

# Project 3: Access Control

## Part1

### Task 0: Initial setup

```
Last login: Sat Feb 14 20:55:27 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
[redacted]:~ [redacted]'s password:

1 device has a firmware upgrade available.
Run `fwupdmgr get-upgrades` for more information.

Web console: https://[redacted] or https://[redacted]/

Last login: Sun Feb 15 20:27:20 2026
[en1@localhost:~$ sudo whoami
[[sudo] password for en1:
root
[en1@localhost:~$ 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:      35
en1@localhost:~$ ]]
```

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.

```
[root@localhost ~]# sudoausearch -m avc -ts recent
...
time=>Sat Feb 14 16:12:42 2026
type=AVC msg=audit(1771183562.980:422): avc: denied { name_bind } for pid=1832 comm="httpd" src=10429 scontext=system_u:system_r:httpd_t:s0 tcontext=system_u:object_r:unreserved_port_t:s0 tclass=tcp_socket permissive=0
...
time=>Sat Feb 14 16:12:42 2026
type=AVC msg=audit(1771183562.980:423): avc: denied { name_bind } for pid=1832 comm="httpd" src=10429 scontext=system_u:system_r:httpd_t:s0 tcontext=system_u:object_r:unreserved_port_t:s0 tclass=tcp_socket permissive=0
[root@localhost ~]# sudo systemctl status httpd
* httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
   Drop-In: /usr/lib/systemd/system/httpd.service
             └─_httpd.conf
     Active: failed (Result: exit-code) since Sat 2026-02-14 16:12:42 EST; 2min 21s ago
       Duration: 1min 11.292s
     Invocation: 5a4d65faced448d9df8e83a/c3b84e2
       Process: 1832 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=1/FAILURE)
      Main PID: 1832 (code=exited, status=1/FAILURE)
        Status: "Reading configuration..."
       Memory: 45ms
          CPU: 45ms

Feb 14 16:12:42 localhost.localdomain systemd[1]: Starting httpd service - The Apache HTTP Server...
Feb 14 16:12:42 localhost.localdomain httpd[1832]: httpd.service: Referenced but unset environment variable evaluates to an empty string: OPTIONS
Feb 14 16:12:42 localhost.localdomain httpd[1832]: AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using localhost.localdomain. Set the 'ServerName' directive globally to suppress this message
Feb 14 16:12:42 localhost.localdomain httpd[1832]: (13)Permission denied: AH00872: make_sock: could not bind to address [::]:10429
Feb 14 16:12:42 localhost.localdomain httpd[1832]: no listening sockets available, shutting down
Feb 14 16:12:42 localhost.localdomain httpd[1832]: AH00015: Unable to open logs
Feb 14 16:12:42 localhost.localdomain httpd[1832]: AH00016: Fatal error, failing start
Feb 14 16:12:42 localhost.localdomain httpd[1832]: AH00016: Failed with result 'exit-code'.
Feb 14 16:12:42 localhost.localdomain systemd[1]: Failed to start httpd.service - The Apache HTTP Server.

[root@localhost ~]# sudo systemctl restart httpd
* httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
   Drop-In: /usr/lib/systemd/system/httpd.service
             └─_httpd.conf
     Active: failed (Result: exit-code) since Sat 2026-02-14 16:15:26 EST; 4s ago
       Duration: 1min 11.292s
     Invocation: 55847de4a294c5b0d4292de405d07ef
       Docs: manhttpd.service(8)
      Process: 1876 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=5/FAILURE)
      Main PID: 1876 (code=exited, status=5/FAILURE)
        Status: "Reading configuration..."
       Memory: 3M
          CPU: 42ms

Feb 14 16:15:26 localhost.localdomain systemd[1]: Starting httpd service - The Apache HTTP Server...
Feb 14 16:15:26 localhost.localdomain httpd[1876]: httpd.service: Referenced but unset environment variable evaluates to an empty string: OPTIONS
Feb 14 16:15:26 localhost.localdomain httpd[1876]: AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using localhost.localdomain. Set the 'ServerName' directive globally to suppress this message
Feb 14 16:15:26 localhost.localdomain httpd[1876]: (13)Permission denied: AH00872: make_sock: could not bind to address [::]:10429
Feb 14 16:15:26 localhost.localdomain httpd[1876]: AH00015: Unable to open logs
Feb 14 16:15:26 localhost.localdomain httpd[1876]: AH00016: Fatal error, failing start
Feb 14 16:15:26 localhost.localdomain httpd[1876]: AH00016: Failed with result 'exit-code'.
Feb 14 16:15:26 localhost.localdomain systemd[1]: Failed to start httpd.service - The Apache HTTP Server.

[root@localhost ~]# sudoausearch -m avc -ts recent
...
time=>Sat Feb 14 16:12:42 2026
type=AVC msg=audit(1771183562.980:422): avc: denied { name_bind } for pid=1832 comm="httpd" src=10429 scontext=system_u:system_r:httpd_t:s0 tcontext=system_u:object_r:unreserved_port_t:s0 tclass=tcp_socket permissive=0
...
time=>Sat Feb 14 16:12:42 2026
type=AVC msg=audit(1771183562.980:423): avc: denied { name_bind } for pid=1832 comm="httpd" src=10429 scontext=system_u:system_r:httpd_t:s0 tcontext=system_u:object_r:unreserved_port_t:s0 tclass=tcp_socket permissive=0
...
time=>Sat Feb 14 16:15:26 2026
type=AVC msg=audit(1771183726.234:455): avc: denied { name_bind } for pid=1876 comm="httpd" src=10429 scontext=system_u:system_r:httpd_t:s0 tcontext=system_u:object_r:unreserved_port_t:s0 tclass=tcp_socket permissive=0
...
time=>Sat Feb 14 16:15:26 2026
type=AVC msg=audit(1771183726.234:456): avc: denied { name_bind } for pid=1876 comm="httpd" src=10429 scontext=system_u:system_r:httpd_t:s0 tcontext=system_u:object_r:unreserved_port_t:s0 tclass=tcp_socket permissive=0
[root@localhost ~]#
```

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, port [REDACTED] is labeled 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.

```
[root@localhost ~]# sudo semanage port -a | grep http_port_t
http_port_t          tcp      81, 443, 488, 8080, 8089, 8443, 9000
http_port_t          tcp      9898
[root@localhost ~]# curl http://localhost
[{"text": "HTTP Server for the FEDORA Project"}, {"text": "HTTP Server for the FEDORA Project"}]
```



```
[root@localhost ~]# curl https://localhost
[{"text": "HTTP Server for the FEDORA Project"}, {"text": "HTTP Server for the FEDORA Project"}]
```



```
[en1@localhost:~]$ sudo semanage port -l
http_port_t          top      80, 443, 488, 8080, 8089, 8443, 9080
http_port_t          top      5050
[en1@localhost:~]$ curl http://localhost:10429
<!DOCTYPE html>
<html>
<head>
<meta charset='utf-8'>
<meta name='viewport' content='width=device-width, initial-scale=1'>
<title>Apache VHost Test</title>
</head>
```

- After adding port [REDACTED] `http_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-port [REDACTED] permanent` 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 approval [REDACTED] 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

```
[en1@localhost:~]$ ls
[en1@localhost:~]$ ls
enmonng_T2.cast
[en1@localhost:~]$ asciinema rec -i 2 enmonng_T2.cast
ascinema: recording asciicast to enmonng_T2.cast
ascinema: press <ctrl-d> or type "exit" when you're done
[en1@localhost:~]$ sudo dnf install -y httpd mod_ssl
[sudo] password for en1:
Updating and loading repositories:
Repositories:
Package "httpd-2.4.66-1.fc42.aarch64" is already installed.

Transaction Summary:
Installing:
  mod_ssl                                     Arch       Version           Repository      Size
[mod_ssl]                                         aarch64   1:2.4.66-1.fc42   updates        273.3 KiB

Transaction Summary:
Installing: 1 package

Total size of inbound packages is 101 KiB. Need to download 101 KiB.
After this operation, 273 KiB extra will be used (install 273 KiB, remove 0 B).
[1/1] mod_ssl-1:2.4.66-1.fc42.aarch64
[1/1] Total
Running transaction
[1/1] Verify package files
[2/2] Prepare transaction
[3/3] Installing mod_ssl-1:2.4.66-1.fc42.aarch64
Complete!
[en1@localhost:~]$ sudo httpd -M | grep ssl
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using localhost.localdomain. Set the 'ServerName' directive globally to suppress this message
[en1@localhost:~]$ sudo openssl req -x509 -nodes -days 365 \
-newkey ec -pkeyopt ec_paramgen_curve:secp384r1 \
-keyout /etc/pki/tls/private/server.key \
-out /etc/pki/tls/certs/server.crt
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
[Country Name (2 letter code) [XX]:US
[State or Province Name (full name) [ ]MA
[en1@localhost:~]$ sudo chmod 600 /etc/pki/tls/private/server.key
[en1@localhost:~]$ sudo chown root:root /etc/pki/tls/private/server.key
[en1@localhost:~]$ sudo chmod 644 /etc/pki/tls/certs/server.crt
[en1@localhost:~]$ sudo chown root:root /etc/pki/tls/certs/server.crt
[en1@localhost:~]$ ls -lZ /etc/pki/tls/private/server.key
-rw-----. 1 root root unconfined_u:object_r:cert_t:s0 306 Feb 14 21:04 /etc/pki/tls/private/server.key
[en1@localhost:~]$ ls -lZ /etc/pki/tls/certs/server.crt
-rw-r--r--. 1 root root unconfined_u:object_r:cert_t:s0 981 Feb 14 21:04 /etc/pki/tls/certs/server.crt
[en1@localhost:~]$
```

- 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.

```
<code>/etc/nginx/nginx.conf</code></p>
<p><strong>For systems using <a href="https://caddyserver.com">Caddy</strong></a>:<br>
You should now put your content in a location of your choice and
edit the root configuration directive in the Caddy configuration
file <a href="etc/caddy/Caddyfile">/code</a>.</p>
<div id="logos">
<a href="https://getfedora.org/" id="fedora-poweredby"><img src= "icons/poweredbypng" alt="Powered by Fedora" /></a> <!-- Fedora -->
 <!-- webserver -->
</div>
</div>

<footer class="col-sm-12">
<a href="https://apache.org">Apache&trade;</a> is a registered trademark of <a href="https://apache.org">the Apache Software Foundation</a> in the United States and/or other countries.<br />
<a href="https://nginx.org">NGINX<sup>x</sup></a> is a registered trademark of <a href="https://www.f5.com">F5, Inc.</a>.
<a href="https://caddyserver.com">Caddy</a> is a registered trademark of Stack Holdings GmbH.
</footer>

</body>
</html>
[en1@localhost: ~]$ curl -kv https://localhost 2>&1 | head -40
* Host localhost:443 was resolved.
* IPv6 ::1
* IPv4: 127.0.0.1
* Total:  % Received % Xferd Average Speed Time Time Time Current
          % Total   Dload Upload Total Spent Left Speed
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0* Trying [::1]:443...
* ALPN: curl offers h2,http/1.1
} [5 bytes data]
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
} [51 bytes data]
* TLSv1.3 (OUT), TLS handshake, Server hello (2):
{ [122 bytes data]
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
{ [21 bytes data]
* TLSv1.3 (IN), TLS handshake, Certificate (11):
{ [64 bytes data]
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
{ [111 bytes data]
* TLSv1.3 (IN), TLS handshake, Finished (20):
{ [62 bytes data]
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
} [1 bytes data]
* TLSv1.3 (OUT), TLS handshake, Finished (20):
{ [51 bytes data]
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384 / x25519 / id-ecPublicKey
* ALPN: server accepted http/1.1
* Server certificate:
* subject: C=US; ST=MA; L=Boston;
* start date: Feb 15 02:04:51 2021 -04:00
* expire date: Feb 15 02:04:51 2027 GMT
* issuer: C=US; ST=MA; L=Boston; O=NEU
* SSL certificate verify result: self-
* Certificate level 0: Public key type EC/sep384r1 (384/192 Bits/secBits), signed using ecdsa-with-SHA256
* Connected to localhost (::1) port 443
* using HTTP/1.x
} [5 bytes data]
> GET / HTTP/1.1
> Host: localhost
> User-Agent: curl/8.11.1
> Accept: */*
[en1@localhost: ~]$
```

```

<h2>If you are the website administrator:</h2>
<p>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.</p>
<p><strong>For systems using
<a href="https://centosproject.org/en-US/guides-docs/getting-started-with-apache-http-server/index.html">Apache Webserver</strong></a>:
You may now add content to the directory <code>/var/www/html</code>.
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
<code>/etc/httpd/conf.d/welcome.conf</code>.</p>
<p><strong>For systems using
<a href="https://fedoraproject.org/wiki/Nginx">Nginx</strong></a>:
You should now put your content in a location of your
choice and edit the <code>root</code> configuration directive
in the <strong>nginx</strong> configuration file
<code>/etc/nginx/nginx.conf</code>.</p>
<p><strong>For systems using <a href="https://caddyserver.com">Caddy</strong></a>:
You should now put your content in a location of your choice and
edit the root configuration directive in the Caddy configuration
file <code>/etc/caddy/Caddyfile</code>.</p>
<div id="logos">
<a href="https://getfedora.org/" id="fedora-poweredby"> <!-- webserver -->
 <!-- Fedora -->
</div>
</div>

<footer class="col-um-12">
<a href="https://apache.org">Apache&trade;</a> is a registered trademark of <a href="https://apache.org">the Apache Software Foundation</a> in the United States and/or other countries.<br />
<a href="https://nginx.org">NOTNX&ltsup>®</sup></a> is a registered trademark of <a href="https://www.f5.com">F5, Inc.</a>.
<a href="https://caddyserver.com">Caddy</a> is a registered trademark of Stack Holdings GmbH.
</footer>
</body>
</html>
[en1@localhost ~]$ openssl x509 -in /etc/pki/tls/certs/server.crt -text -noout | head -20
Certificate:
Data:
Version: 3 (0x2)
Serial Number:
        4e:11:86:61:f:a:c:65:2:d:4:67:c:a:dd:63:65:9:86:45:55:c9
Signature Algorithm: ecdsa-with-SHA256
Issuer: C=US, S=MA, L=Boston, O=MPU
Validity
    Not Before: Feb 15 02:04:51 2026 GMT
    Not After : Feb 15 02:04:51 2027 GMT
Subject: C=US, S=MA, L=Boston, O=NEU,
Subject Public Key Info:
    Public Key Algorithm: id-ecPublicKey
        Public-Key: (384 bit)
            pub:
                04:c1:43:f2:88:8d:fb:18:2e:99:36:9:c3:e:a6:d4:
                34:14:99:aa:72:62:a1:4:b:a:9d:78:9f:4:e:38:4c:
                45:e5:4e:7:d:37:62:ff:7:1:a:aa:ed:98:b:c:d:f194:ba:
                7:1:c:f4:9:30:92:61:0:7:10:39:77:a:a:a:e1:42:70:
                b:9:99:5d:d9:17:76:4:f:id:5b:79:c6:48:a:a:ds:rf:
[en1@localhost ~]$ 
```

- 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.

```
[en1@localhost ~]$ nano index.html
[en1@localhost ~]$ ls
emong_T1.cast emmong_T2.cast emmong_T3.cast index.html
[en1@localhost ~]$ pwd
/home/en1
[en1@localhost ~]$ ls -lZ ~/index.html
-rw-r--r--. 1 en1 en1 unconfined_u:object_r:user_home_t:s0 150 Feb 15 20:46 /home/en1/index.html
[en1@localhost ~]$ sudo mv ~/index.html /var/www/html/index.html
[sudo] password: 
[en1@localhost ~]$ ls -lZ /var/www/html/index.html
-rw-r--r--. 1 en1 en1 unconfined_u:object_r:user_home_t:s0 150 Feb 15 20:46 /var/www/html/index.html
[en1@localhost ~]$ ls -lZ /var/www/html/
total 4
-rw-r--r--. 1 en1 en1 unconfined_u:object_r:user_home_t:s0 150 Feb 15 20:46 index.html
[en1@localhost ~]$ curl https://localhost/index.html --cacert "/http://www.w3.org/TR/html4/strict.dtd"
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
</body></html>
[en1@localhost ~]$ curl https://localhost
curl: (60) SSL certificate problem: self-signed certificate
More details here: https://curl.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the webpage mentioned above.
[en1@localhost ~]$ curl -k https://localhost
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
</body></html>
[en1@localhost ~]$
```

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.

```
[root@localhost ~]# sudo ausearch -m avc -ts recent
time=Sun Feb 15 20:50:04 2026
type=AVC msg=audit(1711206538.491:201): avc: denied { read } for pid=960 comm="httpd" name="index.html" dev="dm-0" ino=8918531 scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:user_home_t:s0 tclass=file permissive=0
time=Sun Feb 15 20:50:04 2026
type=AVC msg=audit(1711206604.786:206): avc: denied { read } for pid=963 comm="httpd" name="index.html" dev="dm-0" ino=8918531 scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:user_home_t:s0 tclass=file permissive=0
[root@localhost ~]#
```

- 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 port [REDACTED] and HTTPS 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.

```
[en1@localhost:~$ sudo restorecon -v /var/www/html/index.html
Relabeled /var/www/html/index.html from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_content_t:s0
[en1@localhost:~$ ls -lZ /var/www/html/index.html
-rw-r--r--. 1 en1 en1 unconfined_u:object_r:httpd_sys_content_t:s0 150 Feb 15 20:46 /var/www/html/index.html
[en1@localhost:~$ curl -k https://localhost
<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 -k https://localhost
<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:~$ ]
```

After running `restorecon -v /var/www/html/index.html` to relabel the file from `user_home_t` to `httpd_sys_content_t`, both HTTP on [REDACTED] and 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, guest_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 semanage user -l | grep cy5130_u
sudo: semanage: command not found
[en1@localhost:~$ sudo semanage user -l | grep cy5130
cy5130_u          user_r      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:~$ 
The authenticity of host [REDACTED] is established.
ED25519 key fingerprint is SHA256:CDNqoE1705ffmtA28mmVYD68po03WkFaGU4b05cZuts.
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
drwxr-xr-x. 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)

```
[en1@localhost:-$ sudo visudo -cf /etc/sudoers.d/en2-webadm  
/etc/sudoers.d/en2-webadm: parsed OK  
[en1@localhost:-$ sudo cat /etc/sudoers.d/en2-webadm  
en2 ALL=(ALL) TYPE=webadm_t ROLE=webadm_r /bin/sh
```



```
[en2@localhost:-$ id -Z  
cy5130_u:staff_r:staff_t:s0  
[en2@localhost:-$ sudo /bin/sh
```

We trust you have received the usual lecture from the local System Administrator. It usually boils down to these three things:

- #1) Respect the privacy of others.
- #2) Think before you type.
- #3) With great power comes great responsibility.

For security reasons, the password you type will not be visible.

```
[sudo] password for en2:  
[sh-5.2# id -Z  
cy5130_u:staff_r:staff_t:s0  
[en2@localhost:-$ sudo /bin/sh  
sh-5.2#
```



```
Last login: Thu Feb 19 10:05:57 2020  
[en2@localhost:-$ id -Z  
cy5130_u:staff_r:staff_t:s0  
[en2@localhost:-$ sudo /bin/sh  
[sudo] password for en2:  
[sh-5.2# id -Z  
cy5130_u:webadm_r:webadm_t:s0  
sh-5.2#
```

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:~$ ]
```

  

```
[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 http [REDACTED]  
<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 [REDACTED]  
<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 localhost [REDACTED] 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 localhost [REDACTED] 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.