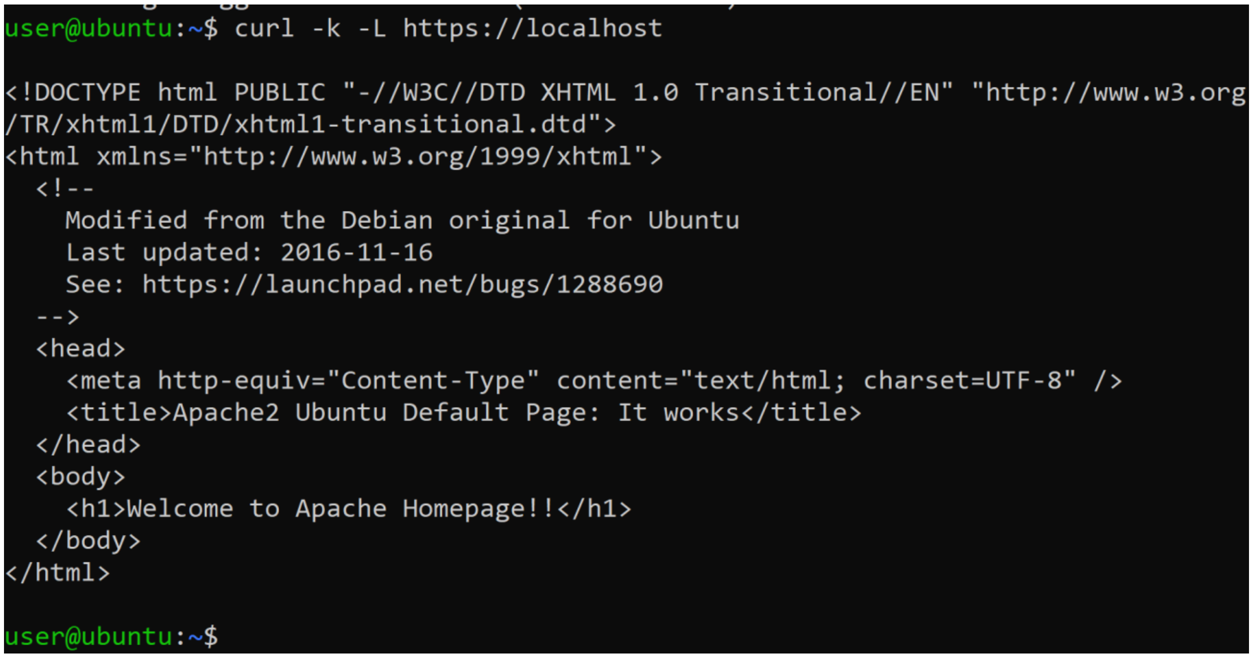
Lab 6: Software Security

PART 1: Setting Up the Lab

We set up the container in host VM, verified that ssh and apache are running on the container by ssh into the server via ssh admin@172.17.0.3 and curl -k -L https://localhost as seen on figure 1.

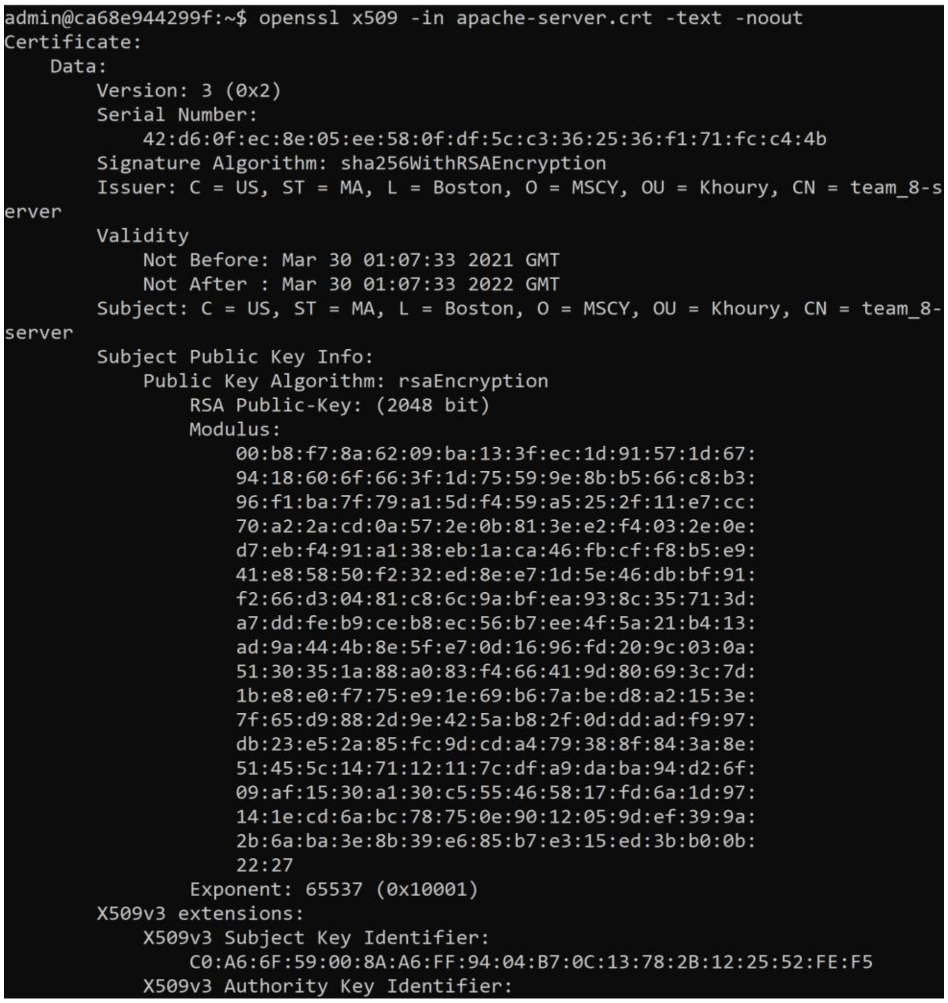


```
user@ubuntu:~$ curl -k -L https://localhost

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org
/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2016-11-16
    See: https://launchpad.net/bugs/1288690
  -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Apache2 Ubuntu Default Page: It works</title>
  </head>
  <body>
    <h1>Welcome to Apache Homepage!!</h1>
  </body>
</html>

user@ubuntu:~$
```

Figure 1

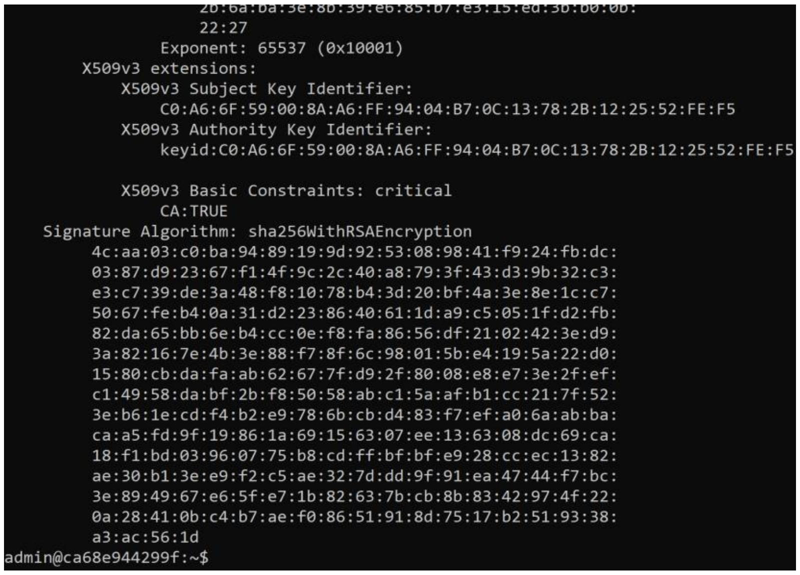
PART 2: Generating a Self-Signed Certificate

Generated a self-signed certificate inside the docker as seen below.



```
admin@ca68e944299f:~$ openssl x509 -in apache-server.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            42:d6:0f:ec:8e:05:ee:58:0f:df:5c:c3:36:25:36:f1:71:fc:c4:4b
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, ST = MA, L = Boston, O = MSCY, OU = Khoury, CN = team_8-s
erver
        Validity
            Not Before: Mar 30 01:07:33 2021 GMT
            Not After : Mar 30 01:07:33 2022 GMT
        Subject: C = US, ST = MA, L = Boston, O = MSCY, OU = Khoury, CN = team_8-
server
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:b8:f7:8a:62:09:ba:13:3f:ec:1d:91:57:1d:67:
                    94:18:60:6f:66:3f:1d:75:59:9e:8b:b5:66:c8:b3:
                    96:f1:ba:7f:79:a1:5d:f4:59:a5:25:2f:11:e7:cc:
                    70:a2:2a:cd:0a:57:2e:0b:81:3e:e2:f4:03:2e:0e:
                    d7:eb:f4:91:a1:38:eb:1a:ca:46:fb:cf:f8:b5:e9:
                    41:e8:58:50:f2:32:ed:8e:e7:1d:5e:46:db:bf:91:
                    f2:66:d3:04:81:c8:6c:9a:bf:ea:93:8c:35:71:3d:
                    a7:dd:fe:b9:ce:b8:ec:56:b7:ee:4f:5a:21:b4:13:
                    ad:9a:44:4b:8e:5f:e7:0d:16:96:fd:20:9c:03:0a:
                    51:30:35:1a:88:a0:83:f4:66:41:9d:80:69:3c:7d:
                    1b:e8:e0:f7:75:e9:1e:69:b6:7a:be:d8:a2:15:3e:
                    7f:65:d9:88:2d:9e:42:5a:b8:2f:0d:dd:ad:f9:97:
                    db:23:e5:2a:85:fc:9d:cd:a4:79:38:8f:84:3a:8e:
                    51:45:5c:14:71:12:11:7c:df:a9:da:ba:94:d2:6f:
                    09:af:15:30:a1:30:c5:55:46:58:17:fd:6a:1d:97:
                    14:1e:cd:6a:bc:78:75:0e:90:12:05:9d:ef:39:9a:
                    2b:6a:ba:3e:8b:39:e6:85:b7:e3:15:ed:3b:b0:0b:
                    22:27
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                C0:A6:6F:59:00:8A:A6:FF:94:04:B7:0C:13:78:2B:12:25:52:FE:F5
            X509v3 Authority Key Identifier:
```

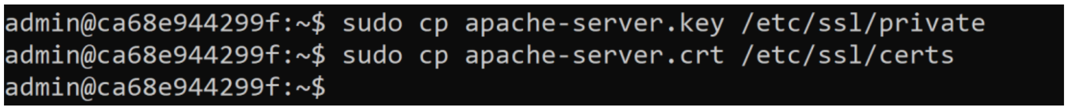
Figure 2



```
                    2b:6a:ba:3e:8b:39:e6:85:b7:e3:15:ed:3b:b0:0b:
                    22:27
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                C0:A6:6F:59:00:8A:A6:FF:94:04:B7:0C:13:78:2B:12:25:52:FE:F5
            X509v3 Authority Key Identifier:
                keyid:C0:A6:6F:59:00:8A:A6:FF:94:04:B7:0C:13:78:2B:12:25:52:FE:F5

            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
         4c:aa:03:c0:ba:94:89:19:9d:92:53:08:98:41:f9:24:fb:dc:
         03:87:d9:23:67:f1:4f:9c:2c:40:a8:79:3f:43:d3:9b:32:c3:
         e3:c7:39:de:3a:48:f8:10:78:b4:3d:20:bf:4a:3e:8e:1c:c7:
         50:67:fe:b4:0a:31:d2:23:86:40:61:1d:a9:c5:05:1f:d2:fb:
         82:da:65:bb:6e:b4:cc:0e:f8:fa:86:56:df:21:02:42:3e:d9:
         3a:82:16:7e:4b:3e:88:f7:8f:6c:98:01:5b:e4:19:5a:22:d0:
         15:80:cb:da:fa:ab:62:67:7f:d9:2f:80:08:e8:e7:3e:2f:ef:
         c1:49:58:da:bf:2b:f8:50:58:ab:c1:5a:af:b1:cc:21:7f:52:
         3e:b6:1e:cd:f4:b2:e9:78:6b:cb:d4:83:f7:ef:a0:6a:ab:ba:
         ca:a5:fd:9f:19:86:1a:69:15:63:07:ee:13:63:08:dc:69:ca:
         18:f1:bd:03:96:07:75:b8:cd:ff:bf:bf:e9:28:cc:ec:13:82:
         ae:30:b1:3e:e9:f2:c5:ae:32:7d:dd:9f:91:ea:47:44:f7:bc:
         3e:89:49:67:e6:5f:e7:1b:82:63:7b:cb:8b:83:42:97:4f:22:
         0a:28:41:0b:c4:b7:ae:f0:86:51:91:8d:75:17:b2:51:93:38:
         a3:ac:56:1d
admin@ca68e944299f:~$
```

Figure 3

Copied the certificate and private key to ssl folder as seen below.

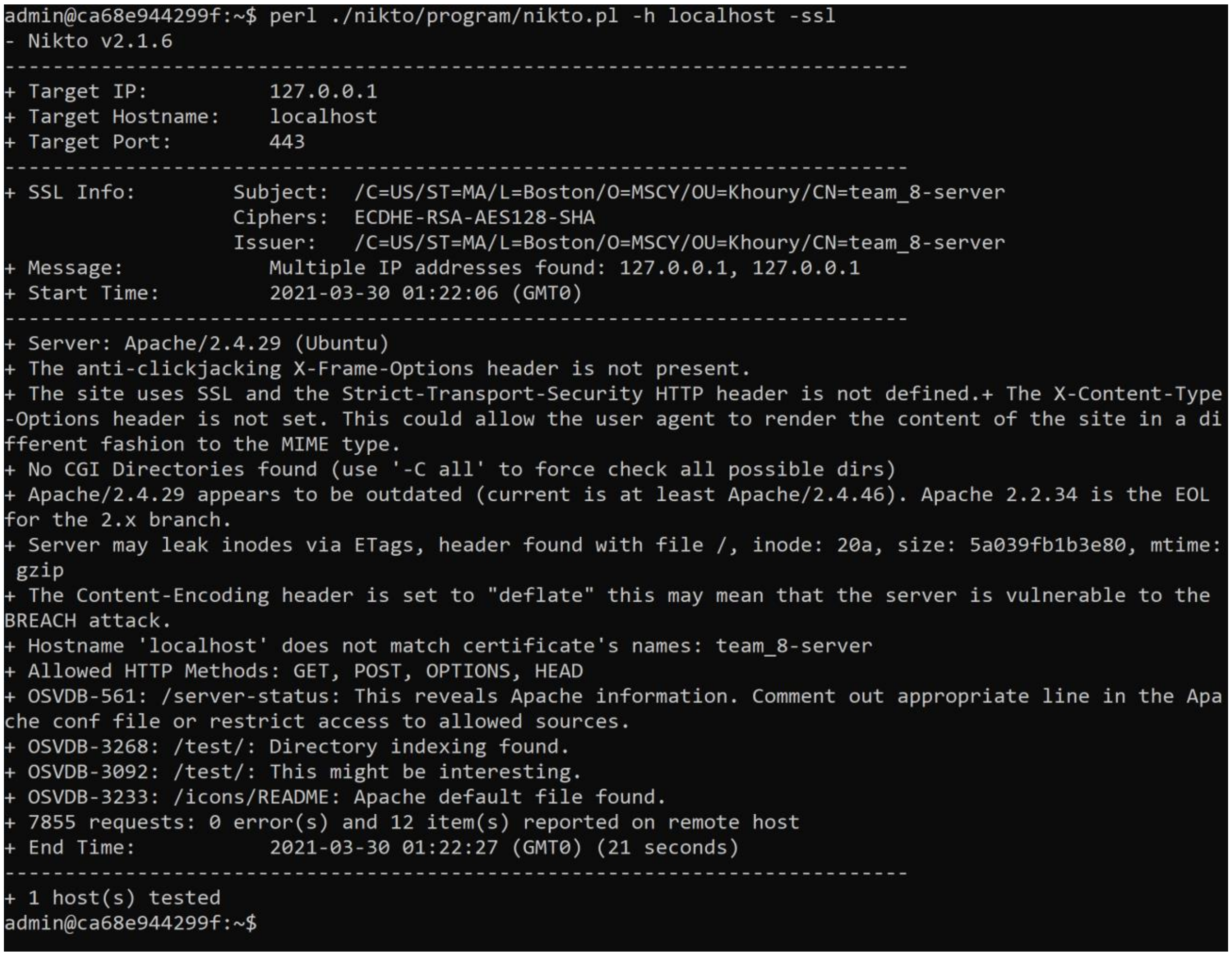


```
admin@ca68e944299f:~$ sudo cp apache-server.key /etc/ssl/private
admin@ca68e944299f:~$ sudo cp apache-server.crt /etc/ssl/certs
admin@ca68e944299f:~$
```

Figure 4

PART 3: Scanning with Nikto

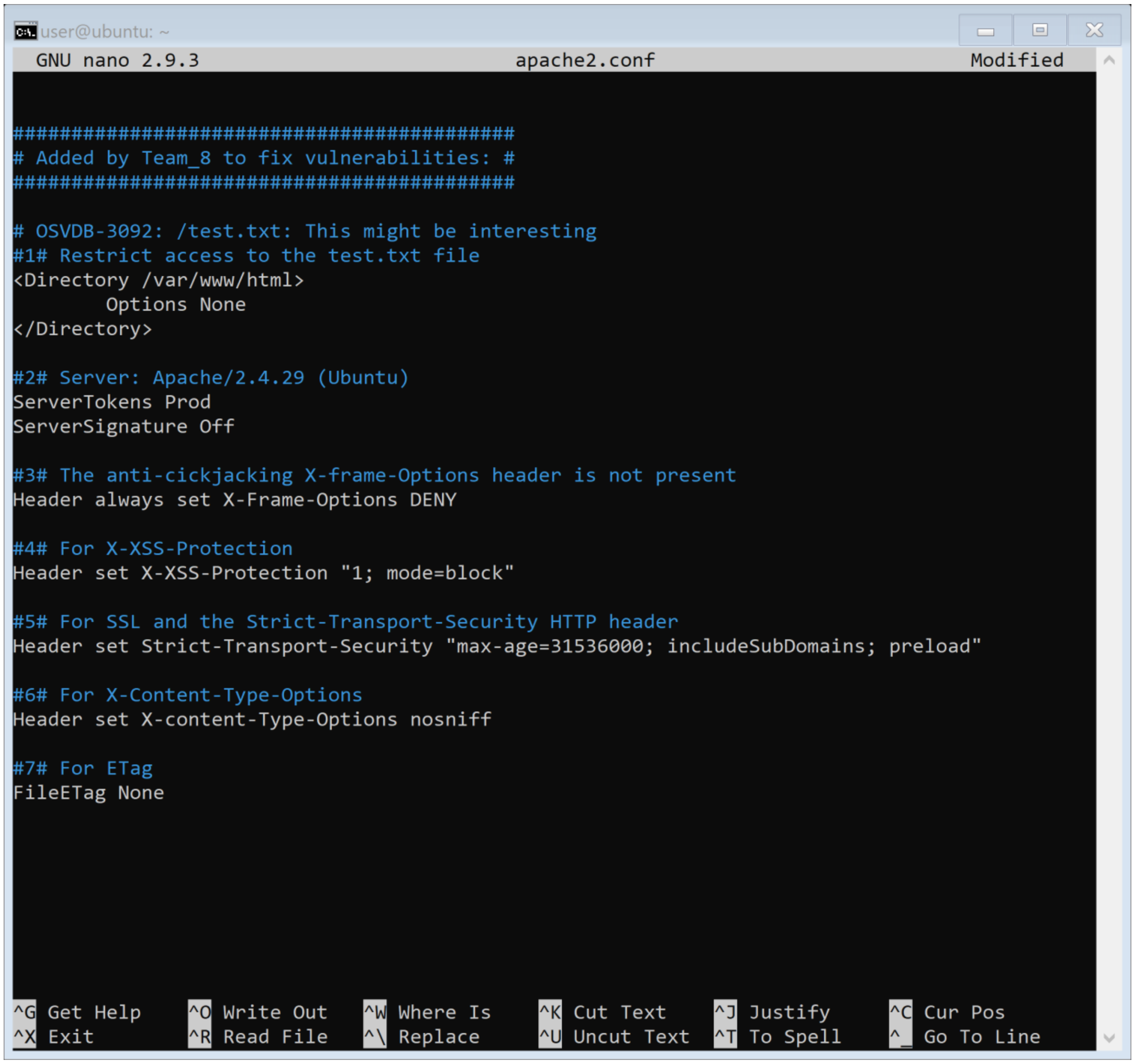
Scanned the apache webserver inside the container using nikto as seen in screenshot 3.



```
admin@ca68e944299f:~$ perl ./nikto/program/nikto.pl -h localhost -ssl
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          127.0.0.1
+ Target Hostname:    localhost
+ Target Port:        443
---------------------------------------------------------------------------
+ SSL Info:        Subject:  /C=US/ST=MA/L=Boston/O=MSCY/OU=Khoury/CN=team_8-server
                   Ciphers:  ECDHE-RSA-AES128-SHA
                   Issuer:   /C=US/ST=MA/L=Boston/O=MSCY/OU=Khoury/CN=team_8-server
+ Message:            Multiple IP addresses found: 127.0.0.1, 127.0.0.1
+ Start Time:         2021-03-30 01:22:06 (GMT0)
---------------------------------------------------------------------------
+ Server: Apache/2.4.29 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The site uses SSL and the Strict-Transport-Security HTTP header is not defined.+ The X-Content-Type
-Options header is not set. This could allow the user agent to render the content of the site in a di
fferent fashion to the MIME type.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.29 appears to be outdated (current is at least Apache/2.4.46). Apache 2.2.34 is the EOL
for the 2.x branch.
+ Server may leak inodes via ETags, header found with file /, inode: 20a, size: 5a039fb1b3e80, mtime:
 gzip
+ The Content-Encoding header is set to "deflate" this may mean that the server is vulnerable to the
BREACH attack.
+ Hostname 'localhost' does not match certificate's names: team_8-server
+ Allowed HTTP Methods: GET, POST, OPTIONS, HEAD
+ OSVDB-561: /server-status: This reveals Apache information. Comment out appropriate line in the Apa
che conf file or restrict access to allowed sources.
+ OSVDB-3268: /test/: Directory indexing found.
+ OSVDB-3092: /test/: This might be interesting.
+ OSVDB-3233: /icons/README: Apache default file found.
+ 7855 requests: 0 error(s) and 12 item(s) reported on remote host
+ End Time:           2021-03-30 01:22:27 (GMT0) (21 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
admin@ca68e944299f:~$
```

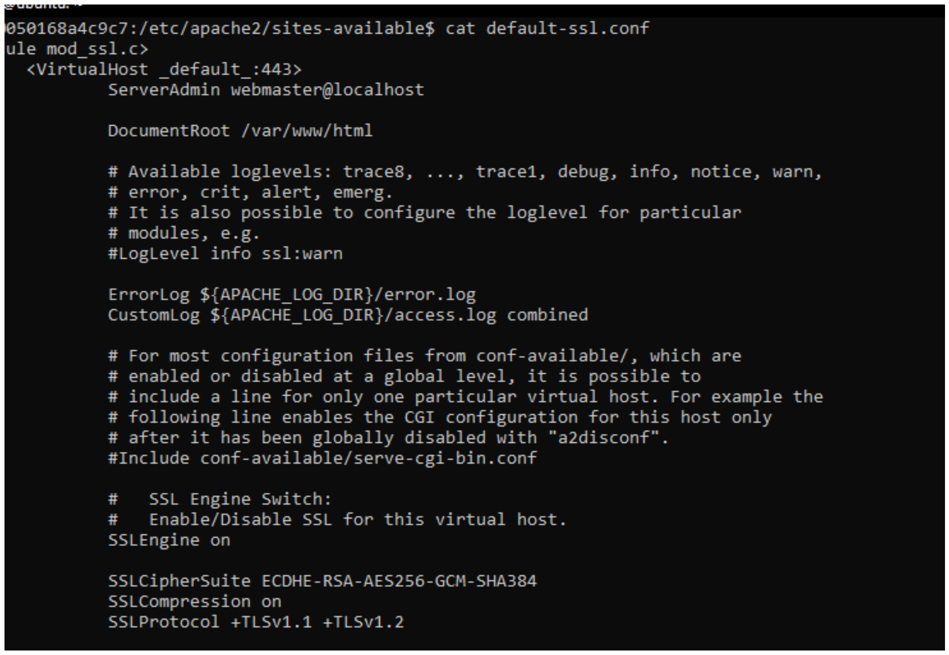
Figure 5: Scanning with nikto

PART 4: Fixing Vulnerabilities



```
user@ubuntu: ~
  GNU nano 2.9.3                        apache2.conf                        Modified


###########################################
# Added by Team_8 to fix vulnerabilities: #
###########################################

# OSVDB-3092: /test.txt: This might be interesting
#1# Restrict access to the test.txt file
<Directory /var/www/html>
        Options None
</Directory>

#2# Server: Apache/2.4.29 (Ubuntu)
ServerTokens Prod
ServerSignature Off

#3# The anti-cickjacking X-frame-Options header is not present
Header always set X-Frame-Options DENY

#4# For X-XSS-Protection
Header set X-XSS-Protection "1; mode=block"

#5# For SSL and the Strict-Transport-Security HTTP header
Header set Strict-Transport-Security "max-age=31536000; includeSubDomains; preload"

#6# For X-Content-Type-Options
Header set X-content-Type-Options nosniff

#7# For ETag
FileETag None






^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line
```

Figure 6_1 to 4_6



```
@ubuntu:~
050168a4c9c7:/etc/apache2/sites-available$ cat default-ssl.conf
ule mod_ssl.c>
  <VirtualHost _default_:443>
          ServerAdmin webmaster@localhost

          DocumentRoot /var/www/html

          # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
          # error, crit, alert, emerg.
          # It is also possible to configure the loglevel for particular
          # modules, e.g.
          #LogLevel info ssl:warn

          ErrorLog ${APACHE_LOG_DIR}/error.log
          CustomLog ${APACHE_LOG_DIR}/access.log combined

          # For most configuration files from conf-available/, which are
          # enabled or disabled at a global level, it is possible to
          # include a line for only one particular virtual host. For example the
          # following line enables the CGI configuration for this host only
          # after it has been globally disabled with "a2disconf".
          #Include conf-available/serve-cgi-bin.conf

          #   SSL Engine Switch:
          #   Enable/Disable SSL for this virtual host.
          SSLEngine on

          SSLCipherSuite ECDHE-RSA-AES256-GCM-SHA384
          SSLCompression on
          SSLProtocol +TLSv1.1 +TLSv1.2
```

Figure 7: Changing the cipher suite

Explanation to the fixes:

1. **OSVDB-3092: /test.txt: This might be interesting …**
   a. The test.txt file had no restrictions when it came to access. Anyone could have access to the file as well as the directory where the file was saved. Exposing files and folders under root. For instance, if one typed http://localhost/test they would be able to see the directories and subdirectories with root access. Thus, it is a potential vulnerability because it exposed information which could help an attacker.
   b. Issue was fixed by restricting access to the whole "/var/www/html" directory as shown above.
   c. By restricting access to the html directory, uses will have "You don't have permission to access /test" instead of getting all the files and folders in this directory. Hence, and attacker will have less information regarding what files and folders have root privilege.
2. **Server: Apache/2.4.29 (Ubuntu).**
   a. When scanned, nikto reveals that along with server name Apache, it displays the server version and the underlying OS name as a part of server response header. Here we are providing more information than necessary to any potential hacker (for instance, exploit any version specific vulnerabilities).
   b. So, displaying a generic information like just the name of the web server, we are making it more secure. This can be done by adding the line "ServerTokens Prod" in the apache2.conf file. This makes sure that we display only Apache in the response headers sent back to the client.

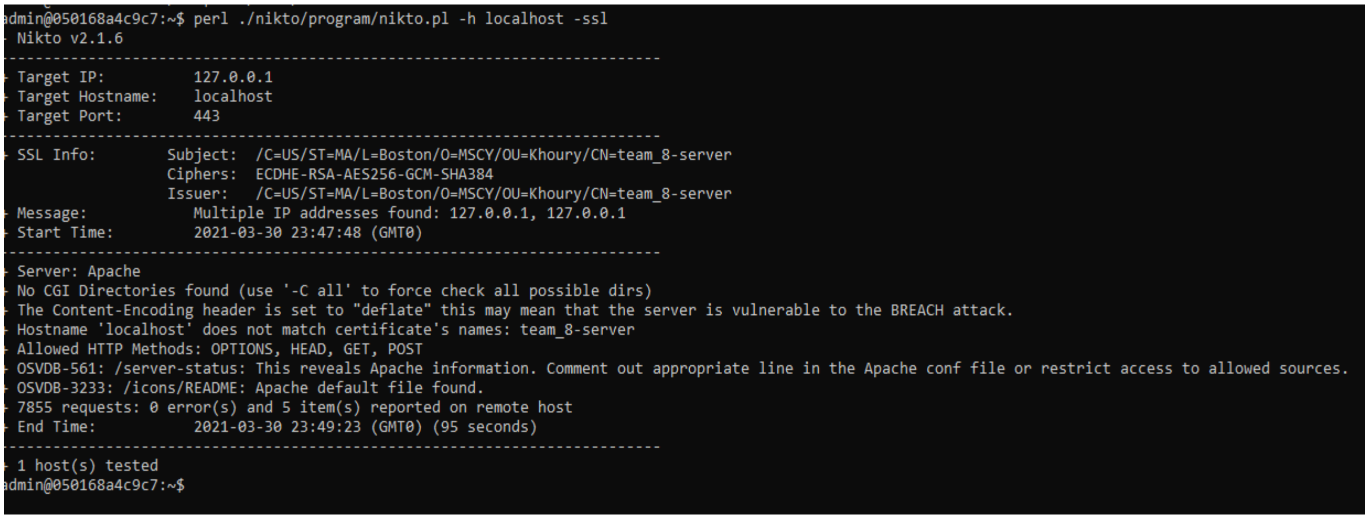
   c. In addition, we can add the line “ServerSignature Off” too to ensure that these version details are not displayed as a footer line in the server generated documents.

3. **The anti-clicking X-Frame-Options header is not present.**
   a. The absence of X-Frame-Options header in the server response points out that the websites might be vulnerable to clickjacking attack.
   b. Including this header in the server response ensures that X-Frame-Options allows sites to prevent their own content from being used in an invisible frame by attackers to carryout clickjacking attack.
   c. The X-Frame-Options header which has 2 directives sameorigin and deny. Deny is the most secure option which ensures that even the current page is not used in a frame to carry out the attack. This is done by adding the line “Header always set X-Frame-Options DENY” in the apache2.conf file.
4. **The X-XSS-Protection header is not defined.**
   a. XSS commonly known as Cross Site Scripting is an attack where the attacker injects a malicious code onto some trusted webpages. These webpages act as a medium to make the client browser to execute this code when they visit it.
   b. To protect against it we add X-XSS-Protection Header by including the line “X-XSS-Protection: 1; mode=block” that blocks pages from loading when they detect XSS attack (contributors).
   c. This reduces the risk of being attacked by blocking resources when XSS attacks are detected.
5. **The site uses SSL and the Strict-Transport-Security HTTP header is not defined.**
   a. Strict Transport Security HTTP header ensures all communication to and from the web browser are sent over HTTP Secure automatically. By having this header defined, we will avoid HTTP requesting HTTPS everytime.
   b. Added the line “Header Set Strict-Transport-Security “max-age=31536000; includeSubDomains; preload” to fix it. This informs visiting websites that the current website and all its subdomains are HTTPS-only. This should be accessed over the next 2 years – the number of max-age (Zbigniew).
   c. By setting this header we avoid unnecessary redundancy while ensuring that our communication channels are secured via HTTPS.
6. **The X-Content-Type-Options header is not set.**
   a. MIME sniffing is done by browsers where it infers the MIME type of the web resource rather than the content-type header. By doing this it is more vulnerable to cross site scripting. This can be fixed by specifying the X-Content-Type-Options to set it to no sniff.
7. **Server may leak inodes via ETags, header found with file /.**
   a. ETags provide sensitive informations such as inode number and child processes. Inodes tend to contain all the administrative data needed to read a file (Trounce). Thus, if exposed, can be a very high-risk weakness on a site.
   b. In “/etc/apache2/apache2.conf”, we added the line FileTag None.

   c. Of course, without ETag, sensitive information such as child processes and inodes will not be exposed reducing the risk of being attacked.

8. **Ciphers: ECDHE-RSA-AES128-SHA:**
   Here we update the cipher suite to more a more secure one which holds algorithms like sha384, AES256. We make sure that TLSv1.2 is enabled. This protocol is the mostly widely used secure protocol and supports the cipher-suite listed.

Scanned the apache web server hosted in the container after fixing the above Vulnerabilities. Below is a screen of the after effect.

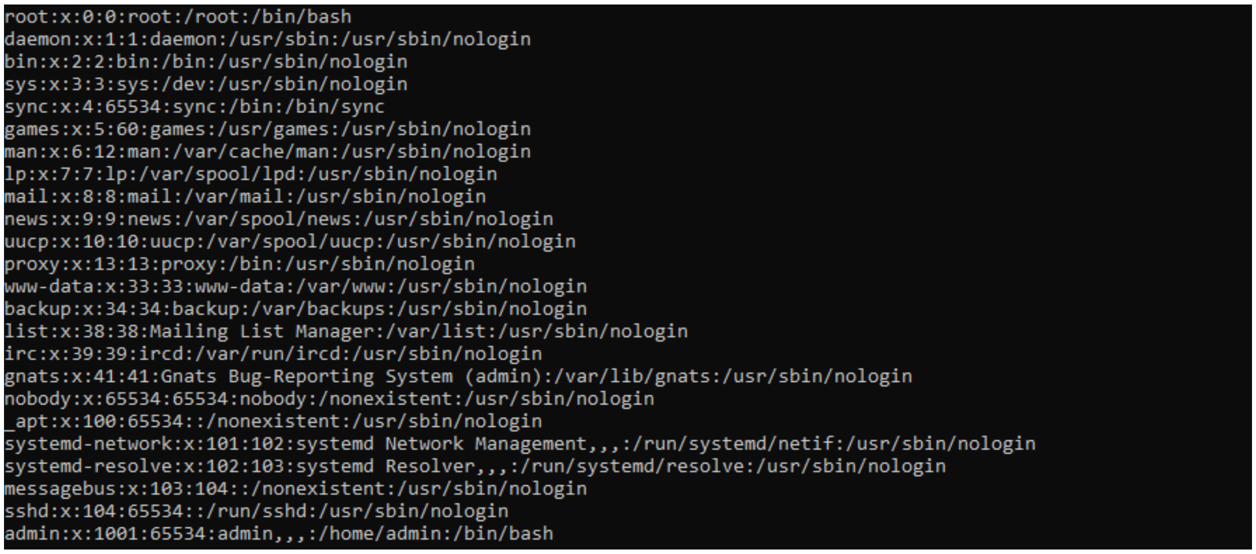


```
dmin@050168a4c9c7:~$ perl ./nikto/program/nikto.pl -h localhost -ssl
 Nikto v2.1.6
---------------------------------------------------------------------------
 Target IP:          127.0.0.1
 Target Hostname:    localhost
 Target Port:        443
---------------------------------------------------------------------------
 SSL Info:        Subject:  /C=US/ST=MA/L=Boston/O=MSCY/OU=Khoury/CN=team_8-server
                  Ciphers:  ECDHE-RSA-AES256-GCM-SHA384
                  Issuer:   /C=US/ST=MA/L=Boston/O=MSCY/OU=Khoury/CN=team_8-server
 Message:            Multiple IP addresses found: 127.0.0.1, 127.0.0.1
 Start Time:         2021-03-30 23:47:48 (GMT0)
---------------------------------------------------------------------------
 Server: Apache
 No CGI Directories found (use '-C all' to force check all possible dirs)
 The Content-Encoding header is set to "deflate" this may mean that the server is vulnerable to the BREACH attack.
 Hostname 'localhost' does not match certificate's names: team_8-server
 Allowed HTTP Methods: OPTIONS, HEAD, GET, POST
 OSVDB-561: /server-status: This reveals Apache information. Comment out appropriate line in the Apache conf file or restrict access to allowed sources.
 OSVDB-3233: /icons/README: Apache default file found.
 7855 requests: 0 error(s) and 5 item(s) reported on remote host
 End Time:           2021-03-30 23:49:23 (GMT0) (95 seconds)
---------------------------------------------------------------------------
 1 host(s) tested
dmin@050168a4c9c7:~$
```

Figure 8: Scan after fixing

PART 5: Hardening the OS

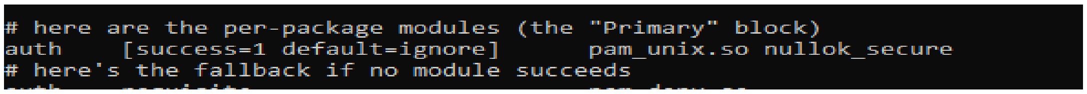
1. **Check user accounts in the system.**
   We can check all the users who can login into the system using the /etc/passwd file. The users with the 7th field holding the login shell are root and admin.



```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
messagebus:x:103:104::/nonexistent:/usr/sbin/nologin
sshd:x:104:65534::/run/sshd:/usr/sbin/nologin
admin:x:1001:65534:admin,,,:/home/admin:/bin/bash
```

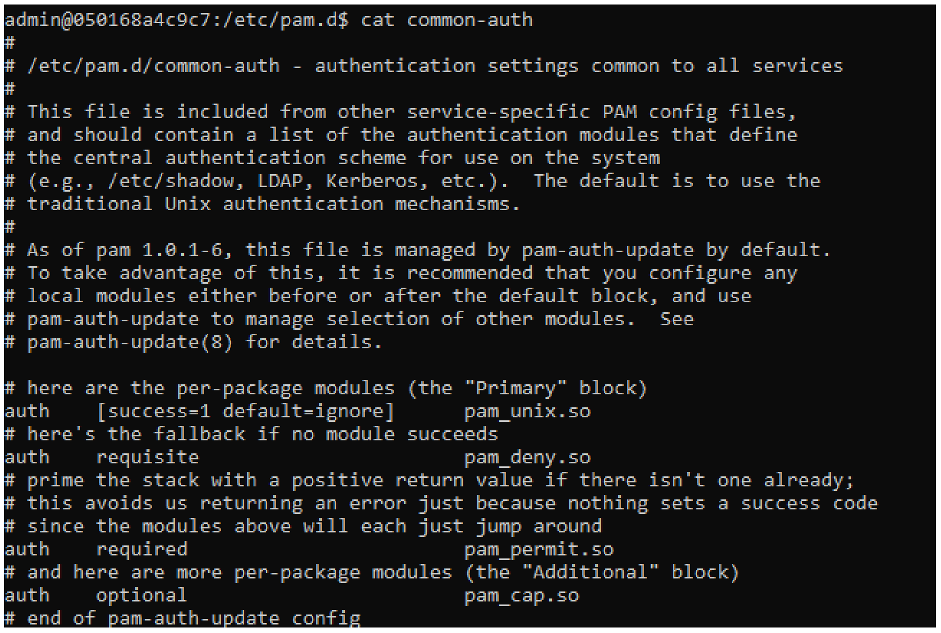
Figure 9: /etc/passwd

We are required to make sure that the can't login with empty passwords or insecure hashing methods. To do that we edited the common-auth file in /etc/pam.d and removed nullok_secure option to disable authentication with empty passswords as seen in the figures below.
Also, to make sure insecure hashing is not use we checked the common-password file to find that salted SHA-512 passwords are enabled. Since SHA-512 is secure we didn't make any changes.



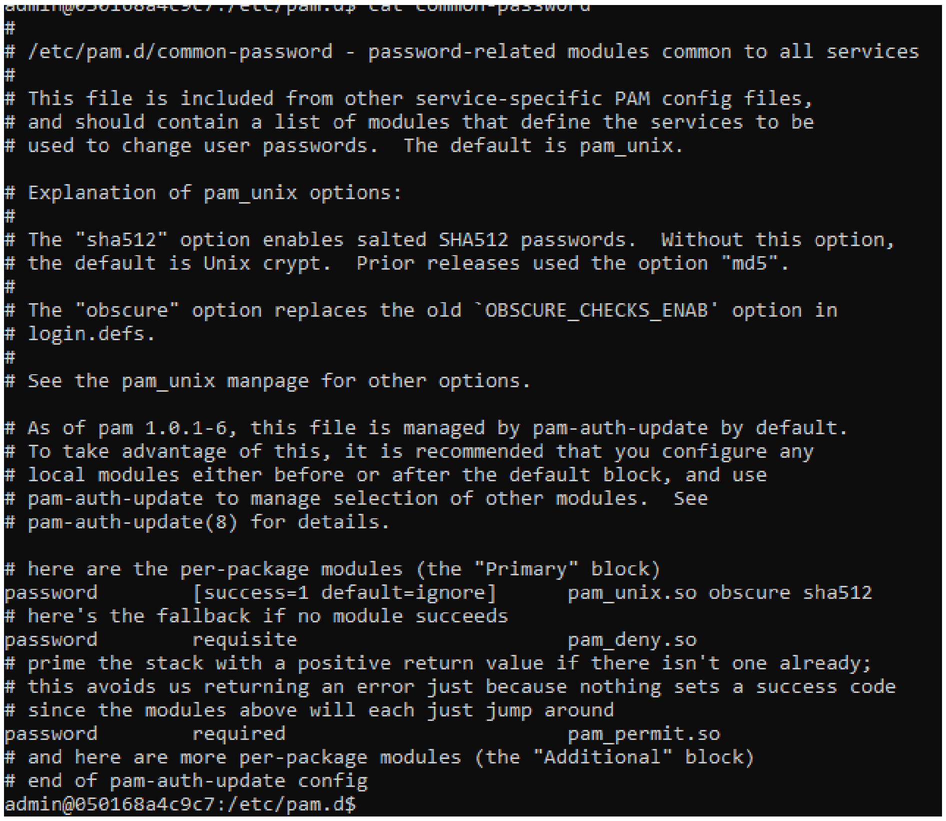
```
# here are the per-package modules (the "Primary" block)
auth     [success=1 default=ignore]      pam_unix.so nullok_secure
# here's the fallback if no module succeeds
```

Figure 10: common-auth (before)



```
admin@050168a4c9c7:/etc/pam.d$ cat common-auth
#
# /etc/pam.d/common-auth - authentication settings common to all services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authentication modules that define
# the central authentication scheme for use on the system
# (e.g., /etc/shadow, LDAP, Kerberos, etc.).  The default is to use the
# traditional Unix authentication mechanisms.
#
# As of pam 1.0.1-6, this file is managed by pam-auth-update by default.
# To take advantage of this, it is recommended that you configure any
# local modules either before or after the default block, and use
# pam-auth-update to manage selection of other modules.  See
# pam-auth-update(8) for details.

# here are the per-package modules (the "Primary" block)
auth    [success=1 default=ignore]      pam_unix.so
# here's the fallback if no module succeeds
auth    requisite                       pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
auth    required                        pam_permit.so
# and here are more per-package modules (the "Additional" block)
auth    optional                        pam_cap.so
# end of pam-auth-update config
```

Figure 11: common-auth (after)

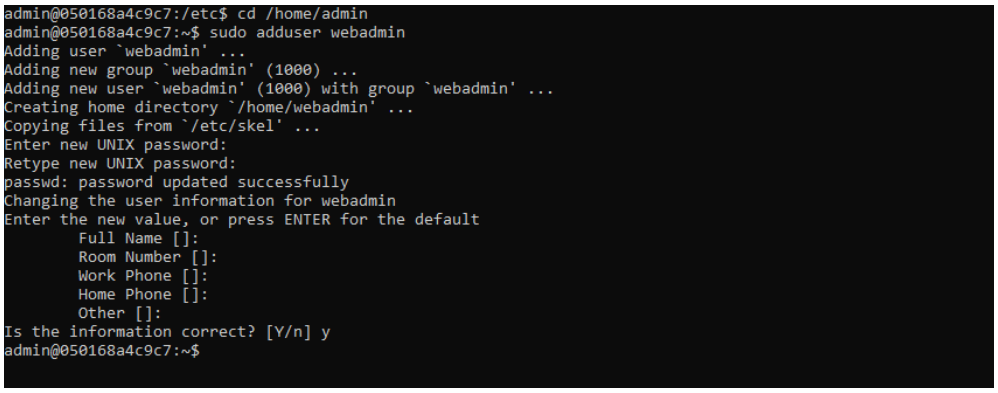


```
#
# /etc/pam.d/common-password - password-related modules common to all services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of modules that define the services to be
# used to change user passwords.  The default is pam_unix.

# Explanation of pam_unix options:
#
# The "sha512" option enables salted SHA512 passwords.  Without this option,
# the default is Unix crypt.  Prior releases used the option "md5".
#
# The "obscure" option replaces the old `OBSCURE_CHECKS_ENAB' option in
# login.defs.
#
# See the pam_unix manpage for other options.

# As of pam 1.0.1-6, this file is managed by pam-auth-update by default.
# To take advantage of this, it is recommended that you configure any
# local modules either before or after the default block, and use
# pam-auth-update to manage selection of other modules.  See
# pam-auth-update(8) for details.

# here are the per-package modules (the "Primary" block)
password        [success=1 default=ignore]      pam_unix.so obscure sha512
# here's the fallback if no module succeeds
password        requisite                       pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
password        required                        pam_permit.so
# and here are more per-package modules (the "Additional" block)
# end of pam-auth-update config
admin@050168a4c9c7:/etc/pam.d$
```

Figure 12: common-password

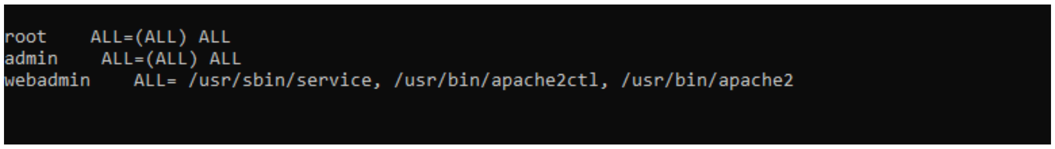
2. Sudo configuration:

- We created and added the user ‘webadmin’ with the password ‘webadmin’ as seen below.
- We then edited the sudoers file using visudo command to make sure that the webadmin user can sudo only apache2 and apache2ctl command as seen in figure 9.
- In the next step, we created a group called ‘web’, following the requirements, we added ‘admin’,’root’ and ‘webadmin’ users to this group. Additional users can simply be added to such groups which utilize the privileges it grants.
- Once the group is created, we change the group ownership of ‘/var/www/html’ to web which has the required write access.



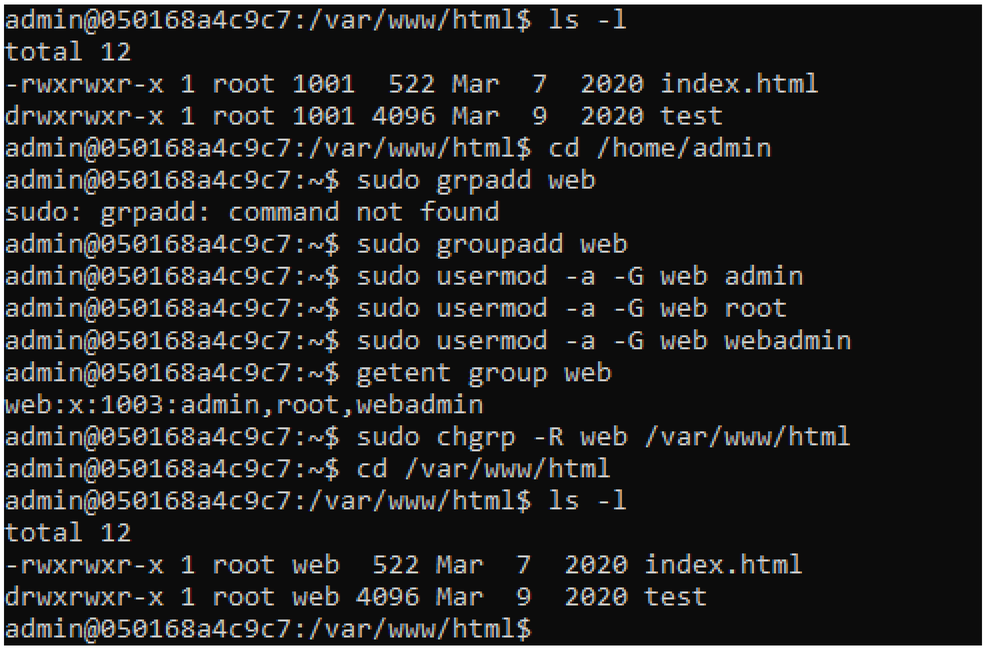
```
admin@050168a4c9c7:/etc$ cd /home/admin
admin@050168a4c9c7:~$ sudo adduser webadmin
Adding user `webadmin' ...
Adding new group `webadmin' (1000) ...
Adding new user `webadmin' (1000) with group `webadmin' ...
Creating home directory `/home/webadmin' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for webadmin
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
admin@050168a4c9c7:~$
```

Figure 13: adduser



```
root    ALL=(ALL) ALL
admin    ALL=(ALL) ALL
webadmin    ALL= /usr/sbin/service, /usr/bin/apache2ctl, /usr/bin/apache2
```

Figure 14: sudo permission for webadmin

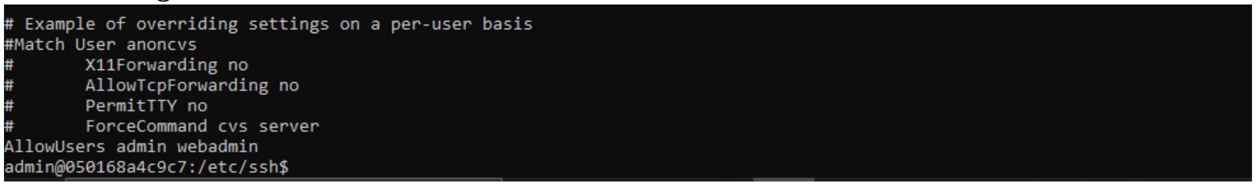


```
admin@050168a4c9c7:/var/www/html$ ls -l
total 12
-rwxrwxr-x 1 root 1001  522 Mar  7  2020 index.html
drwxrwxr-x 1 root 1001 4096 Mar  9  2020 test
admin@050168a4c9c7:/var/www/html$ cd /home/admin
admin@050168a4c9c7:~$ sudo grpadd web
sudo: grpadd: command not found
admin@050168a4c9c7:~$ sudo groupadd web
admin@050168a4c9c7:~$ sudo usermod -a -G web admin
admin@050168a4c9c7:~$ sudo usermod -a -G web root
admin@050168a4c9c7:~$ sudo usermod -a -G web webadmin
admin@050168a4c9c7:~$ getent group web
web:x:1003:admin,root,webadmin
admin@050168a4c9c7:~$ sudo chgrp -R web /var/www/html
admin@050168a4c9c7:~$ cd /var/www/html
admin@050168a4c9c7:/var/www/html$ ls -l
total 12
-rwxrwxr-x 1 root web  522 Mar  7  2020 index.html
drwxrwxr-x 1 root web 4096 Mar  9  2020 test
admin@050168a4c9c7:/var/www/html$
```

Figure 15: Changing /var/www/html permissions

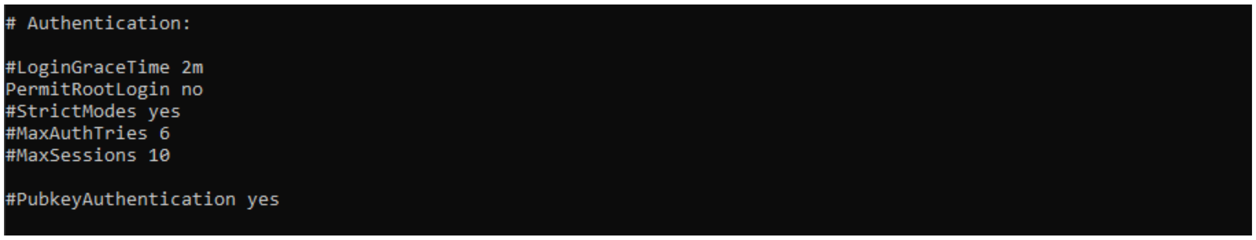
3. **Protections of the SSH server:**

- All users are allowed to ssh by default. Hence, we modified the /etc/ssh/sshd_config file to ensure that only admin and webadmin can ssh into the docker container by adding the line 'AllowUsers admin webadmin.
- AllowUsers keyword if mentioned, ensures that remote login is allowed only for usernames that matches one of the patterns specified.
- We also set 'PermitRootLogin no' to disable root login through ssh.
- The default port used for ssh is port 22. To change this, we added the line 'Port 475' in the sshd_config file as seen in figure 12 and restarted the ssh service to reflect the changes.
- We verified it by running netstat tool to check that the ssh port was changed successfully as seen in figures 13 and 14.



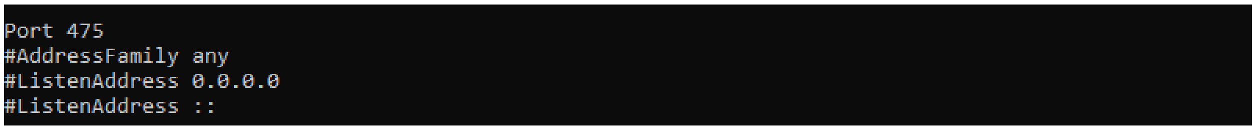
```
# Example of overriding settings on a per-user basis
#Match User anoncvs
#       X11Forwarding no
#       AllowTcpForwarding no
#       PermitTTY no
#       ForceCommand cvs server
AllowUsers admin webadmin
admin@050168a4c9c7:/etc/ssh$
```

Figure 14: AllowUsers



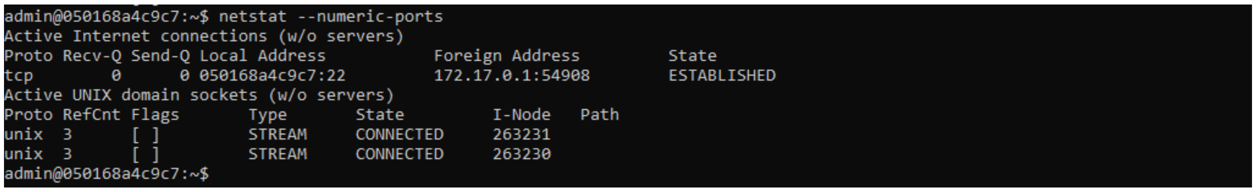
```
# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes
```

Figure 15: Disable RootLogin



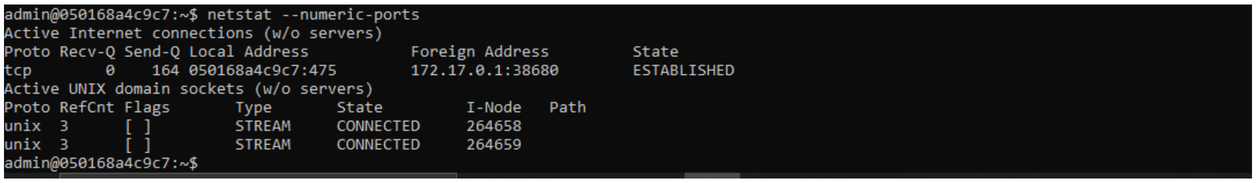
```
Port 475
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

Figure 16: Changing default port



```
admin@050168a4c9c7:~$ netstat --numeric-ports
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 050168a4c9c7:22         172.17.0.1:54908        ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  3      [ ]         STREAM     CONNECTED     263231
unix  3      [ ]         STREAM     CONNECTED     263230
admin@050168a4c9c7:~$
```

Figure 17: Netstat (port 22)

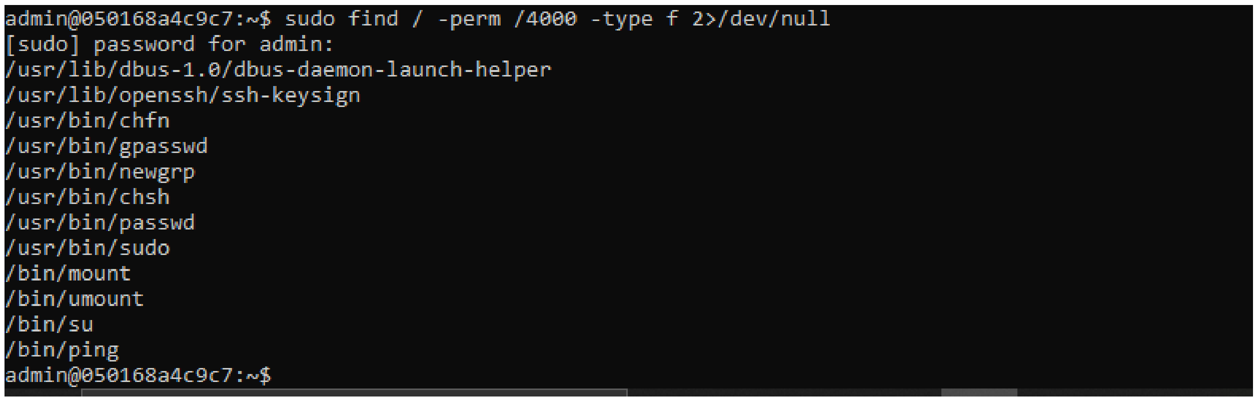


```
admin@050168a4c9c7:~$ netstat --numeric-ports
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0    164 050168a4c9c7:475        172.17.0.1:38680        ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  3      [ ]         STREAM     CONNECTED     264658
unix  3      [ ]         STREAM     CONNECTED     264659
admin@050168a4c9c7:~$
```

Figure 18: Netstat (port 475)

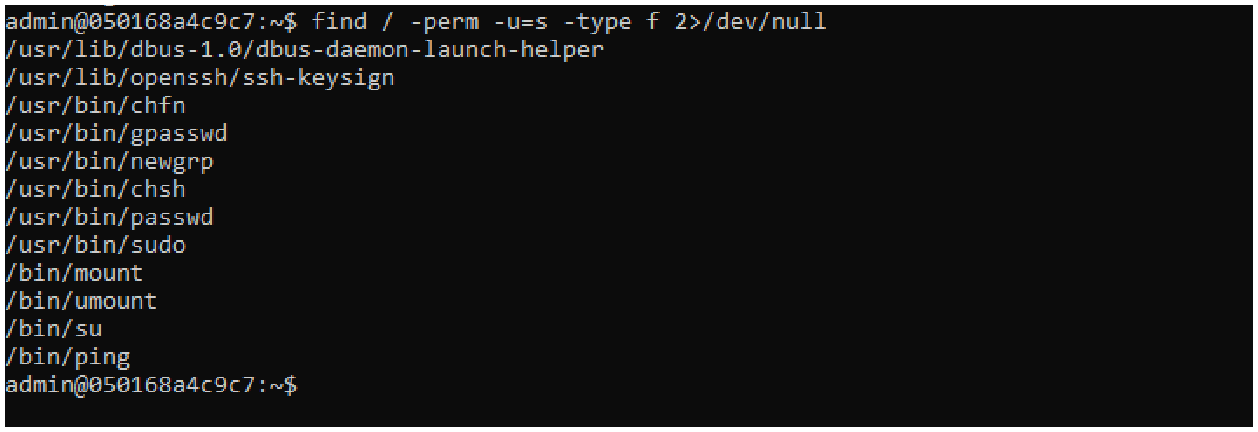
4. **Look for privilege elevation files:**

- To search for files with setuid and setgid in place we used find command. Permission for setuid is 4000. So, we searched the root directory (/) i.e the entire filesystem to find files with the the permission set to 4000.
- Alternatively, we can also use the command: **find / -perm -u=s -type f 2>/dev/null** where you are searching the entire directory where the user field permission has ‘s’ (setuid) set.
- 2>/dev/null is used to discard the stderr generated as result into character device /dev/null which simply just discards it.
- Similarly, for setgid the permission to search for is 2000. We can repeat the search as above using this permission value. (or by using the group option -g)



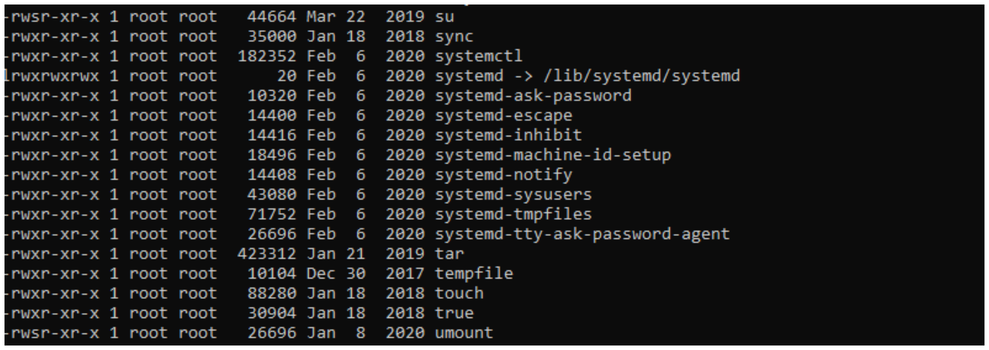
```
admin@050168a4c9c7:~$ sudo find / -perm /4000 -type f 2>/dev/null
[sudo] password for admin:
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/sudo
/bin/mount
/bin/umount
/bin/su
/bin/ping
admin@050168a4c9c7:~$
```

Figure 16: setuid check



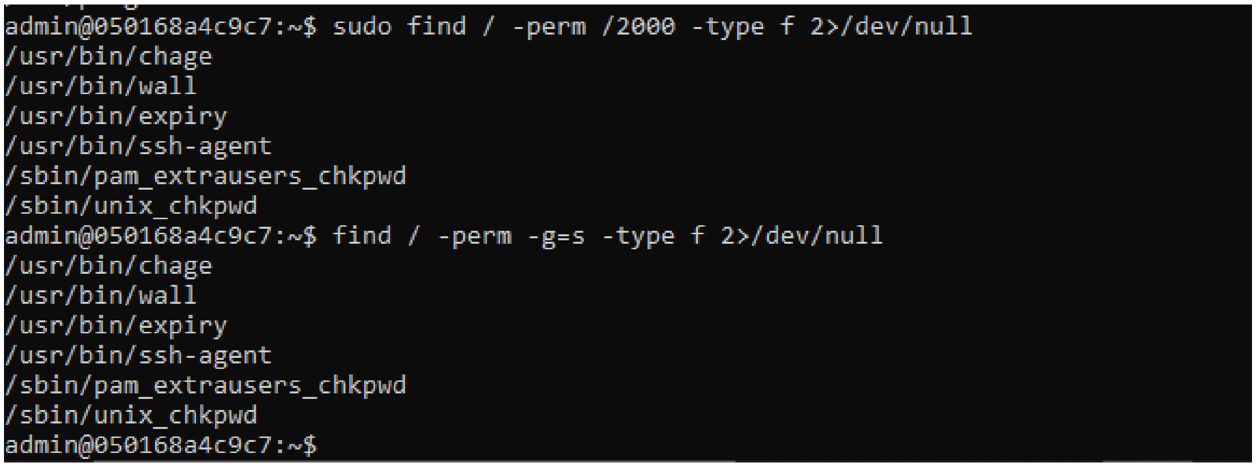
```
admin@050168a4c9c7:~$ find / -perm -u=s -type f 2>/dev/null
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/sudo
/bin/mount
/bin/umount
/bin/su
/bin/ping
admin@050168a4c9c7:~$
```

Figure 17: setuid check (alternate)



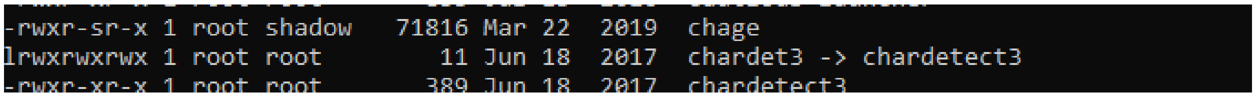
```
-rwsr-xr-x 1 root root   44664 Mar 22  2019 su
-rwxr-xr-x 1 root root   35000 Jan 18  2018 sync
-rwxr-xr-x 1 root root  182352 Feb  6  2020 systemctl
lrwxrwxrwx 1 root root      20 Feb  6  2020 systemd -> /lib/systemd/systemd
-rwxr-xr-x 1 root root   10320 Feb  6  2020 systemd-ask-password
-rwxr-xr-x 1 root root   14400 Feb  6  2020 systemd-escape
-rwxr-xr-x 1 root root   14416 Feb  6  2020 systemd-inhibit
-rwxr-xr-x 1 root root   18496 Feb  6  2020 systemd-machine-id-setup
-rwxr-xr-x 1 root root   14408 Feb  6  2020 systemd-notify
-rwxr-xr-x 1 root root   43080 Feb  6  2020 systemd-sysusers
-rwxr-xr-x 1 root root   71752 Feb  6  2020 systemd-tmpfiles
-rwxr-xr-x 1 root root   26696 Feb  6  2020 systemd-tty-ask-password-agent
-rwxr-xr-x 1 root root  423312 Jan 21  2019 tar
-rwxr-xr-x 1 root root   10104 Dec 30  2017 tempfile
-rwxr-xr-x 1 root root   88280 Jan 18  2018 touch
-rwxr-xr-x 1 root root   30904 Jan 18  2018 true
-rwsr-xr-x 1 root root   26696 Jan  8  2020 umount
```

Figure 18: Verifying with su and umount



```
admin@050168a4c9c7:~$ sudo find / -perm /2000 -type f 2>/dev/null
/usr/bin/chage
/usr/bin/wall
/usr/bin/expiry
/usr/bin/ssh-agent
/sbin/pam_extrausers_chkpwd
/sbin/unix_chkpwd
admin@050168a4c9c7:~$ find / -perm -g=s -type f 2>/dev/null
/usr/bin/chage
/usr/bin/wall
/usr/bin/expiry
/usr/bin/ssh-agent
/sbin/pam_extrausers_chkpwd
/sbin/unix_chkpwd
admin@050168a4c9c7:~$
```

Figure 19: check for setgid



```
-rwxr-sr-x 1 root shadow   71816 Mar 22  2019  chage
lrwxrwxrwx 1 root root        11 Jun 18  2017  chardet3 -> chardetect3
-rwxr-xr-x 1 root root       389 Jun 18  2017  chardetect3
```

Figure 20: verify with chage

PART 6: Submit the container

docker pull cy5010team8/lab6_software_security:team8

QUESTIONS:

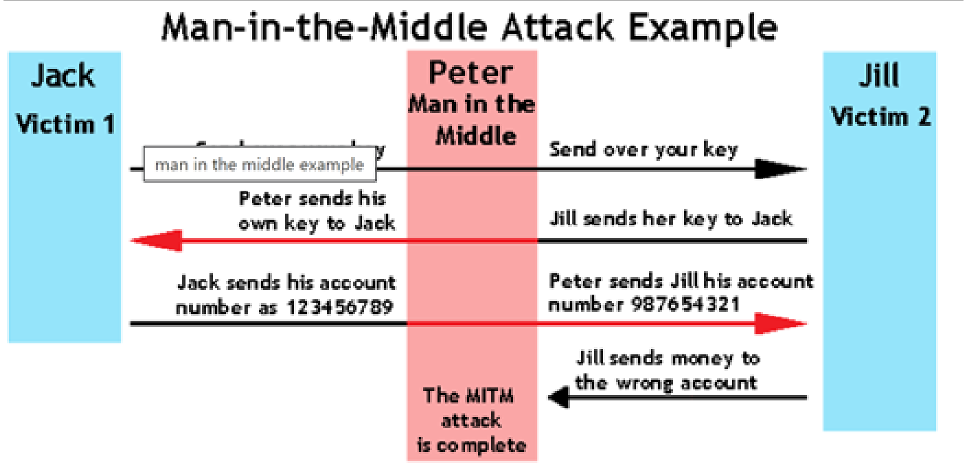
1. **Describe technically one attack which could compromise the confidentiality of a HTTP communication between a web server and a Browser. Would this attack be effective over a HTTPS communication?**

   MITM Attack:

   Man in the middle attack in HTTP communication are famous for eavesdropping the communication between the client – website to the server. Technically, the attacker intervenes the conversation and pretends to be a legitimate participant. This enables the attacker to get sensitive information in undetected ways.

   A good example of this attack as explained from Veracode.com is as follows:

   An attacker (Peter) position's himself in between Victim A and B with a fake chat service that impersonates a well-known bank.


Man-in-the-Middle Attack Example
Jack
Victim 1
Peter
Man in the Middle
Jill
Victim 2
man in the middle example
Send over your key
Peter sends his own key to Jack
Jill sends her key to Jack
Jack sends his account number as 123456789
Peter sends Jill his account number 987654321
Jill sends money to the wrong account
The MITM attack is complete

(Man in the Middle (MITM) Attack)

This attack could be stopped using HTTPS. HTTP Secure uses public-private key exchange and server validation using SLL certificates signed by a trusted third-party Certificate Authority. Thus, websites should force the use of HTTPS on all requests.

2. **You configured the web server using a self-signed certificate, what implications do you think that has according to the confidentiality of the communication between the browser and the web server?**

A Self-Signed Certificate is a security certificate that is not signed by a certificate authority (CA). These are free of cost and easy to generate. Even though a self-signed certificate can provide encryption like a normal certificate it is not well desired in most scenarios. Considering communication between the browser and the web server, the parties involved usually don't know one another, hence requiring a trusted third-party authority to validate the identification of the server to the client i.e Certificate Authority. When Self-signed certificate is in place it is more vulnerable to spoofing/man in the middle attacks. Hence, the usage of these certificates is limited to internal sites where the end parties are well-known to each other.
Another problem with certificates not issued by a public CA is when SSL interception is done. This is often the case not just in several companies but also several antivirus products do it. In this case sites using a certificate which cannot be verified will often simply be blocked by the SSL interception and one would need to explicitly add exceptions.

References

contributors, MDN. *MDN Web Docs*. 19 Feb 2021. Web. 29 Mar 2021. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection>.

"Man in the Middle (MITM) Attack." 2021. *Veracode.* Web. 30 Mar 2021. <https://www.veracode.com/security/man-middle-attack#:~:text=A%20man%2Din%2Dthe%2Dmiddle%20attack%20is%20a%20type,to%20be%20both%20legitimate%20participants.>.

Trounce, David. *Help Desk Geek*. 29 Feb 2020. Web. 29 Mar 2021. <https://helpdeskgeek.com/linux-tips/what-are-inodes-in-linux-and-how-are-they-used/>.

Zbigniew, Banach. "HTTP Security Headers: An Easy Way to Harden Your Web Applications." 05 Jun 2020. *Netsparker.* Web. 30 Mar 2021. <https://www.netsparker.com/blog/web-seccurity/http-security-headers/>.