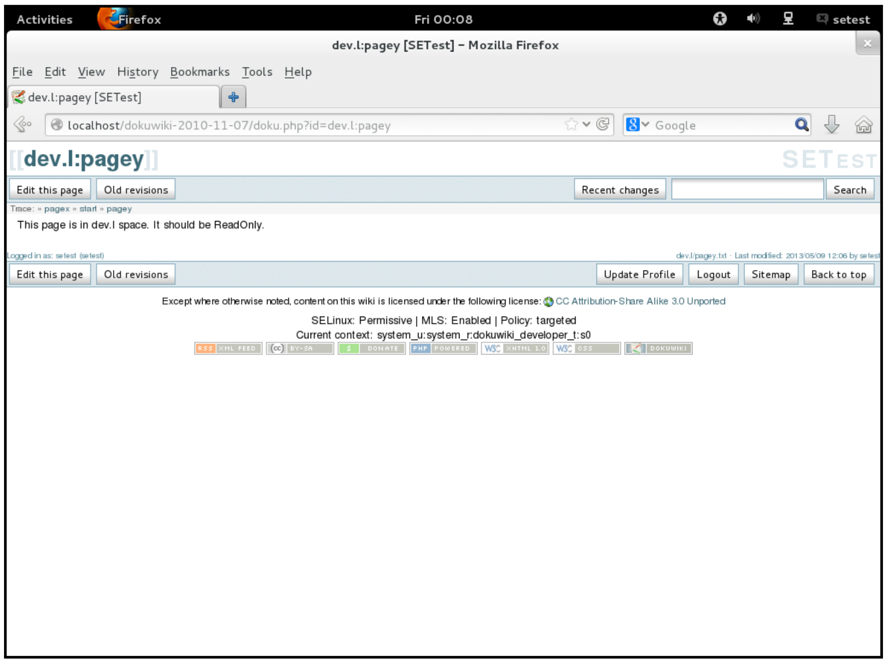
Except where otherwise noted, content on this wiki is licensed under the following license: CC Attribution-Share Alike 3.0 Unported

SELinux: Enabled | MLS: Enabled | Policy: targeted
Current context: system_u:system_r:dokuwiki_developer_t:s0

Recall that before the SELinux `dokuwiki` module was setup, this page was (due to a bug) editable. To make sure that it is indeed SELinux that is preventing DokuWiki from allowing unauthorised editing, switch SELinux to permissive mode.

```
$ sudo set enforce 0
```

Refresh the page. The footnote next to SELinux label should say *Permissive* instead of *Enabled*. The page should be editable now.


Activities
Firefox
Fri 00:08
setest
dev.l:pagey [SETest] – Mozilla Firefox
File Edit View History Bookmarks Tools Help
dev.l:pagey [SETest]
localhost/dokuwiki-2010-11-07/doku.php?id=dev.l:pagey
Google
[[dev.l:pagey]]
SETEST
Edit this page
Old revisions
Recent changes
Search
Trace: » pagex » start » pagey

This page is in dev.l space. It should be ReadOnly.

Logged in as: setest (setest)
dev.l/pagey.txt · Last modified: 2013/05/09 12:06 by setest
Edit this page
Old revisions
Update Profile
Logout
Sitemap
Back to top

Except where otherwise noted, content on this wiki is licensed under the following license: CC Attribution-Share Alike 3.0 Unported

SELinux: Permissive | MLS: Enabled | Policy: targeted
Current context: system_u:system_r:dokuwiki_developer_t:s0

So it is definitely SELinux that is guarding against the vulnerability. Switch SELinux enforcing mode back on.

```
$ sudo set enforce 1
```

You might ask, how come DokuWiki changes the button caption to reflect the correct permissions when SELinux is enabled? Does't the ACL operate independently of SELinux? (in which case we would expect the button to still provide the option of editing, but some kind of error would occur when attempting to change the page content). What is happening is that DokuWiki is checking the access permissions for the target page's file before displaying the page, and thus – since SELinux forbids write access – DokuWiki changes the page view accordingly.

4.12 Discussion

It may seem that in this chapter we have gone through a lot of trouble to fix a known (and fixed) DokuWiki vulnerability by using SELinux. The point, of course, is that this sort of double-hulled approach security has the potential to protect against unknown vulnerabilities, too. Not only that, but SELinux can be used to detect such vulnerabilities. For instance, in the current setup, whenever a DokuWiki user from the `developer` group goes to a page in the `dokuwiki_content_dev.l` space, SELinux generates the following error:

```
$sudo cat /var/log/audit/audit.log | grep AVC
.
.
.
type=AVC msg=audit(1403840178.646:400): avc:  denied  { write } for  pid=2163 comm="httpd"
    name="pagey.txt" dev="dm-0" ino=799491 scontext=system_u:system_r:dokuwiki_developer_t:
    s0 tcontext=system_u:object_r:dokuwiki_content_dev_l_type:s0 tclass=file
.
.
.
```

This error is a an indication that the ACL allows something that SELinux does not, at which point investigation can be carried out to determine whether it's SELinux that is too restrictive, or indeed if there is a problem with the ACL.

Despite being effective, the SELinux policy we have created for our example is not very flexible. Any time a new group is created, or permissions are changed in DokuWiki, the policy module would have to be modified to reflect those changes, then recompiled and reinstalled. Also, our example does not support a user being assigned to multiple groups (for that we would have to create a new domain for every possible combination of groups). However, it is not all that bad really – a script could be written that extracts information from DokuWiki about groups and permissions, and outputs appropriate policy rules. It still would need to be run every time there were changes to the ACL configuration. Possibly a clever use of categories in the MLS part of the SELinux context could make the double-hull scheme more manageable.

4.13 Summary

In this chapter we have written a custom SELinux policy module with a new set of types, labels and rules that mirror the permissions of DokuWiki's ACL security. We have configured a PHP plugin that triggers a domain transition on the *httpd* fork serving a given client depending on the client's DokuWiki credentials. Figure 4.12 shows what happens on the server (from the SELinux standpoint) before a client performs a login to DokuWiki, and then both after they login as `setest`, and after they login as `admin`. The process serving the client gets transitioned into a domain corresponding to the group of the DokuWiki user and its permissions to view or manipulate files of different SELinux types are affected appropriately.

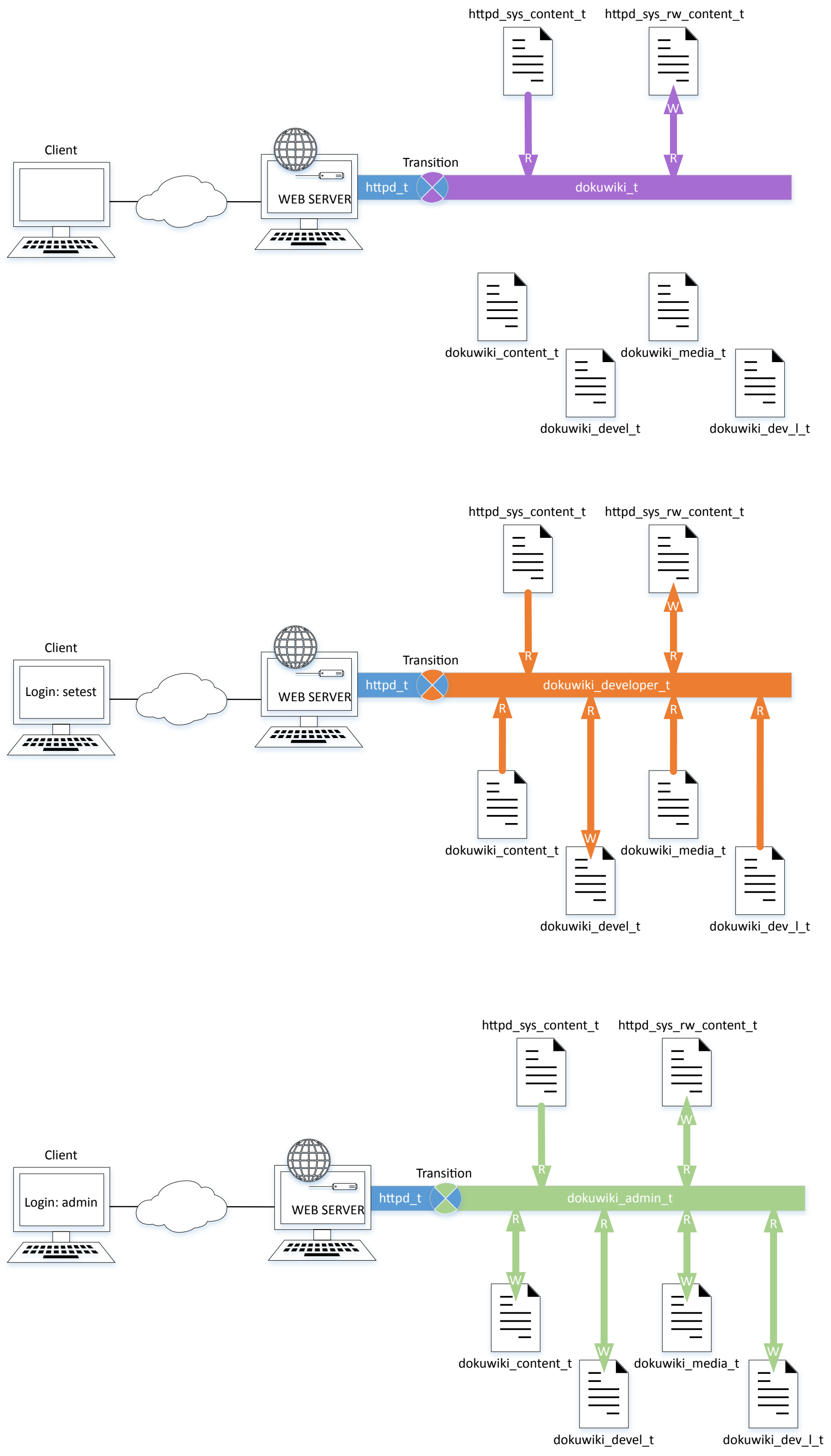


Figure 4.1: Apache server running in the "dokuwiki_t", "dokuwiki_developer_t" or "dokuwiki_admin_t" domain, depending on the login credentials of the DokuWiki user.

Bibliography

[1] P. Loscocco and S. Smalley, “Integrating flexible support for security policies into the linux operating system,” in *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference*, Berkeley, CA, USA: USENIX Association, 2001, pp. 29–42, ISBN: 1-880446-10-3. [Online]. Available: `http://www.nsa.gov/research/_files/publications/security_policies_linux_os.pdf` (visited on 07/05/2014).

[2] P. Hosek, M. Migliavacca, I. Papagiannis, D. Eyers, D. Evans, B. Shand, J. Bacon, and P. Pietzuch, “Safeweb: a middleware for securing ruby-based web applications,” in *ACM/IFIP/USENIX 12th International Middleware Conference (Middleware 2011)*, Lisbon, Portugal, Dec. 2011.

[3] (2012). Red Hat Enterprise Linux 6 Security-Enhanced Linux - User Guide, [Online]. Available: `https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/pdf/Security-Enhanced_Linux/Red_Hat_Enterprise_Linux-6-Security-Enhanced_Linux-en-US.pdf` (visited on 06/25/2014).

[4] D. Walsh. (2013). Your visual how-to guide for SELinux policy enforcement, [Online]. Available: `http://opensource.com/business/13/11/selinux-policy-guide` (visited on 06/25/2014).

[5] C. Wright, C. Cowan, S. Smalley, J. Morris, and G. Kroah-Hartman, “Linux security modules: general security support for the linux kernel,” in *Proceedings of the 11th USENIX Security Symposium*, Berkeley, CA, USA: USENIX Association, 2002, pp. 17–31, ISBN: 1-931971-00-5. [Online]. Available: `http://dl.acm.org/citation.cfm?id=647253.720287`.

[6] S. Smalley, C. Vance, and W. Salamon. (2001). Implementing SELinux as a Linux Security Module, [Online]. Available: `http://www.nsa.gov/research/_files/selinux/papers/module.pdf` (visited on 07/12/2014).

[7] D. Walsh. (2008). Understanding SELinux Process Transitions, [Online]. Available: `http://danwalsh.livejournal.com/23944.html` (visited on 06/26/2014).

[8] (2012). Typerules, [Online]. Available: `http://selinuxproject.org/page/TypeRules#type_transition_Rule` (visited on 06/26/2014).

[9] K. Kohei. (2009). Introduction of the Apache/SELinux plus, [Online]. Available: `https://code.google.com/p/sepgsql/wiki/Apache_SELinux_plus` (visited on 07/03/2014).

[10] ——, (2012). PHP-SELinux package README, [Online]. Available: `http://svn.php.net/viewvc/pecl/selinux/trunk/README?view=markup` (visited on 07/03/2014).

[11] (2014). Access Control Lists (ACL), [Online]. Available: `https://www.dokuwiki.org/acl` (visited on 07/03/2014).