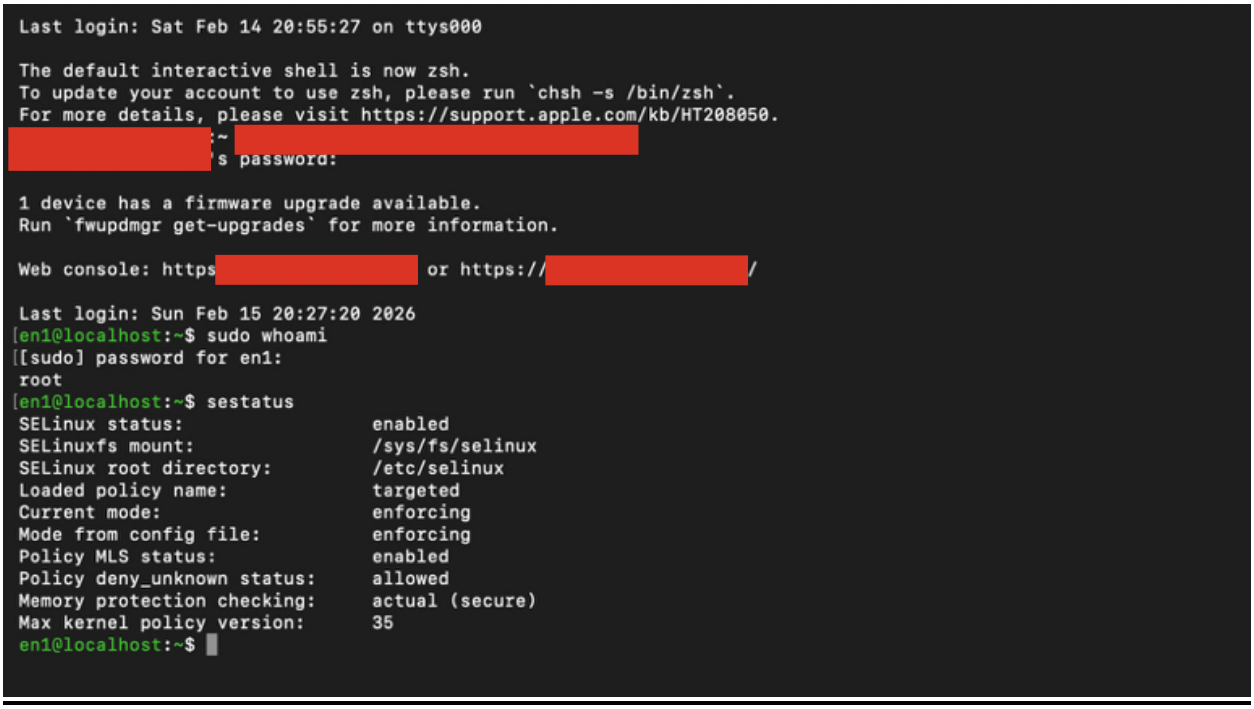
Project 3: Access Control

Part1

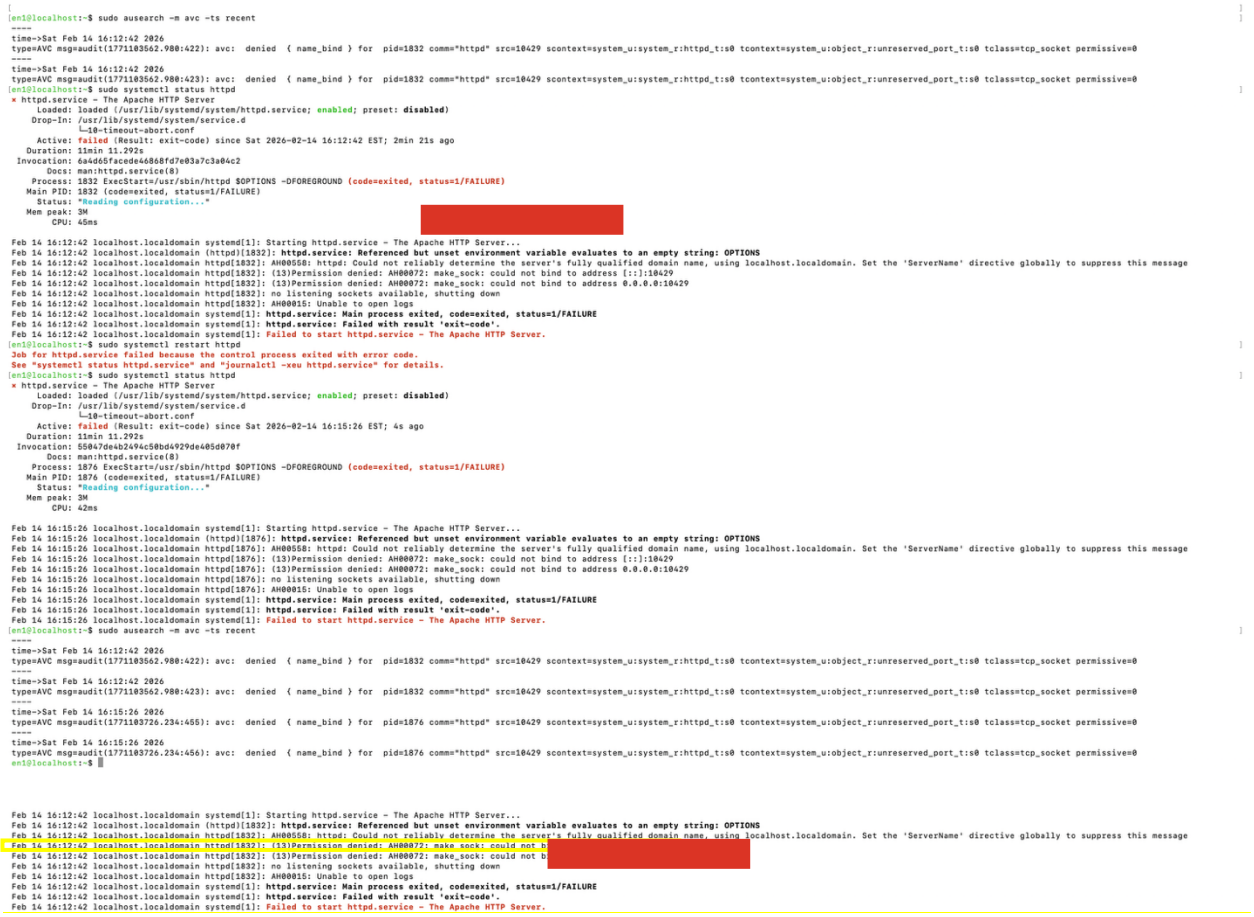
Task 0: Initial setup



The *sestatus* output confirms that SELinux is enabled and running in enforcing mode with the targeted policy loaded. This means SELinux is actively blocking policy violations for confined services like Apache.

Task 1: Changing httpd port mandatory policy

Deliverable 1.1 : Take a screenshot of your output.



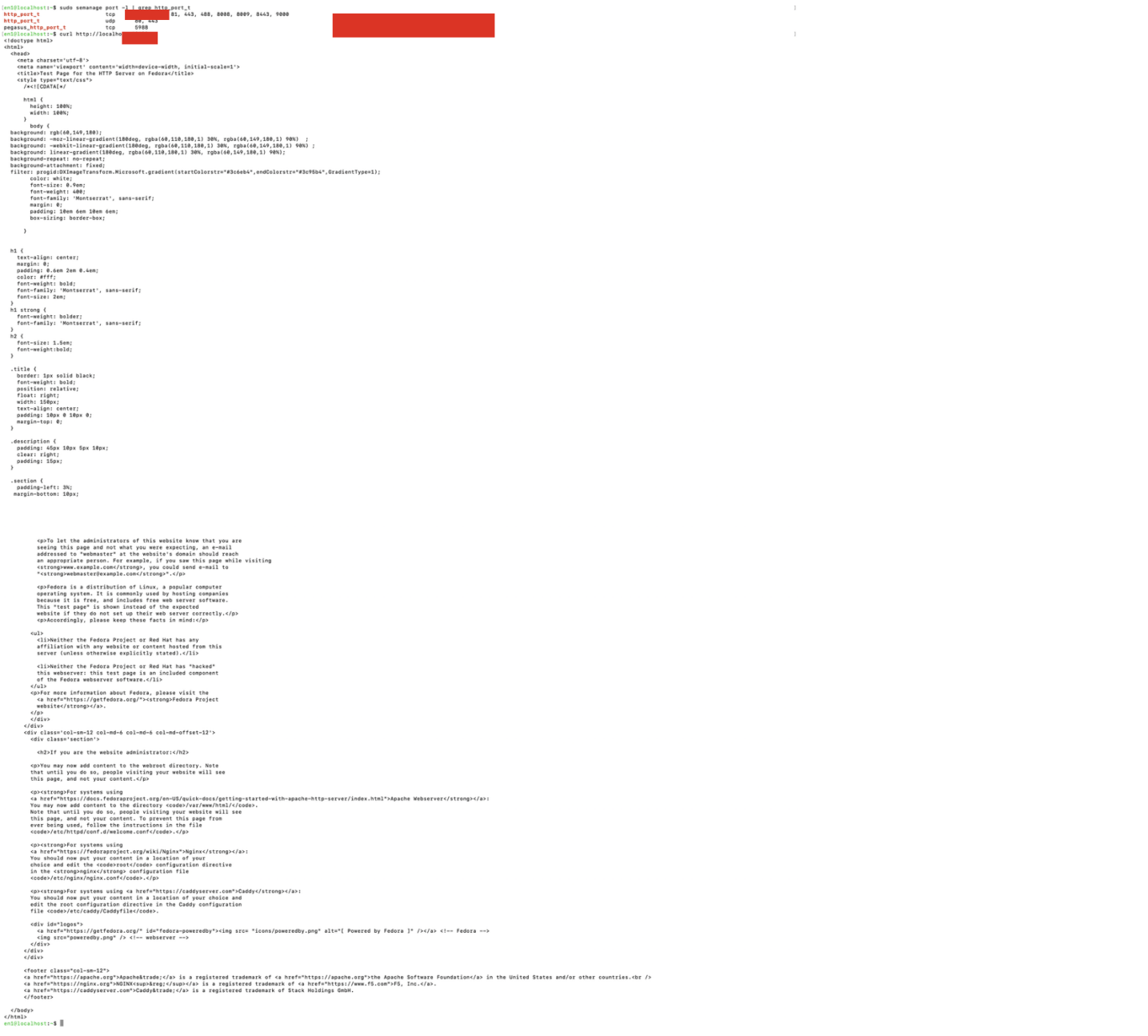
The screenshot shows the httpd service failing to restart after changing the listening port to [redacted]. The error message indicates a permission denied when attempting to bind to the new port.

Deliverable 1.2: Look into the error you are having and provide some reasoning about why it's there and what its relation is with SElinux.

- The restart failed because SELinux is blocking Apache (httpd_t) from binding to port [redacted] (my NEU-assigned port). SELinux uses type enforcement to control the actions that processes are allowed to perform. The httpd_t domain (Apache's confined type) is only permitted to bind to ports labeled http_port_t. However, po[redacted]led unreserved_port_t, so SELinux does not recognize it as an allowed HTTP port. Although Linux DAC (file permissions and ownership) would normally allow Apache to bind to this port, SELinux's Mandatory Access Control (MAC) layer adds an additional restriction and denies the operation. This demonstrates the core principle of SELinux: it enforces security policies on top of traditional permissions.

The AVC log also confirms that httpd_t attempted to name_bind to a tcp_socket labeled unreserved_port_t, and SELinux denied the request.

Deliverable 1.3: Once you have updated the SElinux port policy and the firewalld configuration, take a screenshot of the output of the previous semanage command, and a capture of the successful connection to httpd on port 10xxx.

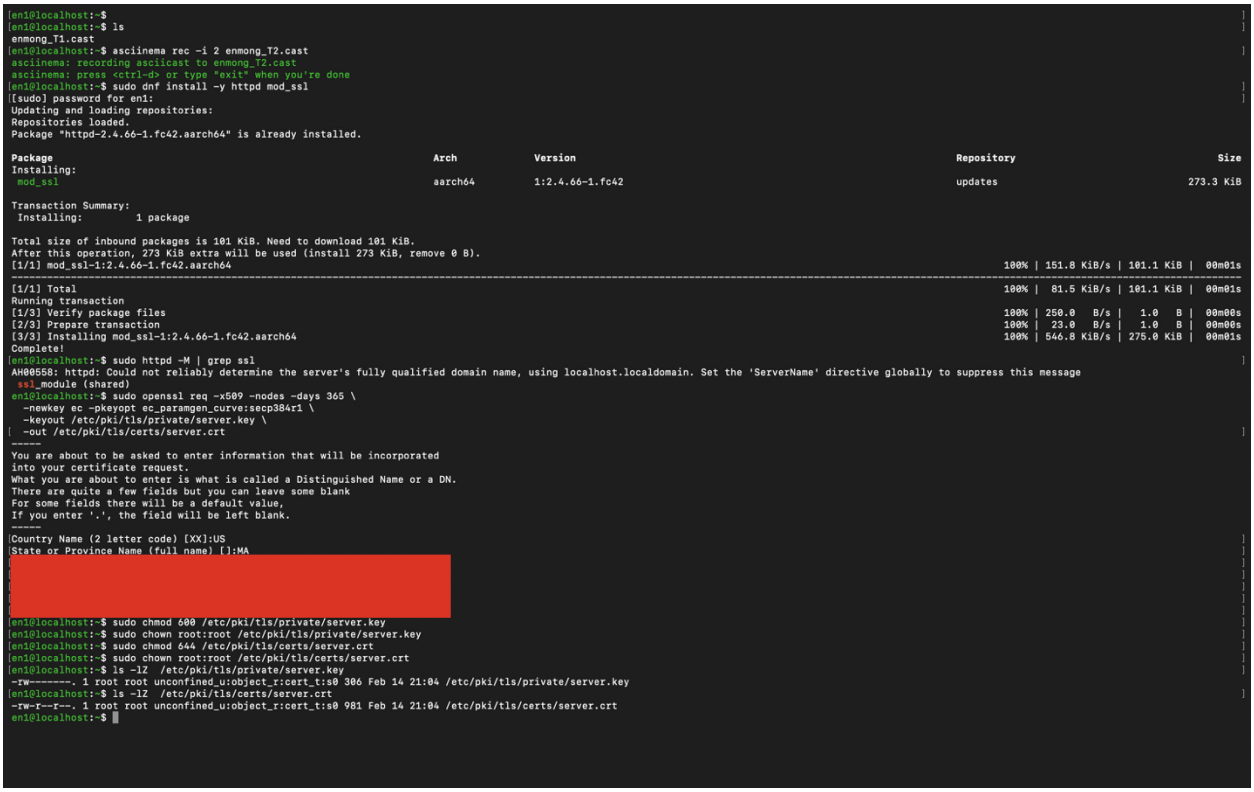


- After adding po[redacted]ttp_port_t type using *semanage port -a*, the SELinux policy now recognizes this port as a valid HTTP port. The *semanage port -l* output confirms[redacted] appears alongside the default HTTP ports (80, 443, etc.). The firewall was updated using *firewall-cmd --add-por[redacted]ermanent* to allow incoming TCP traffic on the new port. The successful curl response demonstrates that Apache is now binding to the new port with full SELinux approv[redacted]d by the absence of any AVC denials in the audit log.

Part 2.

Task 2: Enabling HTTPS and certificate files SElinux context

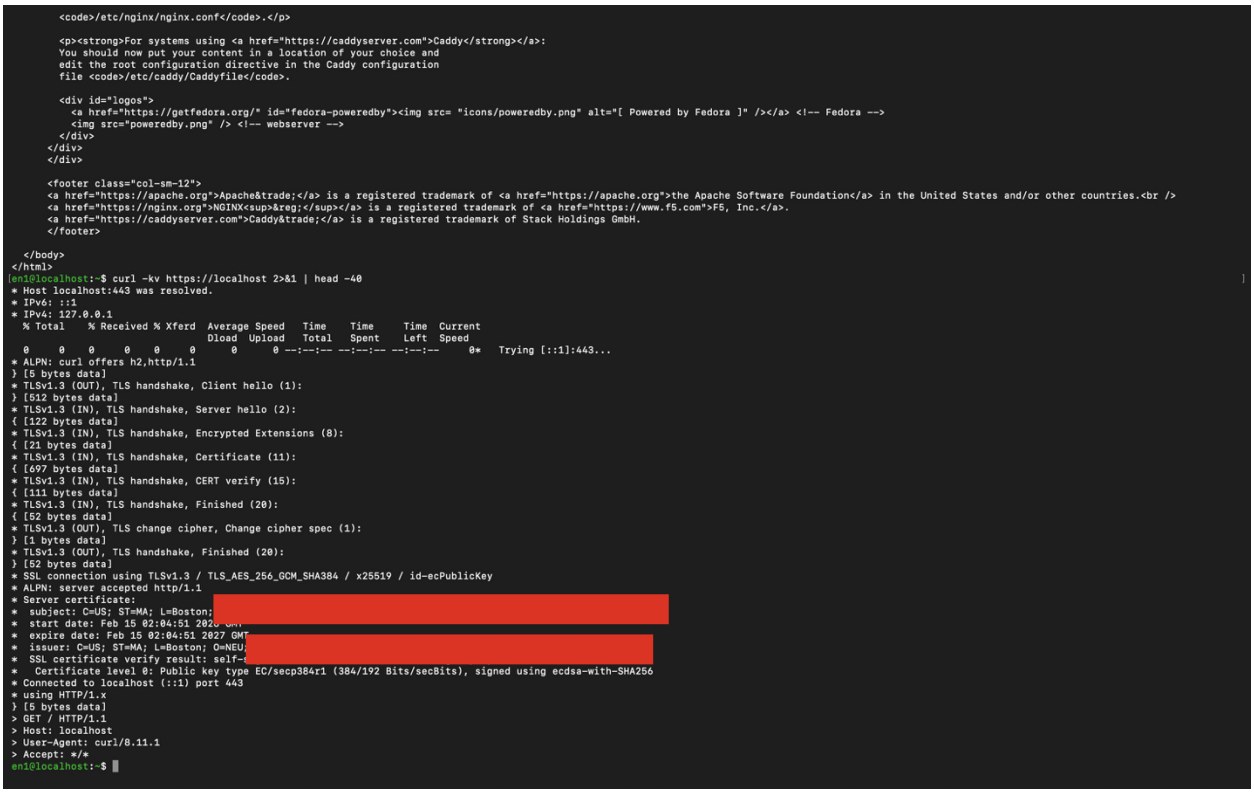
Deliverable 2.1: Include a screenshot of the locations, permissions and context of the certificate and private key you generated, and provide proper reasoning to support the SElinux context that you have established for them

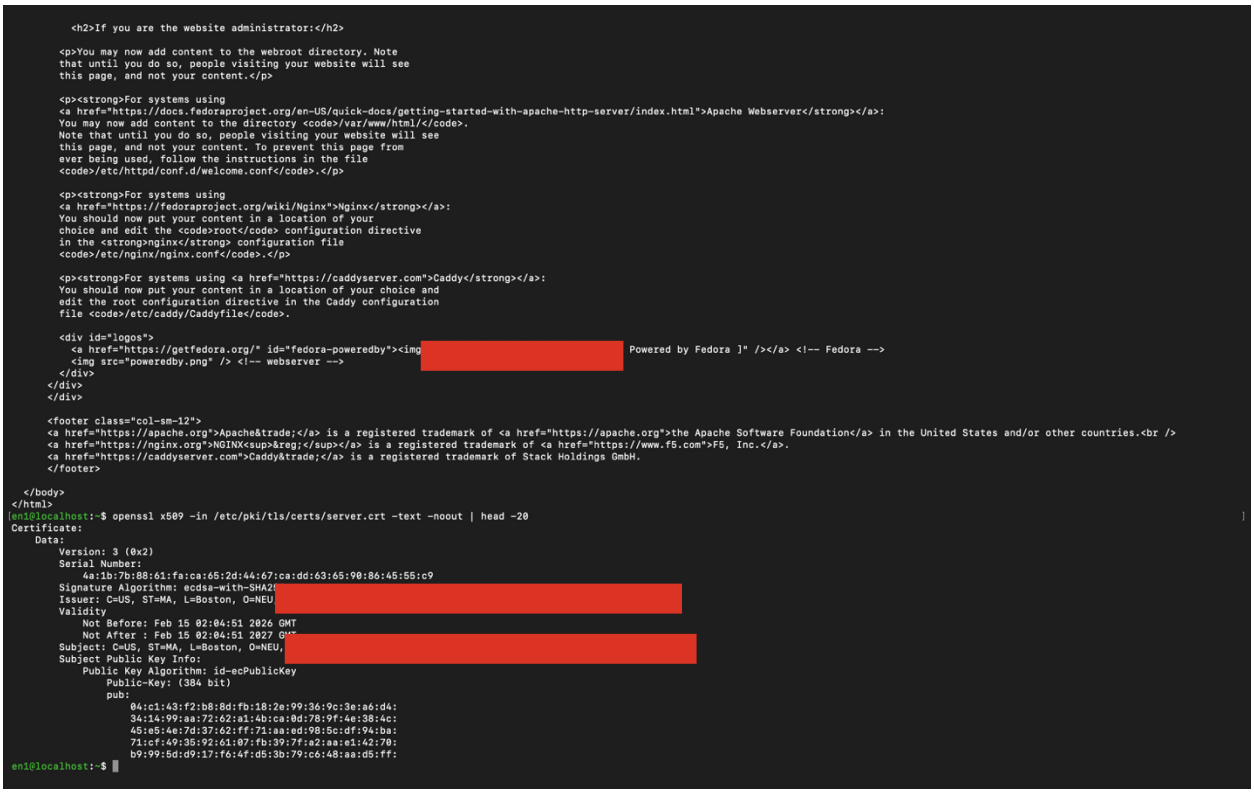


- The certificate and private key are stored in /etc/pki/tls/certs/ and /etc/pki/tls/private/, which follow the standard locations defined by Red Hat's filesystem policy. These directories are pre-labeled in the SELinux policy, so files created directly within them automatically inherit the cert_t type without requiring manual context changes. Both files

have the SELinux context cert_t, the type that the httpd_t domain is permitted to read when loading TLS materials. The private key is set to restrictive 600 permissions (read/write for the owner only) to protect sensitive key material. In contrast, the certificate uses 644 permissions since it contains public information that is shared during the TLS handshake.

Deliverable 2.2: Take a screenshot of a successful connection to https on port 443, and also include the public key certificate provided by your web server and highlight your certificate information.





- The HTTPS connection was successfully established using TLS 1.3, with ECDSA P-384 handling both the key exchange and the digital signature. The connection negotiated TLS_AES_256_GCM_SHA384 for symmetric encryption with x25519 for ephemeral key exchange, providing both strong confidentiality and perfect forward secrecy. The certificate's Subject field shows my personal details [redacted] confirming that the server is presenting my custom self-signed certificate. The certificate was placed in the standard /etc/pki/tls/ directories and assigned the correct cert_t SELinux context, which allows the httpd_t domain to access the necessary TLS files. Since the certificate is self-signed and not included in the system's trusted CA store, the -k flag was needed to bypass certificate verification during the connection test.

Part 3.

Task 3: SElinux context for web server CONTENT.

Deliverable 3.1: Take a screenshot of the output you obtain when you connect to your web server.

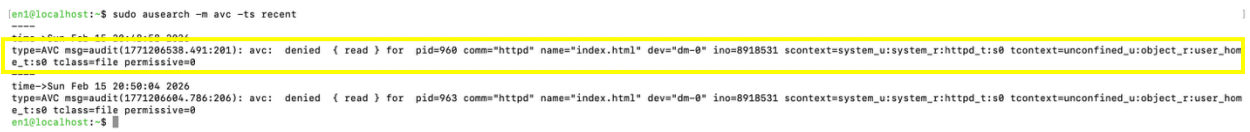
Both curl commands to HTTP on [redacted] HTTPS on port 443 return a 403 Forbidden error. Apache is running and reachable, but it cannot serve the index.html file due to an SELinux context mismatch caused by using mv instead of cp to place the file.

Deliverable 3.2: SElinux is preventing you from accessing the content of your recently moved index.html file, provide reasoning that justifies this behavior.

- The 403 Forbidden error occurs because SELinux is preventing Apache (httpd_t) from reading the index.html file, even though the file is now located in /var/www/html/, the correct directory for web content. The issue arises from the file's SELinux context. Since the file was originally created in /home/en1/, it was automatically assigned the user_home_t type, the default label for files in a user's home directory. When it was moved with the mv command, this context traveled with it because mv performs a filesystem-level rename rather than creating a new file, so SELinux preserves the original label. Apache, running under the httpd_t confined domain, is allowed to read files labeled httpd_sys_content_t (and related types like httpd_sys_content_rw_t), but it has no policy permission to read files labeled user_home_t. SELinux therefore treats the file as a home-directory file that Apache should not access, regardless of its physical location. This highlights the difference between DAC and MAC in SELinux, although the file's 644 permissions would allow any process to read it under traditional DAC rules, SELinux's MAC layer denies access because httpd_t is not authorized to read a file of type user_home_t. Both DAC and MAC must permit access for it to succeed. If the file had

instead been copied with cp, it would have been created a new in /var/www/html/ and inherited the correct httpd_sys_content_t context, allowing the page to load successfully.

Deliverable 3.3: To show SElinux is preventing you from accessing the content of your recently moved index.html file, provide a screenshot of the corresponding AVC logs.



- The AVC denial was retrieved using *sudo ausearch -m avc -ts recent*, which filters the audit log for recent SELinux denial events. The AVC denial log shows that SELinux is blocking Apache's attempt to read the index.html file. The source context (scontext=system_u:system_r:httpd_t:s0) confirms that the request came from Apache running in the confined httpd_t domain, while the target context (tcontext=unconfined_u:object_r:user_home_t:s0) reveals the real issue, the file still has the user_home_t label it inherited from when it was originally created in the user's home directory. Because the SELinux targeted policy does not allow httpd_t processes to read files labeled user_home_t, the { read } permission is denied. The permissive=0 field shows the system is in enforcing mode, so SELinux actively blocked the access instead of only logging it. This demonstrates how SELinux's mandatory access control (MAC) adds a separate layer of protection beyond traditional Unix permissions: even though the file has world-readable 644 permissions under DAC, SELinux still denies access because the file's type does not match what Apache is allowed to read. The proper fix is to restore the correct context using *restorecon*, which relabels the file to httpd_sys_content_t according to the default policy for the /var/www/html/ directory.

Deliverable 3.4: Describe what changes need to be done, so your web server can provide access to the index.html file.

- To resolve the 403 Forbidden error, the SELinux context on the index.html file must be updated from user_home_t to httpd_sys_content_t, which is the correct type for web content served from /var/www/html/. Running the command *sudo restorecon -v /var/www/html/index.html* automatically applies the appropriate context based on SELinux's default file-context policy for that directory, and the -v flag simply shows the transition from the incorrect label to the correct one. This approach is better than using *chcon,* which manually assigns a label but does not persist if SELinux policies are reapplied later. Using *restorecon* ensures the context remains consistent and automatically correct in the future. Assigning the file the httpd_sys_content_t type gives Apache's httpd_t domain the necessary read-only permissions to serve the file, following the principle of least

privilege while preventing unauthorized modification. Once the file is relabeled, both HTTP on po[redacted]TTPS on port 443 successfully serve the page because the httpd_sys_content_t type grants httpd_t read and open permissions on the file, but not write or execute, ensuring Apache can serve the page without being able to modify it.

Deliverable 3.5: Take a screenshot of the output you obtain connecting to the server, once you have fixed the problem.

After running *restorecon -v /var/www/html/index.html* to relabel the file from user_home_t to httpd_sys_content_t, both HTTP o[redacted]d HTTPS on port 443 now successfully return the custom index.html content. This confirms that the SELinux context was the only barrier preventing Apache from serving the file.

Part4

Task 4: Create a sudo transition for a cy5130_u to become webadm_r

```
[en1@localhost:~$ sudo semanage user -a -R "staff_r Webadm_r system_r" cy5130_u
[sudo] password for en1:
ValueError: Webadm_r must be an SELinux role:
Valid roles: webadm_r, xguest_r, unconfined_r, logadm_r, nx_server_r, secadm_r, dbadm_r, guest_r, sysadm_r, user_r, auditadm_r, system_r, staff_r
[en1@localhost:~$ sudo semanage user -a -R "staff_r webadm_r system_r" cy5130_u
[en1@localhost:~$ sudo semanager user -l | grep cy5130
sudo: semanager: command not found
[en1@localhost:~$ sudo semanage user -l | grep cy5130
cy5130_u        user       s0        s0                              system_r staff_r webadm_r
[en1@localhost:~$ sudo useradd -Z cy5130_u en2
[en1@localhost:~$ sudo passwd en2
New password:
Retype new password:
passwd: password updated successfully
[en1@localhost:~$ sudo semanage login -l | grep en2
en2                cy5130_u           s0                  *
[en1@localhost:~$ sudo restorecon -RvF /home/en2
[en1@localhost:~$ [redacted]
The authenticity[redacted]e established.
ED25519 key fingerprint is SHA256:CDNqoE1705TfmtAz8mmVYD68po03wKfAgU4bQ5cZuts.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
[redacted](ED25519) to the list of known hosts.
[en2@localhost:~$ ls -la
total 12
drwx------. 2 en2  en2   62 Feb 19 09:55 .
drwxr-xr-x. 4 root root  28 Feb 19 09:55 ..
-rw-r--r--. 1 en2  en2   18 Nov  7  2024 .bash_logout
-rw-r--r--. 1 en2  en2  144 Nov  7  2024 .bash_profile
-rw-r--r--. 1 en2  en2  522 Nov  7  2024 .bashrc
[en2@localhost:~$ id -z
id: option --zero not permitted in default format
[en2@localhost:~$ id -Z
cy5130_u:staff_r:staff_t:s0


[en1@localhost:~$ ls -lZ /etc/sudoers.d/en2-webadm
ls: cannot access '/etc/sudoers.d/en2-webadm': Permission denied
[en1@localhost:~$ sudo ls -lZ /etc/sudoers.d/en2-webadm
-rw-r-----. 1 root root unconfined_u:object_r:etc_t:s0 50 Feb 19 10:01 /etc/sudoers.d/en2-webadm
[en1@localhost:~$ sudo chmod 440 /etc/sudoers.d/en2-webadm
[en1@localhost:~$ sudo ls -lZ /etc/sudoers.d/en2-webadm
-r--r-----. 1 root root unconfined_u:object_r:etc_t:s0 50 Feb 19 10:01 /etc/sudoers.d/en2-webadm
 en1@localhost:~$
```

The cy5130_u SELinux user was created using the command *semanage user -a -R "staff_r webadm_r system_r" cy5130_u*, which assigns three roles: staff_r as the default role, webadm_r for web administration tasks, and system_r to allow sudo role transitions. Next, the Linux user en2 was created with *useradd -Z cy5130_u en2*, linking it to the cy5130_u SELinux user. Running *semanage login -l* confirmed that the mapping was successfully applied. After relabeling the home directory using *restorecon -RvF /home/en2*, I connected as en2 via SSH, and *id -Z* showed the expected context: cy5130_u:staff_r:staff_t:s0. A sudoers drop-in file was then created at /etc/sudoers.d/en2-webadm with the rule: en2 ALL=(ALL) TYPE=webadm_t ROLE=webadm_r /bin/sh. Its permissions were corrected from 640 to the required 440 to ensure it remains read-only and secure.

Deliverable 4.1: Take two screenshots of the output you obtain when you run id -Z , one before the sudo transition is called and one after the sudo is called (it needs to be sequential commands)

The first *id -Z* output shows en2 running under the default staff_r role with the staff_t type, which provides a confined but general-purpose environment for everyday tasks. After *running sudo /bin/sh*, the SELinux context transitions to webadm_r:webadm_t, as configured in the sudoers drop-in file located at /etc/sudoers.d/en2-webadm. The SELinux user (cy5130_u) does not change because SELinux keeps the user identity consistent across role transitions, only the role and type are updated. This behavior illustrates SELinux's Role-Based Access Control (RBAC), rather than giving en2 full, unconfined root privileges, the sudo transition places the shell inside the webadm_t domain, restricting it to web-administration tasks only. This follows the principle of least privilege that en2 gains the specific capabilities needed for web server management without obtaining unrestricted system access.

Part5

Task 5: Create a content page for the user

Deliverable 5.1: Take a screenshot of the /etc/httpd/conf.d/userdir.conf file with the appropriate settings.

```
[en1@localhost:~$ sudo cat /etc/httpd/conf.d/userdir.conf
#
# UserDir: The name of the directory that is appended onto a user's home
# directory if a ~user request is received.
#
# The path to the end user account 'public_html' directory must be
# accessible to the webserver userid.  This usually means that ~userid
# must have permissions of 711, ~userid/public_html must have permissions
# of 755, and documents contained therein must be world-readable.
# Otherwise, the client will only receive a "403 Forbidden" message.
#
<IfModule mod_userdir.c>
    #
    # UserDir is disabled by default since it can confirm the presence
    # of a username on the system (depending on home directory
    # permissions).
    #
    #UserDir disabled

    #
    # To enable requests to /~user/ to serve the user's public_html
    # directory, remove the "UserDir disabled" line above, and uncomment
    # the following line instead:
    #
    UserDir public_html
</IfModule>

#
# Control access to UserDir directories.  The following is an example
# for a site where these directories are restricted to read-only.
#
<Directory "/home/*/public_html">
    AllowOverride FileInfo AuthConfig Limit Indexes
    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
    Require method GET POST OPTIONS
</Directory>

en1@localhost:~$
```

The userdir.conf file was updated by changing UserDir disabled to UserDir public_html, which allows Apache to serve content from each user's public_html directory when URLs in the format /~username/ are accessed. The corresponding <Directory> block was also uncommented and configured with Require all granted to permit access. Additionally, the options Indexes and FollowSymLinks were enabled so Apache can display directory listings and follow symbolic links as needed.

Deliverable 5.2: Take a screenshot of the edited context and permissions of the user's public_html directory and the boolean command used.

```
[en1@localhost:~$ sudo semanage fcontext -a -t httpd_user_content_t "/home/en2/public_html(/.*)?"
[en1@localhost:~$ sudo restorecon -Rv /home/en2/public_html
 /home/en2/public_html not reset as customized by admin to cy5130_u:object_r:httpd_user_content_t:s0
 /home/en2/public_html/index.html not reset as customized by admin to cy5130_u:object_r:httpd_user_content_t:s0
[en1@localhost:~$ ls -lZ /home/en2/public_html/
 total 4
 -rw-r--r--. 1 en2 en2 cy5130_u:object_r:httpd_user_content_t:s0 51 Feb 19 10:42 index.html
[en1@localhost:~$ sudo setsebool -P httpd_enable_homedirs on
[en1@localhost:~$ getsebool httpd_enable_homedirs
 httpd_enable_homedirs --> on


[en1@localhost:~$ ls -lZ /home/en2/public_html/
 total 4
 -rw-r--r--. 1 en2 en2 cy5130_u:object_r:httpd_user_content_t:s0 51 Feb 19 10:42 index.html
[en1@localhost:~$ ls -lZ /home/en2/public_html/index.html
 -rw-r--r--. 1 en2 en2 cy5130_u:object_r:httpd_user_content_t:s0 51 Feb 19 10:42 /home/en2/public_html/index.html
[en1@localhost:~$ getsebool httpd_enable_homedirs
 httpd_enable_homedirs --> on
 en1@localhost:~$
```

The public_html directory and its contents were labeled with the httpd_user_content_t type using the command *semanage fcontext -a -t httpd_user_content_t "/home/en2/public_html(/.*)?" followed* by a recursive *restorecon -Rv*. This type is specifically intended for user-published web content and is different from httpd_sys_content_t (which applies to system-managed web files in /var/www/html) and from the default user_home_t label used for regular home-directory files. To allow Apache to read content stored in users' home directories, the SELinux boolean httpd_enable_homedirs was enabled using *setsebool -P httpd_enable_homedirs* on . Both the correct file context and the boolean are necessary, the file context tells SELinux what kind of content the files represent, while the boolean determines whether Apache is permitted to access home-directory files at all. Directory permissions are set to 755 and file permissions to 644, giving Apache the DAC-level read access it needs. The home directory itself uses 711, which allows traversal without exposing other personal files.

Deliverable 5.3: Take a screenshot of the output of curl localhost and localhost/~<username>/ (apache's page and user's page)

```
[en1@localhost:~$ curl htt
 <html>
 <header><title>SELINUX Project</title></header>
 <body>
 I am ..enmong... and this is my SELINUX Project on date ..Feb 15 2026..
 </body>
 </html>
[en1@localhost:~$ curl http
 <html>
 <body>
 Hello from user en2.
 </body>
 </html>
[en1@localhost:~$ curl -k https://localhost/~en2/
 <html>
 <body>
 Hello from user en2.
 </body>
 </html>
```

The first curl request to lo returns the default Apache page served from /var/www/html/index.html, which is labeled with the httpd_sys_content_t SELinux context. The second curl request to localho correctly loads the user's page from /home/en2/public_html/index.html, which has the httpd_user_content_t context.

Both pages load successfully because SELinux allows the httpd_t domain to read httpd_sys_content_t files by default, and it permits access to httpd_user_content_t files when the httpd_enable_homedirs boolean is enabled. This setup illustrates SELinux's fine-grained access control, different types of content receive different labels, and administrators can toggle specific capabilities using SELinux booleans without needing to write custom policies.