SELinux Policy for Web Servers

1st Zhenisbekov Aibek
Cybersecurity bachelor's student
Astana IT University
Astana, Kazakhstan
aibek.zhenisbekov.work@gmail.com
https://orcid.org/0000-0003-3561-569X

2nd Zhunisbek Akerke
Cybersecurity bachelor's student
Astana IT University
Astana, Kazakhstan
akesha.soydere@gmail.com

I. ABSTRACT

Security-Enhanced Linux (SELinux) has emerged as a powerful tool for enforcing mandatory access control policies in various computing environments. In the context of web servers, where security is paramount due to the exposure to potential threats, the development of tailored SELinux policies becomes particularly crucial. This research paper focuses on the design, implementation, and evaluation of SELinux policies specifically crafted for web servers. Through meticulous policy development and testing, this study aims to propose SELinux policies optimised for CentOS-based web server environments. The research methodology involves a thorough analysis of security requirements and common attack vectors prevalent in web server setups, leading to the formulation of precise SELinux rules to mitigate risks effectively. Additionally, this paper discusses the intricacies of SELinux policy development, challenges encountered during policy creation, and strategies employed to address them. The outcomes of this research contribute to the advancement of SELinux policy development practices for securing web server environments, providing valuable insights for system administrators and security practitioners.

II. INTRODUCTION

In recent years, the security landscape of web servers has become increasingly complicated and challenging due to the evolving nature of cyber threats (Mallik & Nath, 2024). Web servers, serving as gateways to valuable data and services, are prime targets for malicious actors seeking to exploit vulnerabilities and compromise systems. To mitigate these risks, robust security measures beyond traditional access controls are imperative.

Security-Enhanced Linux (SELinux) has emerged as a potent solution for bolstering the security posture of Linux-based systems by providing a flexible framework for enforcing mandatory access controls. By defining security policies at the kernel level, SELinux offers granular control over system resources, enhancing resilience against a wide range of security threats.

This research paper focuses on the development of SELinux policies tailored specifically for web servers. The primary objective is to design SELinux policies that effectively mitigate common security risks associated with web server deployments while minimising the impact on system performance and usability.

The methodology employed in the study of SELinux policy for web servers involves an in-depth analysis of the Type Enforcement (TE) security model, which is central to SELinux's architecture. The TE model is utilised for the isolation of processes into domains, thereby enhancing the security of multi-service systems. This isolation is achieved through a comprehensive set of rules defined within the SELinux policy that governs the interactions between processes and resources (Radhika et al., 2020). The policy itself is modular, allowing for runtime augmentation with additional rules or modules, thus providing flexibility in security management.

The introduction of SELinux policies for web servers presents a significant opportunity to enhance the security posture of web infrastructure, providing system administrators and security practitioners with a powerful toolset to combat emerging threats. Through a combination of policy design, testing, and evaluation, this research aims to contribute to the advancement of SELinux policy development practices, thereby fostering a more secure web ecosystem.

III. LITERATURE REVIEW

Navigating the Cyber security Landscape: A Comprehensive Review of Cyber-Attacks, Emerging Trends, and Recent Developments:

This research paper by Mallik and Nath (2024) highlights the significant increase in cybercrime due to factors such as the COVID-19 pandemic and the widespread use of digital environments. The need for robust defence against various cyber-attacks is emphasised. Security challenges arise due to the emergence of technologies such as cloud computing, IoT, social media and cryptocurrencies and the trend of cyber attacks as a service. To combat current and future cyber threats, technical solutions including machine learning, deep learning, and blockchain technology are considered. This Research provides a comprehensive overview of cyberattacks, trends, and recent developments, offering a holistic view of the cybersecurity landscape (Mallik & Nath, 2024).

Consistency analysis and flow secure enforcement of SELinux policies:

Radhika et al. (2020) address the critical issue of SELinux policy consistency and propose a methodology for flow secure enforcement. Their research, published in Computers & Security, outlines a framework that ensures SELinux policies are not only consistent but also robust against unauthorised information flows. By integrating formal methods into the policy development process, Radhika et al. provide a significant contribution to enhancing system security and mitigating potential vulnerabilities inherent in policy inconsistencies.

Implementing SELinux as a Linux security module:

Implementing SELinux as a Linux Security Module by Stephen Smalley, Chris Vance, and Wayne Salamon (2002) discusses the transition from the original SELinux kernel patch to an LSM-based SELinux security module. The authors detail the general changes made to accommodate design constraints imposed by LSM and further review of SELinux controls. They highlight program execution changes, such as the introduction of the `file execute_no_trans` permission and modifications to file descriptor inheritance. The paper also addresses filesystem changes, including persistent labelling and pseudo filesystem labelling, and outlines the internal architecture of the SELinux security module. These adaptations ensure that SELinux can support a variety of non discretionary access control policies without altering the policy enforcement code, thereby maintaining a high level of security within the Linux kernel.

SELinux MLS Policy Analysis:

Hicks et al. (2010) present a comprehensive study on the SELinux mandatory access control (MAC) policy, particularly focusing on the multilevel security (MLS) model. They address the complexity of manually evaluating SELinux MLS policies and propose a logical specification modelled in Prolog for automated analysis. Their tool, PALMS, is capable of verifying information flow properties and policy compliance, ensuring that policies meet the simple security condition and the-property defined by Bell and LaPadula.

Using SELinux security enforcement in Linux-based embedded devices:

The article discusses the implementation of Security Enhanced Linux (SELinux) on the Nokia 770 Internet Tablet, highlighting the need for fine-grained access control in Linux-based embedded devices. It outlines the challenges of integrating SELinux, such as the lack of bootloader support and the need to modify the file system to accommodate extended attributes. The authors detail the process of porting SELinux to the device, including kernel modifications and policy changes, to enable more detailed policies for service providers and users. The work demonstrates the feasibility of enhancing security on embedded devices without significant performance impact, paving the way for more secure multi-user environments. This integration exemplifies the potential benefits of SELinux in protecting sensitive data and managing application permissions on mobile systems.

Methodology

IV. SELINUX POLICY

Theory:

The methodology employed in the study of SELinux policy for web servers is characterised by an in-depth analysis of the Type Enforcement (TE) security model, which is central to SELinux's architecture. The TE model is utilised for the isolation of processes into domains, thereby enhancing the security of multi-service systems. This isolation is achieved through a comprehensive set of rules defined within the SELinux policy that governs the interactions between processes and resources (Radhika et al., 2020). The policy itself is modular, allowing for runtime augmentation with additional rules or modules, thus providing flexibility in security management.

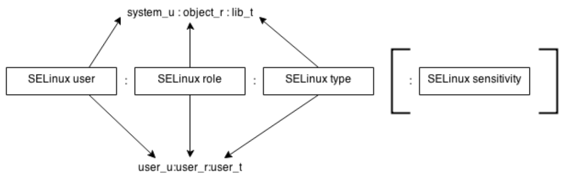


Fig. 1. Type Enforcement model

Furthermore, the role-based access control (RBAC) and multi-level security (MLS) models implemented by SELinux are examined, offering additional layers of security by restricting user roles and incorporating sensitivity levels and categories into security contexts (Radhika et al., 2020). These models ensure that only authorised entities can access or modify web server resources, thereby preventing unauthorised information disclosure or system compromise. The case studies presented provide insights into the practical use of SELinux for enhancing web application security, showcasing the policy's capabilities in action.

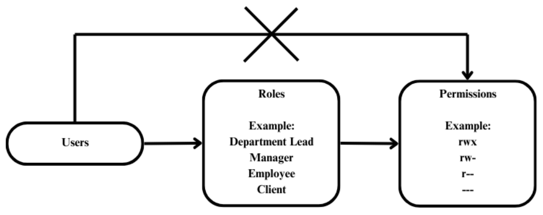


Fig. 2. Role-Based Access Control model

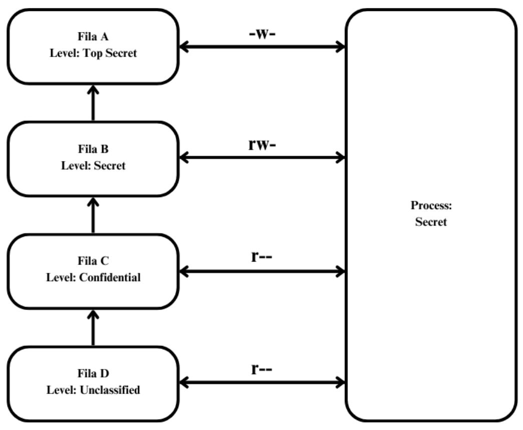


Fig. 3. Multi-Level Security model

Installation:

VirtualBox was used to run CentOS Stream 9 for all of the tutorial's exercises. With the exception of how to install Fedora Linux itself, all installation instructions necessary for the tutorial are given; they are dotted throughout this document and introduce the necessary components as needed. For the tutorial, we created the root user so we don't have to type sudo every time.

Tools:

A set of standard SELinux tools, which allow for policy investigation, is all that is needed for this chapter. To install the aforementioned tools, the following commands should be typed into the terminal:

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

Architecture:

Details about the present state of the SELinux configuration are retrieved by the scripts in the packages that were recently installed. This can be confirmed by executing the following command in 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
Memory protection checking:     actual
(secure)
Max kernel policy version:      33
```

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.

LSM:

The foundation for SELinux is the Linux Security Modules (LSM) framework, a general framework for Mandatory Access Control (MAC) on Linux. In contrast, the default security mechanism of Linux filesystems, where resource owners (files or directories) decide access permissions, is a form of Discretionary Access Control (DAC). For example, using the `chmod` command, a user can grant everyone on the system access to their home directory. MAC security, however, prevents users from making these choices, centralising access control decisions in the policy. While SELinux can be configured to allow any user to change the policy, doing so would undermine the goal of MAC, which is to centralise decision-making regarding access control.

The LSM service informs the kernel about acceptable and unacceptable actions, supplementing but not replacing the current DAC system. Configuring both SELinux and the DAC system correctly can be challenging, as they operate independently. Since Linux kernel version 2.6, hooks have been incorporated to call LSM's API at various points before executing specific operations (such as opening a file). This API provides the object class code, the action to be carried out, and details about the security contexts of the target resource and the process involved. Each kernel hook correlates a specific action with a particular class of object. The action and object class codes for a given LSM query are statically embedded in the API. LSM returns details about whether the process is permitted to perform the action under the current policy. The kernel then decides whether to follow this advice, based on SELinux's mode:
- In `enforcing` mode, the kernel heeds the LSM's advice and aborts any operation that is not permitted.
- In `permissive` mode, the kernel ignores the LSM's advice but logs any denials reported, which is useful for debugging.
- When SELinux is `disabled`, the kernel does not query the LSM at all.

To change the current mode, the `setenforce` command should be used. The options for this command to change the mode are provided below:

```
$ setenforce --help
usage:  setenforce [ Enforcing | Permissive |
1 | 0 ]
```

The effect of the `setenforce` command does not persist through reboots. SELinux can also be fully disabled if necessary. To set the SELinux mode at boot, edit the SELINUX variable in the `/etc/selinux/config` file.

```
$ 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.
# See also:
#
https://access.redhat.com/documentation/en-us
/red_hat_enterprise_linux/9/html/using_selinu
x/changing-selinux-states-and-modes_using-sel
```

```
inux#changing-selinux-modes-at-boot-time_chan
ging-selinux-states-and-modes
#
# NOTE: Up to RHEL 8 release included,
SELINUX=disabled would also
# fully disable SELinux during boot. If you
need a system with SELinux
# fully disabled instead of SELinux running
with no policy loaded, you
# need to pass selinux=0 to the kernel
command line. You can use grubby
# to persistently set the bootloader to boot
with selinux=0:
#
#    grubby --update-kernel ALL --args
selinux=0
#
# To revert back to SELinux enabled:
#
#    grubby --update-kernel ALL --remove-args
selinux
#
SELINUX=enforcing
# SELINUXTYPE= can take one of these three
values:
#     targeted - Targeted processes are
protected,
#     minimum - Modification of targeted
policy. Only selected processes are
protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

The LSM system is not limited to serving the kernel alone. Its API can be utilised by any application to query about allowed and prohibited actions. Essentially, LSM acts as a security advisor, providing policy responses that the kernel or the querying application can then enforce.

Security labels:

Similar to security badges, security labels are strings that specify a security context for a process or a system resource. From a SELinux perspective, the operating system is composed of various agents that interact with different objects. The active processes are the agents, acting as the originators of actions. The objects, which include files, directories, sockets, and other processes, are the targets of these actions. Every object and process on the system is assigned a security label; the file system implementations assist in labelling files and directories, while the kernel handles the labelling of processes.

The SELinux security label has the following format:
`<seuser>:<role>:<type>:<sensitivity>:<categories>`

The significance of each section of the label will be explored in more detail later on. For now, let's examine the current labels in a portion of the filesystem:

```
$ ls -Z
system_u:object_r:admin_home_t:s0
anaconda-ks.cfg
unconfined_u:object_r:admin_home_t:s0 Desktop
unconfined_u:object_r:admin_home_t:s0
Documents
unconfined_u:object_r:admin_home_t:s0
Downloads
unconfined_u:object_r:admin_home_t:s0 Music
unconfined_u:object_r:admin_home_t:s0
Pictures
unconfined_u:object_r:admin_home_t:s0 Public
unconfined_u:object_r:admin_home_t:s0
Templates
unconfined_u:object_r:admin_home_t:s0 Videos
```

The SELinux labels for the items in the current directory are displayed using the ls command with the -Z switch. All but one of the directories in the previous example have the label `unconfined_u:object_r:user_home_t:s0`. This label designates the following security context: sensitivity="s0", type="user_home_t", role="object_r", seuser="unconfined_u", and undefined categories. It is important to note that the DAC flags are still present.

To view the context of the active processes, take the following action:

```
$ ps -eZ
LABEL                                   PID TTY
TIME CMD
system_u:system_r:init_t:s0               1 ?
00:00:03 systemd
system_u:system_r:kernel_t:s0             2 ?
00:00:00 kthreadd
system_u:system_r:kernel_t:s0             3 ?
00:00:00 rcu_gp
system_u:system_r:kernel_t:s0             4 ?
00:00:00 rcu_par_gp
system_u:system_r:kernel_t:s0             5 ?
00:00:00 slub_flushwq
system_u:system_r:kernel_t:s0             6 ?
00:00:00 netns
system_u:system_r:kernel_t:s0            10 ?
00:00:00 mm_percpu_wq
system_u:system_r:kernel_t:s0            11 ?
00:00:05 kworker/u8:1-events_unbound
system_u:system_r:kernel_t:s0            12 ?
00:00:00 rcu_tasks_kthre
...
```

Although they differ, the context labels have the same structure.

To view the context of your shell or the process running your terminal, use the following command:

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

The labels of parent directories are inherited by new directories. Similarly, processes inherit the labels of the processes that spawned them. Modification of security labels on files, directories, and running processes is only possible if permitted by the policy.

Policy:

The policy, frequently referenced, serves as the lexicon delineating appropriate security labels, actions, and regulations governing acceptable behaviour. It exclusively outlines "allow" guidelines; anything not explicitly approved is prohibited. The LSM can query the policy in a binary format much faster than parsing the text-based representation because the policy is authored in the m4 macro language and compiled into binary format.

Policies can be either modular or monolithic, depending on their drafting. Modular policies allow for easy addition of more rules (modules) at runtime due to their design. Conversely, a monolithic policy necessitates recompilation of the entire policy for any changes, rendering it stricter and more secure. While the policy source code may seem daunting for SELinux beginners, note that the targeted policy is modular. A module is a self-contained sub-policy that may or may not rely on other modules, depending on the base module. The base module, which loads first, is where certain policy elements can be defined. Other modules can be activated or deactivated at any time, even during runtime, as long as there are no dependencies with other active modules. You can utilise the `semodule` command, as demonstrated, to view the currently active modules:

```
$ semodule -l
abrt
accountsd
acct
afs
afterburn
aiccu
aide
...
```

Domains:

The partitioning of the Linux system into multiple domains forms the core of the SELinux framework. This concept is akin to virtualization, where various services are segregated and contained within separate virtual machines. The objective of this isolation is to ensure that a security flaw in one service cannot be exploited to access another. While virtualization provides complete isolation, it can be heavy and inflexible, despite its effectiveness. SELinux, on the other hand, employs Type Enforcement (TE), which is adaptable and customizable through policy, to isolate processes and resources into domains. Domain isolation is the cornerstone of SELinux security, although there are other crucial aspects as well.

Within the policy, a dictionary comprising types, object classes, and actions is defined. Additionally, a set of guidelines outlines which types of agents are permitted to execute certain actions on objects of other types. An agent or object can be labelled with a specific type label. In the syntax of the allow rule, one type is designated as the source and another as the object of an action. Even the same type can function as both an agent and an object; for instance, a process attempting to modify its SELinux context serves as both the agent and the object involved in 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".Below is the command that displays which options to use to find the token we need:

```
$ seinfo --help
...
Component Queries:
  -a [ATTR], --attribute [ATTR]
                        Print type
attributes.
  -b [BOOL], --bool [BOOL]
                        Print Booleans.
  -c [CLASS], --class [CLASS]
                        Print object classes.
  -r [ROLE], --role [ROLE]
                        Print roles.
  -t [TYPE], --type [TYPE]
                        Print types.
  -u [USER], --user [USER]
                        Print users.
...
```

To see all the types defined by the currently loaded policy, issue the following command:

```
$ seinfo --type

Types: 5117
   NetworkManager_dispatcher_chronyc_script_t
   NetworkManager_dispatcher_chronyc_t
   NetworkManager_dispatcher_cloud_script_t
   NetworkManager_dispatcher_cloud_t
   NetworkManager_dispatcher_console_script_t
   NetworkManager_dispatcher_console_t
...
```

It's a long list, created with the 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

Classes: 135
   alg_socket
   anon_inode
   appletalk_socket
   association
   atmpvc_socket
   atmsvc_socket
   ax25_socket
   binder
   blk_file
   bluetooth_socket
   bpf
...
```

The policy defines a set of actions relevant to a specific class of object, such as a directory object corresponding to the "dir" class. The permitted actions for objects in the "dir" class can be seen by using the following command:

```
$ seinfo --class=dir -x

Classes: 1
   class dir
inherits file
{
  add_name
  remove_name
  reparent
  rmdir
  search
}
```

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.

Since the source type is always linked to a process attempting to execute an action, it can be considered analogous to the concept of a domain. For instance, the process launching your shell is identified as:

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

– is regarded as operating within the "unconfined_t" domain. Due to the following policy rule, the kernel allows the ls command to list contents of your home directory (which is of the "user_home_t" type):

```
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 };
```

From this, we can infer that processes tagged with the "httpd_t" type and operating in the "httpd_t" domain can read files and directories of the "httpd_sys_content_t" type. The targeted policy's default labeling settings confine the httpd process to the "httpd_t" domain by tagging it with the "httpd_t" type. All files located under /var/www/html have the type "httpd_sys_content_t" attached to them. Consequently, in the default configuration, the web server cannot write to files containing HTTP content, but it can read them.

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

```
$ sesearch --allow -s httpd_t
allow corenet_unlabeled_type
unlabeled_t:association { recvfrom sendto };
allow corenet_unlabeled_type
unlabeled_t:dccp_socket recvfrom;
allow corenet_unlabeled_type unlabeled_t:peer
recv;
allow corenet_unlabeled_type
unlabeled_t:rawip_socket recvfrom;
allow corenet_unlabeled_type
unlabeled_t:tcp_socket recvfrom;
allow corenet_unlabeled_type
unlabeled_t:udp_socket recvfrom;
...
```

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

```
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 } ;
```

Consequently, files and directories of the "httpd_sys_rw_content_t" type can be written to by any processes operating in the "httpd_t" domain. In the upcoming chapter, we will guide you through the process of relabeling specific /var/www/html subdirectories to enable 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

allow httpd_t file_type:dir { getattr open
search };
allow httpd_t httpd_content_type:dir { ioctl
lock read }; [ httpd_builtin_scripting ]:True
allow httpd_t httpd_sys_rw_content_t:dir {
add_name create link remove_name rename
reparent rmdir setattr unlink watch
watch_reads write }; [
httpd_builtin_scripting ]:True
...
```

Transition:

A process transitions into a new domain by changing its type in the security label. Security-wise, there would be no change if the new type's policy contained exactly the same "allow" rules as the old type's did. This is because, although the process would operate in a different domain, its permissions would remain unchanged. However, the typical intention behind changing such labels is to move to a domain with more, less, or sometimes entirely different permissions.

By default, when a process spawns another, the child process inherits its parent's context. Consequently, when launching an executable file labelled "bin_t" from a shell, the type under which your shell is currently operating will be used to identify the new process instead of "bin_t." Most of the time, this behaviour is desired, as child processes should remain within their parent's domain. However, there are scenarios where the child process must transition to a new domain. For instance, the init process, responsible for spawning various services during system boot, is confined to the "initrc_t" domain. It would be suboptimal if all newly launched services were constrained to the parent domain. Since these services are meant to be autonomous, it would pose a security risk for them to share a domain, even momentarily before transitioning to another domain. Hence, the policy provides special syntax for specifying immediate domain transitions when a process is spawned:

```
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
system_u:object_r:crond_exec_t:s0
/usr/sbin/crond
```

The crond processes are created by that file and are launched by an init script that runs only within the "initrc_t" domain during the boot sequence. Type_transition rules are searched for using "initrc_t" as the source type and "crond_exec_t" as the target type.

```
$ sesearch -T -s initrc_t -t crond_exec_t

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 935
?   00:00:00 atd
system_u:system_r:crond_t:s0-s0:c0.c1023 938
?   00:00:00 crond
```

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

Domain transitions may seem somewhat risky initially. However, it's important to note that the policy must explicitly allow a specific transition. Typically, when a process transitions, it cannot exit that domain and enter another. Transitions become akin to a one-time, one-way pass through into a secured area when configured in this manner by policy. Processes should ideally be transitioned upon spawn so that they only ever operate within one domain. Moreover, a transition can be executed at any point during the process's execution. However, this poses more risk because the process will need to operate in two distinct domains for a period of time.

Undefined:

Processes identified by the type "unconfined_t" are said to be operating in the unconfined domain. The guiding principle of the targeted policy is to restrict access to specific services, such as `httpd`. The unconfined domain, which the policy allows for nearly anything, is where processes involving direct user access, like the shell or X-windows, operate. As a result, the SELinux policy focuses on specific services and allows non-technical users to use their Fedora system for standard desktop tasks—often without their knowledge—while carefully regulating the behaviour of a selection of system services.

Role-Based Access:

As previously mentioned, SELinux also utilises a security model called Role Based Access Control (RBAC), which enables different users to have varying permissions based on the roles they can be assigned. It's important to note that Linux users and groups are not equivalent to SELinux users and roles. In this document, SELinux users will be referred to as "seusers" to prevent confusion.

TE is integrated with the RBAC model. A policy defines the roles and users within the set, the domains (types) accessible to a role, and the roles that a specific user can assume. This suggests that a particular user is restricted to specific domains metaphorically. The policy precisely outlines which combinations of `seuser`, role, and type are permissible for legitimate security labels. Remember that the format of a security label is as follows:
`<user>:<role>:<type>:<sensitivity>:<categories>`
To see the list of `seusers` defined by the current policy, issue the following command:

```
$ seinfo --user

Users: 8
   guest_u
   root
   staff_u
   sysadm_u
   system_u
   unconfined_u
   user_u
   xguest_u
```

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

```
$ seinfo --role

Roles: 14
   auditadm_r
   dbadm_r
   guest_r
   logadm_r
   nx_server_r
   object_r
   secadm_r
   staff_r
   sysadm_r
   system_r
   unconfined_r
   user_r
   webadm_r
   xguest_r
```

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

```
$ seinfo -x --user=sysadm_u

Users: 1
   user sysadm_u roles sysadm_r level s0
range s0 - s0:c0.c1023;
```

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

```
$ semanage user -l

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

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
```

Linux `users` and SELinux `seusers` typically do not directly interact. However, in certain scenarios, such as when a user logs into a terminal or an X-windows display manager, it's logical to derive the `seuser` from the system user. Like all processes, the X-windows login interface and the login shell are labelled with SELinux contexts. It follows that the identity of the user should determine the security label of a shell operating on their behalf under Linux. Therefore, there exists a mapping between `users` and `seusers` for this purpose. You can execute the following command to view this mapping:

```
$ 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      *
```

The default mapping designates only two system users: `root` and `system_u`. They are mapped to the "unconfined_u" and "system_u" `seusers`, respectively. All other `users` are assigned the `__default__` mapping, which maps to the "unconfined_u" `seuser`.

Multi-Level Security:

SELinux also supports a third security model called Multi-Level Security (MLS). While MLS enforcement is optional and can be enabled or disabled during policy compilation, TE and RBAC enforcement is always active. MLS security enforces a more universal type of policy since the rules are typically the same for all domains. However, it is possible to adjust MLS to some extent by adding policy exceptions linked to specific domain types.

In SELinux, sensitivity levels and categories are introduced to the security context by MLS:

- Sensitivity levels are considered as levels of security clearance, which can be assigned to processes and resources. These levels are hierarchical, ranging from low to high clearance. Processes labeled with a certain sensitivity level will only be granted read rights to files labeled with the same or lower sensitivity level. However, writing to files of a lower sensitivity level will not be permitted for the same process. This ensures that an entity with high security clearance cannot share information through resources that can be read by an entity with lower clearance. The unintentional (and intentional) leakage of confidential data through the system is thus prevented.
- Categories provide a convenient means of fine-tuning available permissions while remaining in the same domain. Occasionally, it becomes necessary to assign different permissions to different instances of the same process. For example, HTTPS might require different read/write permissions based on the user accessing the website (determined by login credentials, source IP, etc.). Creating different domains for this purpose can be somewhat cumbersome (although this is precisely what will be done in this tutorial). By assigning a set of categories to processes and resources, 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.

MLS is enabled in the targeted flavour of the targeted policy (as well as in the mls flavour). To view the list of defined sensitivities in the current policy, the `seinfo` command can be issued as follows:

```
$ seinfo --sensitivity

Sensitivities: 1
   s0
```

To list all the categories, type:

```
$ seinfo --category

Categories: 1024
   c0
   c1
   c2
   c3
...
   c1023
```

Thus, only one sensitivity level, "s0", exists, while 1024 categories—from "c0" to "c1023"—are present. The reason for having only one sensitivity level is that once MLS is enabled, sensitivity must be specified in the security label, despite the fact that the targeted flavour of the targeted policy is primarily concerned with utilising the categories provided by MLS. When MLS is disabled, sensitivity or categories are not specified in the label format:

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

When MLS is enabled, the format is:

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

However, the `<categories>` portion can still be omitted, indicating that the object or process with the label is not included in any category.

The lowest and maximum clearance levels of the object or process are indicated in the `<sensitivity>` section of the label. The level hierarchy is established by the policy. Sensitivity="s0-s0" occurs when there is only one defined sensitivity level, meaning that the highest and lowest clearances are the same.

For instance, it might be found:

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

A more straightforward specification style is allowed when the lowest and highest clearance levels are identical: sensitivity="s0". Consequently, the following label yields the exact same security context. Similarly to the example above:

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

Any number of categories or any combination of categories can be specified (as in the examples mentioned). Categories="c0" denotes a single category. For instance:

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

Commas are used to separate multiple categories, as in categories="c0,c3". The two categories designated by this example label are "c0" and "c3".

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

The following list of categories is provided: categories="c2,c5", which is equivalent to categories="c2,c3,c4,c5", indicating four categories in this example. We have previously encountered labels that enumerate every category, such as:

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

A security context lacking a category and one that specifies all categories differ in that the former can be used with objects that similarly lack a category, whereas the latter can be used with any combination of categories, even if none are specified.

V. WEB SERVER ON SELINUX

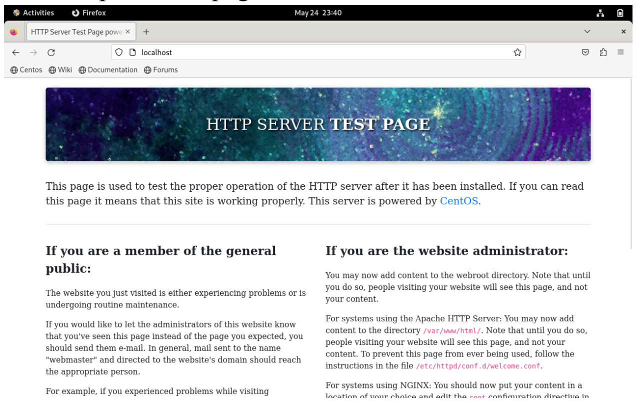
This chapter will cover the installation and protection of DokuWiki (https://www.dokuwiki.org), a wiki web server that stores documents in files on its back end. Due to its relative simplicity, but still requiring more privileges than a static website, DokuWiki serves as a good starting point when working with the current SELinux policy.

Installation:

Before commencing this exercise, the Apache server with PHP support must be installed using command below:

```
$ yum install httpd php
$ systemctl start httpd.service
$ systemctl enable httpd.service
```

To verify if Apache is up and running, attempt to visit localhost in your preferred web browser. You should see the Apache test page.



HTTP SERVER **TEST PAGE**

This page is used to test the proper operation of the HTTP server after it has been installed. If you can read this page it means that this site is working properly. This server is powered by CentOS.

If you are a member of the general public:

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

If you are the website administrator:

You may now add content to the webroot directory. Note that until you do so, people visiting your website will see this page, and not your content.

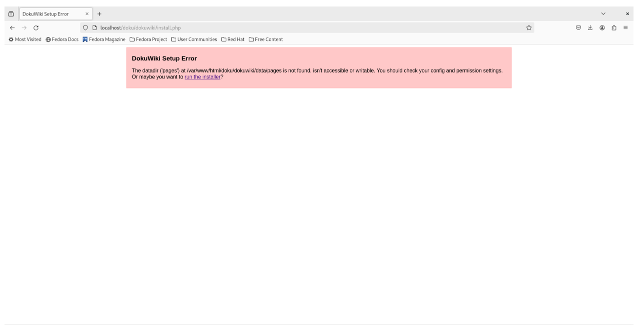
For systems using the Apache HTTP Server: You may now add content to the directory `/var/www/html/`. Note that until you do so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the file `/etc/httpd/conf.d/welcome.conf`.

For systems using NGINX: You should now put your content in a

Fig. 4. Apache localhost

DokuWiki:

Download DokuWiki Anteater – this specific (quite old now) version is required because it has some vulnerabilities that will be later utilised to demonstrate SELinux actions:



DokuWiki Setup Error

The datadir ('pages') at /var/www/html/doku/dokuwiki/data/pages is not found, isn't accessible or writable. You should check your config and permission settings. Or maybe you want to run the installer?

Fig. 5. Permission denied

Several errors are present, all of which are related to the fact that the files and subdirectories in /var/www/html are read-only by default. DokuWiki requires write permissions in a few subdirectories. It's important to consider both SELinux and the standard filesystem DAC permissions as security systems. Since it's usually preferable to handle them one at a time, let's start by setting SELinux to permissive mode. While in permissive mode, the kernel disregards the LSM's recommendations, but SELinux remains present and is consulted regarding permissions.

```
$  setenforce 0
permissive
```

After refreshing the page, the errors should still be there. This indicates that SELinux is being ignored, meaning that DAC permissions are not in order.

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

```
$ ls -l /var/www/html/doku/
total 4
drwxr-xr-x. 8 root root 4096 May 25 08:08
dokuwiki
```

To assign DokuWiki to the Apache group, of which httpd is a member, we will first modify the DAC permissions.

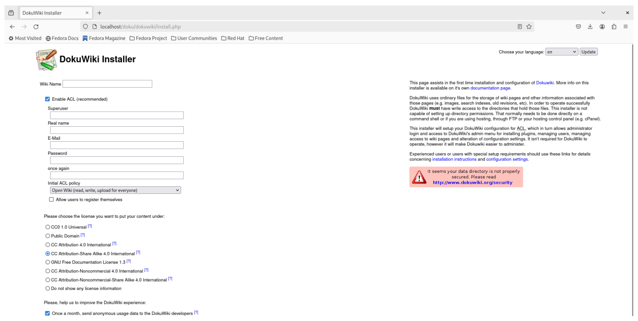
```
$ ls -l /var/www/html/doku/dokuwiki/
total 92
drwxr-xr-x.  2 root root  4096 May 25 08:08
bin
drwxr-xr-x.  2 root root  4096 May 25 08:08
conf
-rw-r--r--.  1 root root 18092 May 25 08:08
COPYING
drwxr-xr-x. 13 root root  4096 May 25 08:08
data
-rw-r--r--.  1 root root  3651 May 25 08:08
doku.php
-rw-r--r--.  1 root root  2034 May 25 08:08
feed.php
drwxr-xr-x. 23 root root  4096 May 25 08:08
inc
-rw-r--r--.  1 root root  2529 May 25 08:08
index.php
-rw-r--r--.  1 root root 21176 May 25 08:08
install.php
drwxr-xr-x.  8 root root  4096 May 25 08:08
lib
-rw-r--r--.  1 root root   308 May 25 08:08
README
-rw-r--r--.  1 root root   862 May 25 08:08
SECURITY.md
drwxr-xr-x. 11 root root  4096 May 25 08:08
vendor
-rw-r--r--.  1 root root    19 May 25 08:08
VERSION
```

Currently, only the owner is permitted to write to any of these directories by DAC. Let's modify the group permissions to enable writing for members of the Apache group as well.

```
$ chown -R apache:apache
/var/www/html/doku/dokuwiki
$ ls -l /var/www/html/doku/
total 4
drwxr-xr-x. 8 liveuser apache 4096 May 25
08:08 dokuwiki
```

```
$ chmod -R g+w
/var/www/html/doku/dokuwiki/conf/
$ chmod -R g+w
/var/www/html/doku/dokuwiki/data/
$ ls -l /var/www/html/doku/dokuwiki/
total 92
drwxr-xr-x.  2 liveuser apache  4096 May 25
08:08 bin
drwxrwxr-x.  2 liveuser apache  4096 May 25
08:08 conf
-rw-r--r--.  1 liveuser apache 18092 May 25
08:08 COPYING
drwxrwxr-x. 13 liveuser apache  4096 May 25
08:08 data
-rw-r--r--.  1 liveuser apache  3651 May 25
08:08 doku.php
-rw-r--r--.  1 liveuser apache  2034 May 25
08:08 feed.php
drwxr-xr-x. 23 liveuser apache  4096 May 25
08:08 inc
-rw-r--r--.  1 liveuser apache  2529 May 25
08:08 index.php
-rw-r--r--.  1 liveuser apache 21176 May 25
08:08 install.php
drwxr-xr-x.  8 liveuser apache  4096 May 25
08:08 lib
-rw-r--r--.  1 liveuser apache   308 May 25
08:08 README
-rw-r--r--.  1 liveuser apache   862 May 25
08:08 SECURITY.md
drwxr-xr-x. 11 liveuser apache  4096 May 25
08:08 vendor
-rw-r--r--.  1 liveuser apache    19 May 25
08:08 VERSION
```

Refresh the page http://localhost/doku/dokuwiki/install.php, and you should get:



DokuWiki Installer

Choose your language: en

Wiki Name

[✓] Enable ACL (recommended)

Superuser

Real name

E-Mail

Password

once again

Initial ACL policy: Open Wiki (read, write, upload for everyone)

[ ] Allow users to register themselves

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

- CC0 1.0 Universal [?]
- Public Domain [?]
- CC Attribution 4.0 International [?]
- [✓ CC Attribution-Share Alike 4.0 International] [?]
- GNU Free Documentation License 1.3 [?]
- CC Attribution-Noncommercial 4.0 International [?]
- CC Attribution-Noncommercial-Share Alike 4.0 International [?]
- Do not show any license information

Please, help us to improve the DokuWiki experience:

[✓] Once a month, send anonymous usage data to the DokuWiki developers [?]

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 http://www.dokuwiki.org/security

Fig. 6. DokuWiki installer

Other than a warning about the data directory not being properly secured, it seems that this will be sufficient to get DokuWiki up and running. The warning pertains to the ability of remote users to access DokuWiki's directories directly through Apache. An appropriate .htaccess configuration can secure this, but for the purposes of this tutorial, we won't worry about it.

SELinux permissions:

After DAC permissions have been granted, let's move on to SELinux. For now, the SELinux mode won't be altered. First, open /var/log/audit/audit.log in your browser and refresh the DokuWiki install page. An AVC message similar to this one should appear near the end of the file:

```
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 log rejections from SELinux. These denials are not impacting the system's functionality when it is in permissive mode, but they are still recorded. To ensure that access through DokuWiki is the reason for these messages, let's turn enforcing mode back on.

```
$ setenforce 1
enforcing
```

Once again, refresh the install page. The errors regarding the writability of the data and conference directories should be restored. The issue is clearly stated in the AVC messages: the directory of type "httpd_sys_content_t" is not write-permitted by the source context of type "httpd_t". The AVC message also indicates that the httpd process is the cause of the issue. We can verify this by enumerating every process that is active in the "httpd_t" domain.

```
$ ps -eZ | grep httpd_t
system_u:system_r:httpd_t:s0      38654 ?
00:00:00 php-fpm
system_u:system_r:httpd_t:s0      38655 ?
00:00:00 php-fpm
system_u:system_r:httpd_t:s0      38656 ?
00:00:00 php-fpm
system_u:system_r:httpd_t:s0      38657 ?
00:00:00 php-fpm
system_u:system_r:httpd_t:s0      38658 ?
00:00:00 php-fpm
system_u:system_r:httpd_t:s0      38659 ?
00:00:00 php-fpm
system_u:system_r:httpd_t:s0      40583 ?
00:00:00 httpd
system_u:system_r:httpd_t:s0      40586 ?
00:00:00 httpd
system_u:system_r:httpd_t:s0      40588 ?
00:00:00 httpd
system_u:system_r:httpd_t:s0      40589 ?
00:00:00 httpd
system_u:system_r:httpd_t:s0      40590 ?
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/doku/dokuwiki/
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 bin
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 conf
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 COPYING
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 data
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 doku.php
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 feed.php
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 inc
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 index.php
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 install.php
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 lib
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 README
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 SECURITY.md
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 vendor
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 VERSION
```

"httpd_sys_content_t" is the type of all the files and subdirectories. Let's check if there are any rules in the policy that use "httpd_t" as the source type and "httpd_sys_content_t" as the target type, just for kicks.

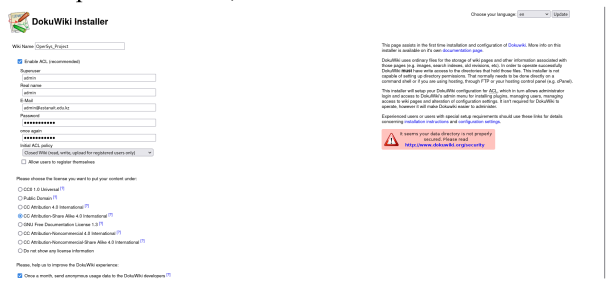
```
$ sudo chcon -t httpd_sys_rw_content_t
/var/www/html/doku/dokuwiki/conf
$ sudo chcon -R -t httpd_sys_rw_content_t
/var/www/html/doku/dokuwiki/data
```

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 -lZ /var/www/html/doku/dokuwiki/
total 92
drwxr-xr-x.  2 apache apache
unconfined_u:object_r:httpd_sys_rw_content_t:
```

```
s0  4096 May 25 08:08 bin
drwxr-xr-x.  2 apache apache
unconfined_u:object_r:httpd_sys_rw_content_t:
s0  4096 May 25 09:40 conf
-rw-r--r--.  1 apache apache
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 18092 May 25 08:08 COPYING
drwxr-xr-x. 13 apache apache
unconfined_u:object_r:httpd_sys_rw_content_t:
s0  4096 May 25 08:08 data
-rw-r--r--.  1 apache apache
unconfined_u:object_r:httpd_sys_rw_content_t:
s0  3651 May 25 08:08 doku.php
-rw-r--r--.  1 apache apache
unconfined_u:object_r:httpd_sys_rw_content_t:
s0  2034 May 25 08:08 feed.php
drwxr-xr-x. 23 apache apache
unconfined_u:object_r:httpd_sys_rw_content_t:
s0  4096 May 25 08:08 inc
-rw-r--r--.  1 apache apache
unconfined_u:object_r:httpd_sys_rw_content_t:
s0  2529 May 25 08:08 index.php
-rw-r--r--.  1 apache apache
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 21176 May 25 08:08 install.php
drwxr-xr-x.  8 apache apache
unconfined_u:object_r:httpd_sys_rw_content_t:
s0  4096 May 25 08:08 lib
-rw-r--r--.  1 apache apache
unconfined_u:object_r:httpd_sys_rw_content_t:
s0   308 May 25 08:08 README
-rw-r--r--.  1 apache apache
unconfined_u:object_r:httpd_sys_rw_content_t:
s0   862 May 25 08:08 SECURITY.md
drwxr-xr-x. 11 apache apache
unconfined_u:object_r:httpd_sys_rw_content_t:
s0  4096 May 25 08:08 vendor
-rw-r--r--.  1 apache apache
unconfined_u:object_r:httpd_sys_rw_content_t:
s0    19 May 25 08:08 VERSION
```

If you refresh the install page in your browser, there should no longer be any errors. Proceed to set up DokuWiki as per the instructions on the page. Name your wiki, activate ACL, set up a superuser account, and ensure that the ACL policy is set to "Closed Wiki." Once you have completed the task, click the Save button.



DokuWiki Installer

Wiki Name: OperSys_Project

[✓] Enable ACL (recommended)

Superuser: admin
Real name: admin
E-Mail: admin@astanait.edu.kz
Password: ●●●●●●●●●●●●
once again: ●●●●●●●●●●●●
Initial ACL policy: Closed Wiki (read, write, upload for registered users only)
[ ] Allow users to register themselves

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

- ○ CC0 1.0 Universal [?]
- ○ Public Domain [?]
- ○ CC Attribution 4.0 International [?]
- ● CC Attribution-Share Alike 4.0 International [?]
- ○ GNU Free Documentation License 1.3 [?]
- ○ CC Attribution-Noncommercial 4.0 International [?]
- ○ CC Attribution-Noncommercial-Share Alike 4.0 International [?]
- ○ Do not show any license information

Please, help us to improve the DokuWiki experience:

[✓] Once a month, send anonymous usage data to the DokuWiki developers [?]

Choose your language: en Update

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.

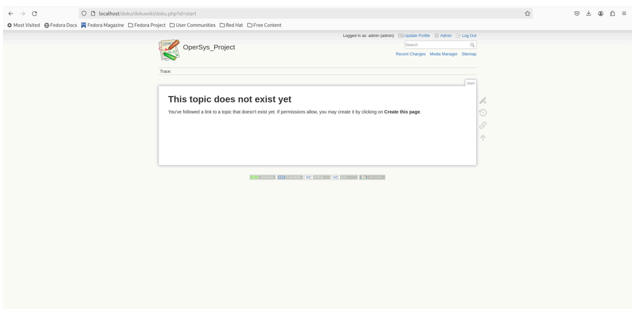
DokuWiki uses ordinary files for the storage of wiki pages and other information associated with those pages (e.g. images, search indexes, old revisions, etc). In order to operate successfully DokuWiki **must** have write access to the directories that hold those files. This installer is not capable 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 http://www.dokuwiki.org/security

Fig. 7. Web Server installing



localhost/doku/dokuwiki/doku.php?id=start

Most Visited Fedora Docs Fedora Magazine Fedora Project User Communities Red Hat Free Content

Logged in as: admin (admin) Update Profile Admin Log Out

OperSys_Project

Recent Changes Media Manager Sitemap

Trace:

start

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 clicking on **Create this page**.

Fig. 8. Working Web Server

VI. DOKUWIKI WITH DOUBLE HULL SECURITY

Installation:

The "httpd_t" domain is always operated in by the httpd process, or SELinux Apache, in the current configuration. A httpd fork, potentially in a different SELinux domain, will be required by every client request for the double hull setup. Additionally, the SELinux API must be accessed using PHP.

Install the required components, which provides SELinux support for Apache:

```
$ dnf install httpd-devel
selinux-policy-devel
$ dnf install php-devel libselinux-devel gcc
```

This enables requests from various clients to be handled by worker processes operating in different SELinux domains. SELinux contexts for clients can be configured in several ways, such as based on the client's IP address, the requested URL, or HTTP authentication. In this tutorial, PHP will be used to initiate context transitions, so the configuration file will not be changed and should only contain the following option:

```
selinuxServerDomain *: s0
```

An extra package from the PECL collection, which is available at http://pecl.php.net/package/selinux, needs to be installed to enable SELinux API calls from PHP. The most recent version of the package (as of the time of writing) can be installed using the following command and to order for the new changes to take effect, restart the `httpd` service::

```
$ wget
https://pecl.php.net/get/selinux-0.6.0.tgz
$ tar -zxvf selinux-0.6.0.tgz
$ cd selinux-0.6.0/
$ phpize
$ ./configure
$ make
$ make install
$ echo "extension=selinux.so" | sudo tee
/etc/php.d/selinux.ini
$ systemctl restart httpd.service
```

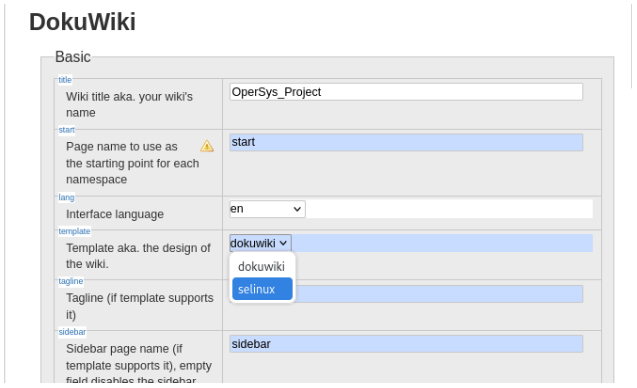
A template that displays the SELinux configuration and the serving process context can be installed directly into DokuWiki pages, thanks to our access to the SELinux API from PHP. This is highly useful for this tutorial and will provide easy access to the SELinux state. After downloading the file `dokuwiki_tpl_selinux.tgz`, unpack it into the templates directory of DokuWiki:

```
$ wget
http://www.cs.otago.ac.nz/space/selinuxtutori
al/archives/dokuwiki_tpl_selinux.tgz
$ tar -zxvf dokuwiki_tpl_selinux.tgz -C
/var/www/html/doku/dokuwiki/lib/tpl
```

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

```
$ ls -lZ /var/www/html/doku/dokuwiki/lib/tpl/
total 12
drwxr-xr-x. 5 apache   apache
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 4096 May 25 08:08 dokuwiki
-rw-r--r--. 1 apache   apache
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 2030 May 25 08:08 index.php
drwxrwxr-x. 3 liveuser liveuser
unconfined_u:object_r:httpd_sys_rw_content_t:
s0 4096 Mar 17  2013 selinux
```

When logged in as an admin to DokuWiki in your browser, navigate to the *Admin* page and select *Configuration Settings*. Under *Basic Settings*, an option for the **`selinux`** template should now be available. Choose this option and press *Save*.



Wiki title aka. your wiki's name	OperSys_Project
Page name to use as the starting point for each namespace	start
Interface language	en
Template aka. the design of the wiki.	dokuwiki dokuwiki selinux
Tagline (if template supports it)	
Sidebar page name (if template supports it), empty field disables the sidebar	sidebar

Fig. 9. Setting SELinux template

Every DokuWiki page will now display SELinux information in the footer, thanks to the new template. Navigating to the main page (http://localhost/doku/dokuwiki/doku.php) should show the standard start page with the added SELinux information in the footer.

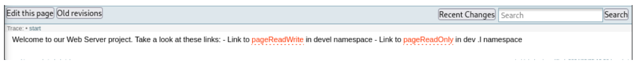
The information in the footer tells you whether SELinux is enabled or not, whether or not the optional MLS/MCS policy is enabled, the current policy name, and the SELinux context of the process that fetched this page (which at this point should be of type="httpd_t"). The template that we just installed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

Navigate to the main page of DokuWiki at http://localhost/doku/dokuwiki/doku.php while still logged in as an administrator, then select "Create this page." Include the following text:

```
Welcome to our Web Server project. Take a
look at these links:
- Link to [[devel:pageX | pageReadWrite]] in
devel namespace
- Link to [[dev.l:pageY | pageReadOnly]] in
dev.l namespace
```



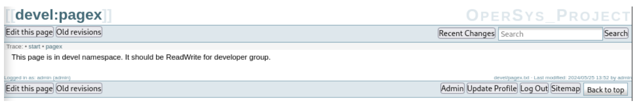
Trace: • start

Welcome to our Web Server project. Take a look at these links: - Link to pageReadWrite in devel namespace - Link to pageReadOnly in dev .l namespace

Fig. 10. First page

After clicking "Save," the mentioned text along with two links should now be visible on the start page. Click on "pageReadWrite." This will take you to the currently non-existent pageX in the DokuWiki devel namespace. After selecting "Create this page," copy and paste the following content:

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



Trace: • start • pagex

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

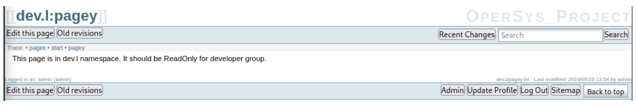
Logged in as: admin (admin)

devel/pagex.txt · Last modified: 2024/05/25 13:52 by admin

Fig. 11. devel:pagex

Click "Save." Return to http://localhost/doku/dokuwiki/doku.php and click the second link, "pageReadOnly." This will direct you to pageY in the dev.l namespace. Click "Create this page" and paste the following text:

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



Trace: • pagex • start • pagey

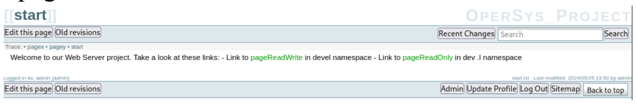
This page is in dev.l namespace. It should be ReadOnly for developer group.

Logged in as: admin (admin)

dev.l/pagey.txt · Last modified: 2024/05/25 13:54 by admin

Fig. 12. dev.l namespace

Press Save. If you go back to http://localhost/doku/dokuwiki/doku.php you should have a page that looks like this:



Trace: • pagex • pagey • start

Welcome to our Web Server project. 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: 2024/05/25 13:50 by admin

Fig. 13. Changes first page

Three pages have been created in the root, devel, and dev.l namespaces of DokuWiki, respectively: start, PageX, and Page Y. Subsequently, a DokuWiki user with read-only permissions in the dev.l space and write permissions in the devel namespace will be generated.

Access Control List:

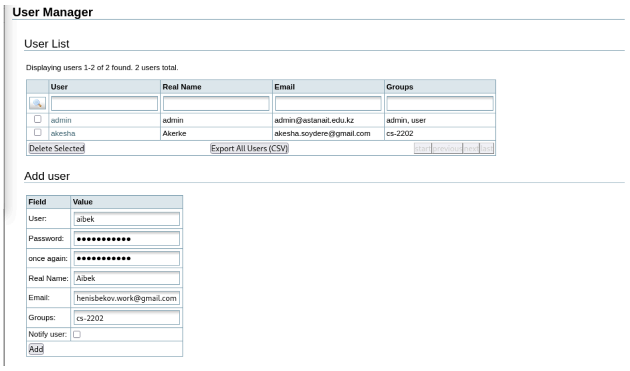
DokuWiki maintains its pages organised by namespaces, corresponding to distinct directories on the server. Users are grouped according to DokuWiki's Access Control Lists (ACL), which also enable setting group permissions for various namespaces. The following permissions are possible, ranked from least to most privileged:

• None: Only the login page will be visible; no access to DokuWiki content is permitted.
• Read: Permitted to peruse data;
• Edit: Permitted to alter already-existing pages;
• Create: Permitted to produce fresh pages;
• Admin: Permitted to manage the website;
• Upload: Permitted to upload media files;
• Delete: Permitted to delete media files.

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

While logged in to DokuWiki as an admin, navigate to the User Manager section from the Admin page. Create a new user named setest. Fill in all the necessary information and enter "developer" in the Group box – since that group doesn't exist yet, DokuWiki will create it.

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



User Manager

User List

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

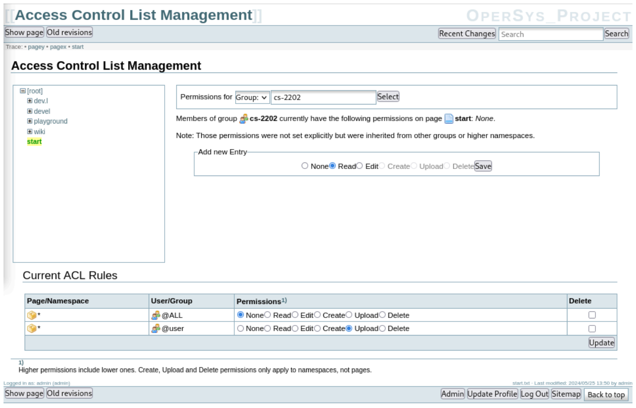
User	Real Name	Email	Groups
admin	admin	admin@astanait.edu.kz	admin, user
akesha	Akerke	akesha.soydere@gmail.com	cs-2202

Add user

Field	Value
User:	aibek
Password:	••••••••••••
once again:	••••••••••••
Real Name:	Aibek
Email:	henisbekov.work@gmail.com
Groups:	cs-2202
Notify user:	☐

Fig. 14. Adding new users.

The ACL permissions for the developer group must then be specified. From the Admin page, navigate to the Access Control List Management section. Choose the [**start**] namespace from the menu on the left. Then, under Permissions for *Group*, enter "**cs-2202**" and press *Select*. Choose "**Read**" from the *Add New Entry* window.



Trace: • pagey • pagex • start

Access Control List Management

Permissions for Group: cs-2202

Members of group cs-2202 currently have the following permissions on page start: *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

Current ACL Rules

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

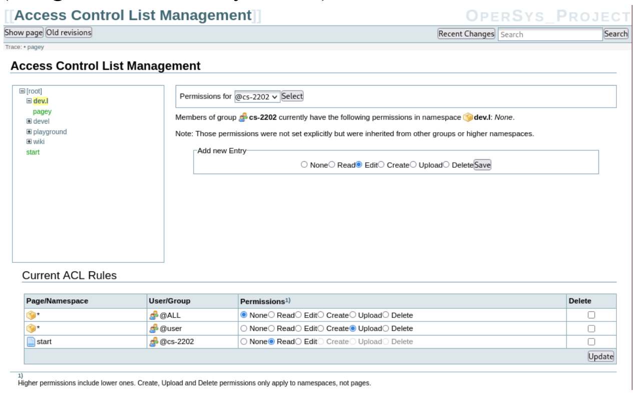
[1] Higher permissions include lower ones. Create, Upload and Delete permissions only apply to namespaces, not pages.

Logged in as: admin (admin)

start.txt · Last modified: 2024/05/25 13:50 by admin

Fig. 15. Access Control management

Click Save. A new line should appear at the bottom of the Current ACL Rules specifying that CS-2202 group has read permissions to the Start namespace (designated with the symbol *)



Trace: • pagey

Access Control List Management

Permissions for @cs-2202

Members of group cs-2202 currently have the following permissions in namespace dev.l: *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

Current ACL Rules

Page/Namespace	User/Group	Permissions[1]	Delete
*	@ALL	[✓ None] Read Edit Create Upload Delete	☐
*	@user	None Read Edit Create [✓ Upload] Delete	☐
start	@cs-2202	None [✓ Read] Edit Create Upload Delete	☐

[1] Higher permissions include lower ones. Create, Upload and Delete permissions only apply to namespaces, not pages.

Fig. 16. Add new Entry

Now, select the *dev.l* namespace from the window on the left. There's no need to type anything there as the **@cs-2202** group should now be specified in the *Permissions* for. Verify that the Add new Entry window has "**Read**" selected, then click *Save*.

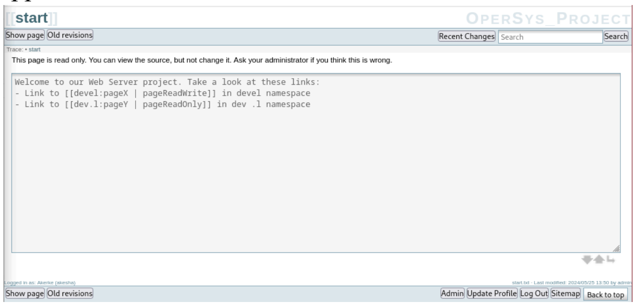
Table 6.1. Group permissions for Web Server

	ALL	CS–2202	Admin

start	None	Read	Upload
dev.l	None	Edit	Upload
devel	None	None	Upload

Role permissions checking:

Now, log out of DokuWiki and log in as setest in your browser. Edit rights for the start page (http://localhost/doku/dokuwiki/doku.php?id=start), which is in the start namespace, won't be available. Instead of the Edit this page button, a Show pagesource button will be displayed. This is due to the assigned CS-2202 group having read-only access to the start namespace. If the Show pagesource button is selected, a message indicating that the content is read-only for the current user will appear.

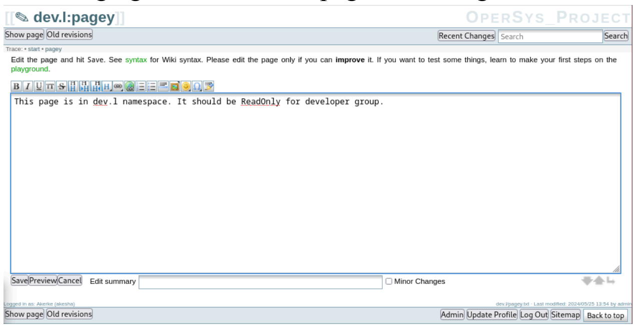


This page is read only. You can view the source, but not change it. Ask your administrator if you think this is wrong.

```
Welcome to our Web Server project. Take a look at these links:
- Link to [[devel:pageX | pageReadWrite]] in devel namespace
- Link to [[dev.l:pageY | pageReadOnly]] in dev .l namespace
```

Fig. 17. Permissions

Navigate to http://localhost/doku/dokuwiki/doku.php?id=dev.l:pagey to access PageY now. The setest user is expected to have read-only permissions to this page because it is located in the dev.l namespace. However, surprisingly, there is still a button to edit this page. You can test it by clicking on it and changing the content; the page will change as a result!



Edit the page and hit Save. See syntax for Wiki syntax. Please edit the page only if you can **improve** it. If you want to test some things, learn to make your first steps on the playground.

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

Fig. 18. Editing exists page

Double hull overview:

As SELinux hasn't been configured to mirror DokuWiki's ACL permissions for various namespaces, it currently doesn't protect against this ACL vulnerability. SELinux can be set up in various ways to address this issue. This tutorial will take the following approach: a set of SELinux types will be created, each labelling a server directory corresponding to a DokuWiki namespace. Additionally, another set of SELinux types corresponding to various DokuWiki groups will be established. When a client with credentials for a particular group is served by an httpd fork, it will switch to the domain of the associated SELinux source type. To ensure that actions for operations on "file" and "dir" objects mirror the DokuWiki ACL permissions for the corresponding group and namespace, allow rules will be created in the policy for each combination of source type and target type. Consequently, an additional layer of security around the DokuWiki system will be added.

The SELinux types and rules planned to be set up are displayed in Table 6.2; note how the first three rows match Table 6.1 (due to SELinux not permitting periods in the label syntax, "dokuwiki_content_dev_l_t" corresponds to the dev.l namespace). Empty cells indicate no allow rules. The domain created for media content is shown in the table's last row. DokuWiki stores this type of content in a different directory, and we require it to offer double hull protection for upload and delete permissions.

Table 6.2. SELinux types and allow rules that mirror DokuWiki's ACL

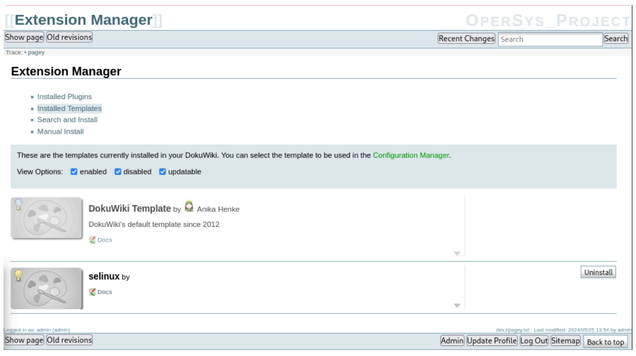
	“dokuwiki_t”	“dokuwiki_cs-2202_t”	“dokuwiki_admin_t”
“dokuwiki_content_t”		read	read,write,create,delete
“dokuwiki_content_devel_t”			read,write,create,delete
“dokuwiki_content_dev_l_t”		read, write	read,write,create,delete
“dokuwiki_content_media_t”		read	read,write,create,delete

Policy modules:

A SELinux policy module will be created to expand the policy to include new types and "allow" rules. Typically, a module contains rules related to a specific service. For instance, the apache module specifies the domain in which httpd operates. Consolidating all the information about our DokuWiki service into a custom module makes sense and serves as a great way to learn about writing SELinux policies.

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

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


[[Extension Manager]]

Trace: • pagey

Extension Manager

- Installed Plugins
- Installed Templates
- Search and Install
- Manual Install

These are the templates currently installed in your DokuWiki. You can select the template to be used in the Configuration Manager.

View Options: ✓ enabled ✓ disabled ✓ updatable

DokuWiki Template by Anika Henke

DokuWiki's default template since 2012

Docs

selinux by

Docs

Logged in as: admin (admin)

dev.l/pagey.txt · Last modified: 2024/05/25 13:54 by admin

Fig. 19. Plugin manager

VII. CONCLUSION

In conclusion, the research presented in this paper highlights the critical importance of SELinux policies for enhancing web server security. Through careful design and evaluation of SELinux policies designed for Linux-based web servers, a proactive approach to mitigating common cyber threats has been demonstrated. The integration of SELinux Type Enforcement, Role-Based Access Control and Multi-Level Security models provides a solid framework for protecting sensitive data and maintaining system integrity from unauthorised access. These results support the widespread adoption and continuous improvement of SELinux policies as an important component of cybersecurity strategies. As cyber threats continue to evolve, it is imperative that system administrators and security professionals remain vigilant and utilise advanced tools such as SELinux to protect web infrastructure. The methodologies and insights presented here contribute to a better understanding of SELinux policy design and its key role in enhancing web server security. Also, this work can be continued with more precise security techniques and new plugins on SELinux.

VIII. REFERENCE LIST

Mallick, M. A. I., & Nath, R. Navigating the Cyber security Landscape: A Comprehensive Review of Cyber-Attacks, Emerging Trends, and Recent Developments.

Radhika, B. S., Kumar, N., Shyamasundar, R. K., & Vyas, P. (2020). Consistency analysis and flow secure enforcement of SELinux policies. *Computers & Security*, *94*, 101816. https://doi.org/10.1016/j.cose.2020.101816

Smalley, S., Vance, C., & Salamon, W. (2001). Implementing SELinux as a Linux security module. *NAI Labs Report*, *1*(43), 139.

Hicks, Boniface & Rueda, Sandra & St.Clair, Luke & Jaeger, Trent & McDaniel, Patrick. (2010). A logical specification and analysis for SELinux MLS policy. ACM Transactions on Information and System Security (TISSEC). 13. 26. 10.1145/1805874.1805982.

Vogel, B., & Steinke, B. (2010). Using SELinux security enforcement in Linux-based embedded devices. *In 1st International ICST Conference on Mobile Wireless Middleware, Operating Systems and Applications.* https://doi.org/10.4108/icst.mobilware2008.2927