HOCHSCHULE
CAMPUS WIEN
UNIVERSITY OF
APPLIED SCIENCES

Assignment 3

Cybercrime Defense

Secure HTTPS Webserver

Author:
Philip Magnus

Student identification number:
c2410537022

Date:
16th of December 2025

Contents

1. Introduction

1.1. Goal of the Assignment

The exercise consists of setting up a secure HTTPS web server with nginx that exclusively uses modern, strong TLS mechanisms.
The main goal is to configure the web server so that it achieves the maximum final score of 100 in the security test with testssl.sh.

1.2. Prerequisites

The assignment was completed with the following software and other prerequisites:

- Operating System: Ubuntu 24.04 LTS
- Docker or a Linux-VM with nginx-Installation
- OpenSSL for generating a self-signed certificate

1.3. Assignment Information

Setzen Sie einen HTTPS Webserver mit nginx auf (z.B. mittels Docker Container).
Dieser soll ausschließlich TLS 1.2 und TLS 1.3 unterstützen. Self-signed Zertifikate sind ausreichend.
Testen Sie Ihren Webserver mit testssl.sh (Website, GitHub). testssl.sh führt ein Rating anhand des “SSL Labs’s SSL Server Rating Guide” durch (Anm.: testssl.sh wird auch von Ivan Ristic empfohlen).
Das Ziel dieser Übung ist es, einen “Final Score” von 100 erreichen (= “Flag”).

Dokumentieren Sie in einem Protokoll (PDF) Ihre Vorgangsweise und alle notwendigen Schritte.

Die Übung haben Sie erfolgreich abgeschlossen, wenn

- 4 Punkte (Ziel): Sie einen “Final Score” von 100 erreichen (=“Flag”)
  - 2 Punkte: Ihr Webserver ausschließlich TLS 1.2 und TLS 1.3 unterstützt
  - 2 Punkte: TLS Session Tickets **deaktiviert** sind
    - Achtung: wird nicht farblich markiert von testssl.sh
    - Optional: Links für mehr Informationen
      We need to talk about Session Tickets
      We Really Need to Talk About Session Tickets: A Large-Scale Analysis of Cryptographic Dangers with TLS Session Tickets
  - Sie nur “grüne” Ausgaben von testssl.sh haben, also:
    - 2 Punkte: Keine weak Ciphers unterstützt werden (“not offered”)
    - 2 Punkte: Nur “Forward Secrecy strong encryption (AEAD ciphers)” unterstützt werden
    - 2 Punkte: Keine der getesteten Schwachstellen vorhanden sind (“not vulnerable”)
    - Ausnahmen:
      - Hinsichtlich (self-signed) Zertifikat: SAN, Trust, OCSP
      - DNS CAA RR
    - Ausführung von testssl.sh mittels (für Akzeptanz von Self-signed Cert): ./testssl.sh –add-ca </path/to/selfsigned.crt> :
      - via https://github.com/drwetter/testssl.sh/issues/1700#issuecomment-673520807
- Ihre Vorgehensweise nachvollziehbar dokumentiert haben
- Folgende Dateien in einem .zip-Archiv abgegeben haben:
  - Ihr Protokoll als PDF

- Ihren private Key,
- das zugehörige Zertifikat und
- Ihre (vollständige) finale nginx Config.

Links/Ressourcen:

- testssl.sh:
  - https://testssl.sh/
  - https://github.com/drwetter/testssl.sh
    - Für Self-signed Certs: https://github.com/drwetter/testssl.sh/issues/1700#issuecomment-673520807
- “SSL Labs’s SSL Server Rating Guide”
- OpenSSL Cookbook (Ivan Ristic)
- Sichere nginx TLS Config:
  - https://cipherlist.eu/
  - https://wiki.mozilla.org/Security/Server_Side_TLS
  - https://ssl-config.mozilla.org/

2. Assignment

2.1. Setup

First we create a new directory for our assignment and navigate into it:

```
$ mkdir ccd_assignment3
$ cd ccd_assignment3
```

Listing 1: Creating and navigating into assignment directory.

Before we started the setup of our `nginx` we downloaded the `testsslk.sh` test suite.
The download was done via `git clone` from the official GitHub repository:

```
$ git clone https://github.com/drwetter/testssl.sh
$ cd testssl.sh
$ chmod +x testssl.sh
```

Listing 2: Cloning the testssl.sh repository.

Before starting and setting up our `nginx` server, we need to check that `Docker` is installed and set up correctly.

```
$ docker --version
Docker version 29.1.1, build 0aedba5
```

Listing 3: Checking Docker installation.

After confirming `Docker`is installed we checked if the `Docker` daemon is running.

```
$ sudo systemctl status docker
[sudo] password for philip:
● docker.service - Docker Application Container Engine
    Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
    Active: active (running) since Sun 2025-11-30 19:00:48 CET; 24h ago
TriggeredBy: ● docker.socket
      Docs: https://docs.docker.com
  Main PID: 10839 (dockerd)
     Tasks: 24
    Memory: 30.1M (peak: 40.7M)
       CPU: 4.590s
    CGroup: /system.slice/docker.service
            └─10839 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

 [Ommited for better readability]
```

Listing 4: Checking if Docker service is running.

We also added our user to the `docker` group to be able to run `docker` commands without `sudo`:

```
$ sudo usermod -aG docker $USER
```

Listing 5: Adding user to docker group.

Next we pulled the `nginx` image from the Docker Hub repository to our local machine:

```
$ docker pull nginx
```

Listing 6: Pulling the nginx image from Docker Hub.

After pulling the image we checked if the image is available locally:

```
$ docker images
```

Listing 7: Pulling the nginx image from Docker Hub.

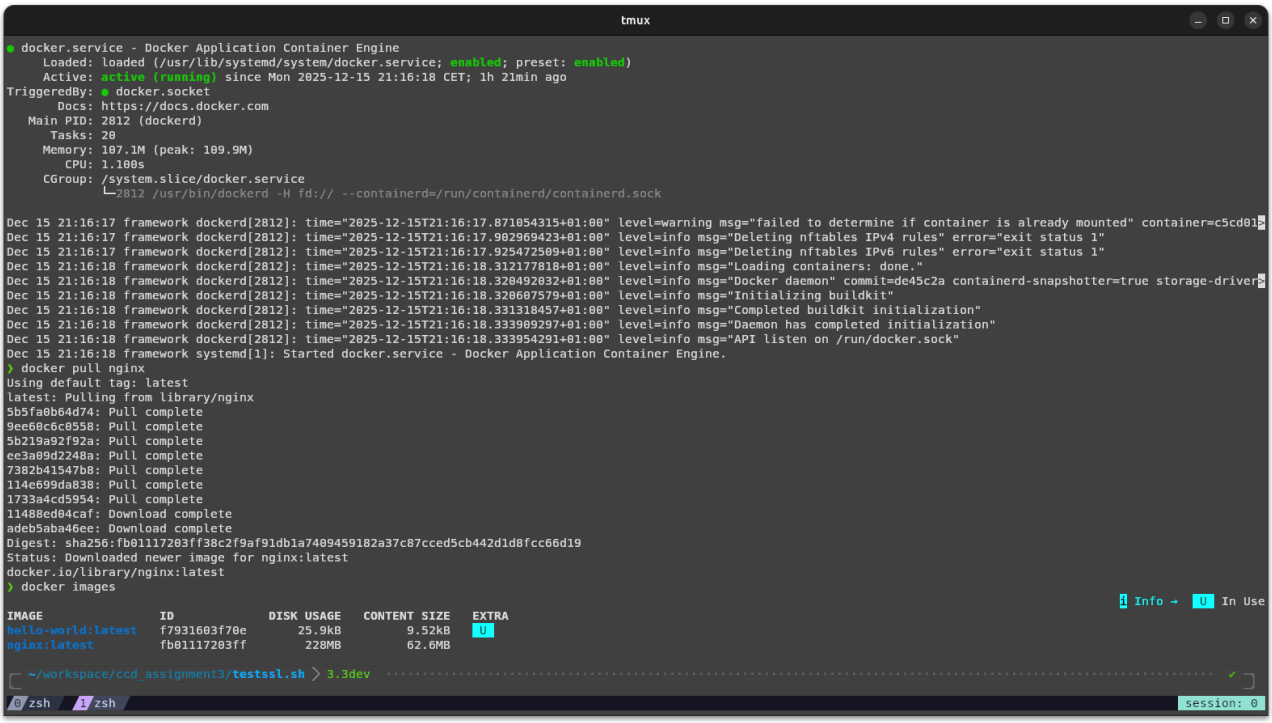

Figure 1: Listing local Docker images.

For a baseline test we started a simple `nginx` container without any custom configuration and ran the `testssl.sh` test suite against it:

```
$ docker run -d -p 80:80 -p443:443 --name nginx_test nginx
```

Listing 8: Running a simple nginx container for baseline testing.

```
$ ./testssl.sh --full localhost:443
```

Listing 9: Running testssl.sh against the nginx container.

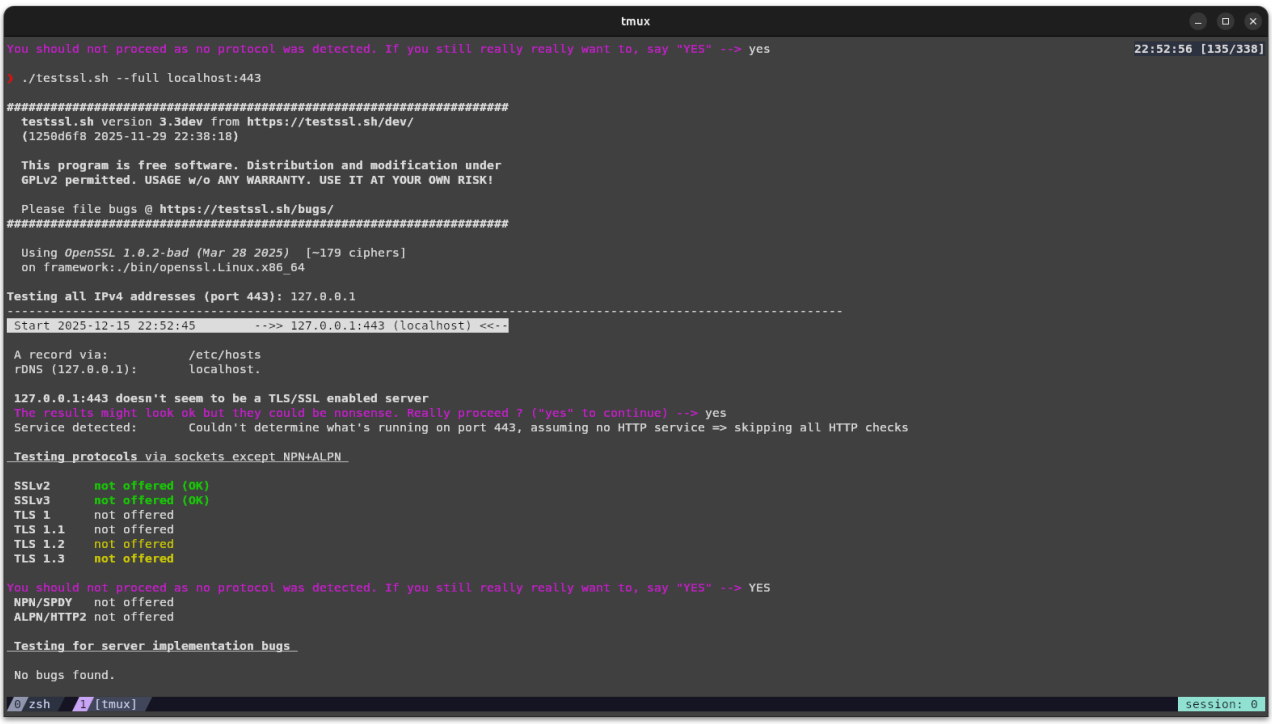

Figure 2: Testssl.sh results for the baseline nginx container.

As we can see from the results above `testssl.sh` says that no valid `TLS/SSL` service is enabled on the server.

2.2. Configuring HTTPS with Self-Signed Certificates

First we took a look at the default `nginx` configuration file located at `/etc/nginx/conf.d/default.conf` inside the container:

```
$ docker exec -it nginx_test /bin/bash

$ root@a2662f959694:/# cat /etc/nginx/conf.d/default.conf

  server {
      listen       80;
      listen  [::]:80;
      server_name  localhost;

      #access_log  /var/log/nginx/host.access.log  main;

      location / {
          root   /usr/share/nginx/html;
          index  index.html index.htm;
      }

      #error_page  404              /404.html;

      # redirect server error pages to the static page /50x.html
      #
      error_page   500 502 503 504  /50x.html;
      location = /50x.html {
          root   /usr/share/nginx/html;
      }

      # proxy the PHP scripts to Apache listening on 127.0.0.1:80
      #
      #location ~ \.php$ {
      #    proxy_pass   http://127.0.0.1;
      #}

      # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
      #
      #location ~ \.php$ {
      #    root           html;
      #    fastcgi_pass   127.0.0.1:9000;
      #    fastcgi_index  index.php;
      #    fastcgi_param  SCRIPT_FILENAME  /scripts$fastcgi_script_name;
      #    include        fastcgi_params;
      #}

      # deny access to .htaccess files, if Apache's document root
      # concurs with nginx's one
      #
      #location ~ /\.ht {
      #    deny  all;
      #}
  }
```

Listing 10: Viewing the default nginx configuration file.

In the configuation file we can see that the `nginx` server is set to listen on port `80` for `HTTP` traffic. There is also no configuration for `TLS/SSL` in form of a `TLS` block or certificates, which makes any `HTTPS` connection impossible.

Since we need to enable `HTTPS` on our server we created a self-signed certificate using `OpenSSL` with the following command:

```
$  openssl req -new -newkey rsa:4096 -nodes -keyout nginx.key -x509 -days 365 -out
nginx_test.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) [AU]:AU
State or Province Name (full name) [Some-State]:Vienna
Locality Name (eg, city) []:Vienna
Organization Name (eg, company) [Internet Widgits Pty Ltd]:HCW
Organizational Unit Name (eg, section) []:CCD
Common Name (e.g. server FQDN or YOUR name) []:localhost
Email Address []:test@hcw.ac.at
```

Listing 11: Generating a self-signed certificate using OpenSSL.

This command generates a new RSA private key (`nginx.key`) and a self-signed certificate (`nginx_test.crt`) valid for `365` days. The information in the certificate such as `Country Name`, `State`, `Organization`, and `Common Name` were filled out as prompted.

For organizational purposes we created a new directory called `tls` to store our generated certificate and key:

```
$ mkdir tls
$ mv nginx.key nginx.crt tls/
```

Listing 12: Move key and cert to tls directory.

To make the certificate and key available to the `nginx` container we need to copy the files into the `docker` container. We can do this using the `docker cp` command:

```
$ docker cp nginx_test.crt nginx_test:/etc/nginx/nginx_test.crt
  Successfully copied 4.1kB to nginx_test:/etc/nginx/nginx_test.crt
$ docker cp nginx.key nginx_test:/etc/nginx/nginx.key
  Successfully copied 5.12kB to nginx_test:/etc/nginx/nginx.key
```

Listing 13: Copying key and cert to nginx container.

Next we need to modify the `nginx` configuration to enable `HTTPS` on port 443 using our self-signed certificate. For this we edit the default configuration file located at `/etc/nginx/conf.d/default.conf` inside the container (for ease of use we create a new `default.conf` file outside the container and copy it in later):

```
$ vim default.conf

  server {
    listen 443 ssl;
    server_name localhost;

    ssl_certificate      /etc/nginx/tls/nginx_test.crt;
    ssl_certificate_key  /etc/nginx/tls/nginx.key;

    ssl_protocols TLSv1.2 TLSv1.3;

    ssl_session_tickets off;

    ssl_ciphers HIGH:!aNULL:!MD5:!SHA1:!RSA:!3DES;

    location / {
        root /usr/share/nginx/html;
        index index.html;
    }
  }
```

Listing 14: Editing the nginx configuration file.

```
$ docker cp default.conf nginx_test:/etc/nginx/conf.d/default.conf

  Successfully copied 2.05kB to nginx_test:/etc/nginx/conf.d/default.conf
```

Listing 15: Copying modified configuration into nginx container.

We finally moved the certificate and key into a new `tls` directory inside the container for better organization:

```
$ root@a2662f959694:/# cd /etc/nginx/
$ root@a2662f959694:/etc/nginx# mkdir tls
$ root@a2662f959694:/etc/nginx# mv nginx_test.crt nginx.key tls/
```

Listing 16: Copying key and certificate to tls directory.

Next we restarted the `nginx` server to apply the new configuration:

```
$ root@a2662f959694:/etc/nginx/tls# nginx -s reload

2025/12/15 22:32:05 [notice] 87#87: signal process started
```

Listing 17: Reloading nginx to apply new configuration.

We can now run the `testssl.sh` test suite again against our `nginx` server to see if `HTTPS` is now enabled:

```
$ /testssl.sh --full --add-ca tls/nginx.crt localhost:443
```

Listing 18: Running testssl.sh against nginx server with HTTPS enabled.

We can see that `SSL/TLS` is now enabled on our server. We used the `--add-ca` option to add our self-signed certificate as a trusted certificate authority for the test (full output in appendix 1).

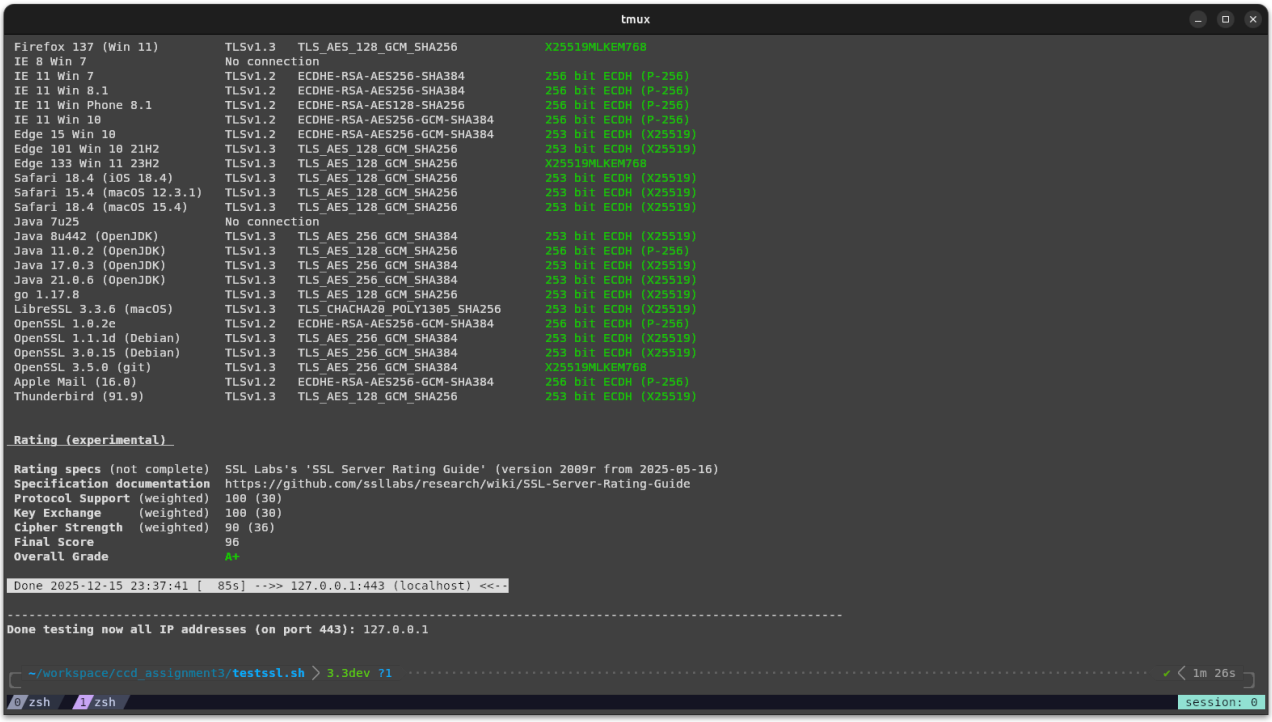

Figure 3: Testssl.sh results for nginx server with HTTPS enabled.

2.3. Fine Tuning and Hardening

However, this is not the final configuration we want to use. We want to further harden our TLS configuration by disabling weak ciphers and protocols.

Since the previous configuration was only used as a simple test, we decided to create a completely new configuration file based on the current Mozilla Security Guidelines (Modern TLS). In order to cleanly replace the existing settings, the previous file was revised and reloaded into the container as before. The previous minimal configuration has now been replaced by a comprehensively hardened server block:

```
$ vim default.conf

  server {
    listen 80;
    server_name localhost;
    # HTTP → HTTPS Weiterleitung
    return 301 https://$host$request_uri;
  }

  server {
      listen 443 ssl;
      server_name localhost;

      ssl_certificate      /etc/nginx/tls/nginx_test.crt;
      ssl_certificate_key  /etc/nginx/tls/nginx.key;

      ssl_protocols TLSv1.2 TLSv1.3;

      ssl_session_tickets off;

      ssl_ciphers 'TLS_AES_256_GCM_SHA384:ECDHE-RSA-AES256-GCM-SHA384';

      # TLS 1.3 Cipher suites overwrite (necessary for full points)
      ssl_conf_command Ciphersuites TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256;

      ssl_ecdh_curve secp384r1;

      ssl_prefer_server_ciphers on;

      ssl_session_cache shared:SSL:1m;
      ssl_session_timeout 1m;

      add_header Strict-Transport-Security "max-age=31536000; includeSubDomains; preload"
always;

      location / {
          root /usr/share/nginx/html;
          index index.html;
      }
    }
```

Listing 19: Running testssl.sh against nginx server with HTTPS enabled.

```
$ docker cp tls/default.conf nginx_test:/etc/nginx/conf.d/default.conf

  Successfully copied 2.56kB to nginx_test:/etc/nginx/conf.d/default.conf
```

Listing 20: Copying hardened configuration into nginx container.

Finally we reloaded the `nginx` server to apply the new hardened configuration:

```
$ docker exec -it nginx_test nginx -s reload

2025/12/15 22:51:35 [notice] 104#104: signal process started
```

Listing 21: Reloading hardened configuration.

We can now run the `testssl.sh` test suite again against our hardened `nginx` server to see the results with the maximum score possible:

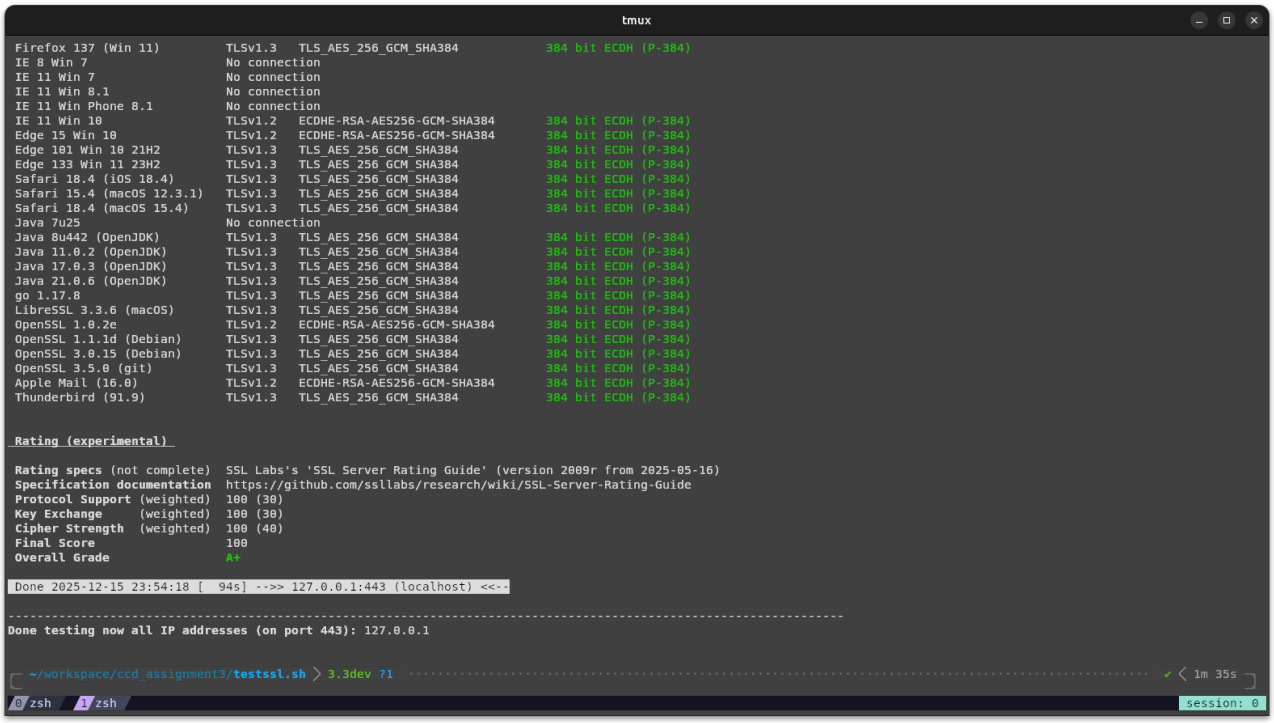

Figure 4: Testssl.sh results for nginx with max score.

The full output of the final testssl.sh run can be found in appendix 2.

Finally the key, certificate, and configuration were packed into a `zip` including this writeup for submission.

3. Appendix 1

```
$ ./testssl.sh --full --add-ca tls/nginx_test.crt localhost:443

  ###########################################################################
    testssl.sh version 3.3dev from https://testssl.sh/dev/
    (1250d6f8 2025-11-29 22:38:18)

    This program is free software. Distribution and modification under
    GPLv2 permitted. USAGE w/o ANY WARRANTY. USE IT AT YOUR OWN RISK!

    Please file bugs @ https://testssl.sh/bugs/
  ###########################################################################

    Using OpenSSL 1.0.2-bad (Mar 28 2025)  [~179 ciphers]
    on framework:./bin/openssl.Linux.x86_64

  Testing all IPv4 addresses (port 443): 127.0.0.1
--------------------------------------------------------------------------------------------------------------
  Start 2025-12-15 23:36:20        -->> 127.0.0.1:443 (localhost) <<--

  A record via:           /etc/hosts
  rDNS (127.0.0.1):       localhost.
  Service detected:       HTTP

  Testing protocols via sockets except NPN+ALPN

  SSLv2      not offered (OK)
  SSLv3      not offered (OK)
  TLS 1      not offered
  TLS 1.1    not offered
  TLS 1.2    offered (OK)
  TLS 1.3    offered (OK): final
  QUIC       Local problem: No OpenSSL QUIC support
  NPN/SPDY   not offered
  ALPN/HTTP2 http/1.1 (offered)

  Testing for server implementation bugs

  No bugs found.

  Testing cipher categories

  NULL ciphers (no encryption)                      not offered (OK)
  Anonymous NULL Ciphers (no authentication)        not offered (OK)
  Export ciphers (w/o ADH+NULL)                     not offered (OK)
  LOW: 64 Bit + DES, RC[2,4], MD5 (w/o export)      not offered (OK)
  Triple DES Ciphers / IDEA                         not offered
  Obsoleted CBC ciphers (AES, ARIA etc.)            offered
  Strong encryption (AEAD ciphers) with no FS       not offered
  Forward Secrecy strong encryption (AEAD ciphers)  offered (OK)


  Testing server's cipher preferences

  Hexcode  Cipher Suite Name (OpenSSL)       KeyExch.   Encryption  Bits     Cipher Suite
Name (IANA/RFC)
```

```
--------------------------------------------------------------------------------------------------------
  SSLv2
  -
  SSLv3
  -
  TLSv1
  -
  TLSv1.1
  -
  TLSv1.2 (no server order, thus listed by strength)
  xc030   ECDHE-RSA-AES256-GCM-SHA384       ECDH 521   AESGCM      256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
  xc028   ECDHE-RSA-AES256-SHA384           ECDH 521   AES         256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
  xcca8   ECDHE-RSA-CHACHA20-POLY1305       ECDH 521   ChaCha20    256
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
  xc077   ECDHE-RSA-CAMELLIA256-SHA384      ECDH 521   Camellia    256
TLS_ECDHE_RSA_WITH_CAMELLIA_256_CBC_SHA384
  xc061   ECDHE-ARIA256-GCM-SHA384          ECDH 521   ARIAGCM     256
TLS_ECDHE_RSA_WITH_ARIA_256_GCM_SHA384
  xc02f   ECDHE-RSA-AES128-GCM-SHA256       ECDH 521   AESGCM      128
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
  xc027   ECDHE-RSA-AES128-SHA256           ECDH 521   AES         128
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
  xc076   ECDHE-RSA-CAMELLIA128-SHA256      ECDH 521   Camellia    128
TLS_ECDHE_RSA_WITH_CAMELLIA_128_CBC_SHA256
  xc060   ECDHE-ARIA128-GCM-SHA256          ECDH 521   ARIAGCM     128
TLS_ECDHE_RSA_WITH_ARIA_128_GCM_SHA256
  TLSv1.3 (no server order, thus listed by strength)
  x1302   TLS_AES_256_GCM_SHA384            ECDH/MLKEM AESGCM      256
TLS_AES_256_GCM_SHA384
  x1303   TLS_CHACHA20_POLY1305_SHA256      ECDH/MLKEM ChaCha20    256
TLS_CHACHA20_POLY1305_SHA256
  x1301   TLS_AES_128_GCM_SHA256            ECDH/MLKEM AESGCM      128
TLS_AES_128_GCM_SHA256

  Has server cipher order?     no (NOT ok)
  (limited sense as client will pick)

  Testing robust forward secrecy (FS) -- omitting Null Authentication/Encryption, 3DES,
RC4

  FS is offered (OK)           TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 ECDHE-
RSA-AES256-GCM-SHA384 ECDHE-RSA-AES256-SHA384 ECDHE-RSA-CHACHA20-POLY1305
                               ECDHE-RSA-CAMELLIA256-SHA384 ECDHE-ARIA256-GCM-SHA384
TLS_AES_128_GCM_SHA256 ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-SHA256
                               ECDHE-RSA-CAMELLIA128-SHA256 ECDHE-ARIA128-GCM-SHA256
  KEMs offered                 X25519MLKEM768
  Elliptic curves offered:     prime256v1 secp384r1 secp521r1 X25519 X448
  Finite field group:          ffdhe2048 ffdhe3072
  TLS 1.2 sig_algs offered:    RSA-PSS-RSAE+SHA512 RSA-PSS-RSAE+SHA384 RSA-PSS-RSAE+SHA256
RSA+SHA512 RSA+SHA384 RSA+SHA256 RSA+SHA224
  TLS 1.3 sig_algs offered:    RSA-PSS-RSAE+SHA512 RSA-PSS-RSAE+SHA384 RSA-PSS-RSAE+SHA256

  Testing server defaults (Server Hello)

  TLS extensions (standard)    "server name/#0" "max fragment length/#1"
"supported_groups/#10" "EC point formats/#11" "application layer protocol negotiation/#16"
                               "encrypt-then-mac/#22" "extended master secret/#23"
```

```
"supported versions/#43" "key share/#51" "renegotiation info/#65281"
  Session Ticket RFC 5077 hint no -- no lifetime advertised
  SSL Session ID support       yes
  Session Resumption           tickets no, ID: no
  TLS 1.3 early data support   no early data offered
  TLS clock skew               Random values, no fingerprinting possible
  Certificate Compression      none
  Client Authentication        none
  Signature Algorithm          SHA256 with RSA
  Server key size              RSA 4096 bits (exponent is 65537)
  Server key usage             --
  Server extended key usage    --
  Serial                       2CAD8D45D7ACEF3F00339C5BB56A42AE404F23B7 (OK: length 20)
  Fingerprints                 SHA1 04CD6B99F36DAC7F4DCCDB1B6C258EB2D53EC793
                                SHA256
478AF59EAF01A4A237EE8571E8F1889D4991EB29675460E94ED4932F1D3B0978
  Common Name (CN)             localhost
  subjectAltName (SAN)         missing (NOT ok) -- Browsers are complaining
  Trust (hostname)             via CN only -- Browsers are complaining (same w/o SNI)
  Chain of trust               Ok
  EV cert (experimental)       no
  Certificate Validity (UTC)   364 >= 60 days (2025-12-15 22:07 --> 2026-12-15 22:07)
  ETS/"eTLS", visibility info  not present
  Certificate Revocation List  --
  OCSP URI                     --
                                NOT ok -- neither CRL nor OCSP URI provided
  OCSP stapling                not offered
  OCSP must staple extension   --
  DNS CAA RR (experimental)    not offered
  Certificate Transparency     --
  Certificates provided        1
  Issuer                       localhost (HCW from AU)
  Intermediate Bad OCSP (exp.) Ok


  Testing HTTP header response @ "/"

  HTTP Status Code             200 OK
  HTTP clock skew              0 sec from localtime
  Strict Transport Security    not offered
  Public Key Pinning           --
  Server banner                nginx/1.29.4
  Application banner           --
  Cookie(s)                    (none issued at "/")
  Security headers             --
  Reverse Proxy banner         --


  Testing vulnerabilities

  Heartbleed (CVE-2014-0160)               not vulnerable (OK), no heartbeat extension
  CCS (CVE-2014-0224)                      not vulnerable (OK)
  Ticketbleed (CVE-2016-9244), experiment.  not vulnerable (OK), no session ticket
extension
  Opossum (CVE-2025-49812)                 ./testssl.sh: line 1976: read: read error: 0:
Connection reset by peer
  not vulnerable (OK)
  ROBOT                                    Server does not support any cipher suites that
use RSA key transport
```

```
  Secure Renegotiation (RFC 5746)           supported (OK)
  Secure Client-Initiated Renegotiation     not vulnerable (OK)
  CRIME, TLS (CVE-2012-4929)                not vulnerable (OK)
  BREACH (CVE-2013-3587)                    no gzip/deflate/compress/br HTTP compression
(OK)  - only supplied "/" tested
  POODLE, SSL (CVE-2014-3566)               not vulnerable (OK), no SSLv3 support
  TLS_FALLBACK_SCSV (RFC 7507)              No fallback possible (OK), no protocol below
TLS 1.2 offered
  SWEET32 (CVE-2016-2183, CVE-2016-6329)    not vulnerable (OK)
  FREAK (CVE-2015-0204)                     not vulnerable (OK)
  DROWN (CVE-2016-0800, CVE-2016-0703)      not vulnerable on this host and port (OK)
                                            make sure you don't use this certificate
elsewhere with SSLv2 enabled services, see
                                            https://search.censys.io/search?resource=
hosts&virtual_hosts=INCLUDE&q=478AF59EAF01A4A237EE8571E8F1889D4991EB29675460E94ED4932F1D3B
0978
  LOGJAM (CVE-2015-4000), experimental      not vulnerable (OK): no DH EXPORT ciphers, no
DH key detected with <= TLS 1.2
  BEAST (CVE-2011-3389)                     not vulnerable (OK), no SSL3 or TLS1
  LUCKY13 (CVE-2013-0169), experimental     potentially VULNERABLE, uses cipher block
chaining (CBC) ciphers with TLS. Check patches
  Winshock (CVE-2014-6321), experimental    not vulnerable (OK)
  RC4 (CVE-2013-2566, CVE-2015-2808)        no RC4 ciphers detected (OK)


  Running client simulations (HTTP) via sockets

  Browser                      Protocol  Cipher Suite Name (OpenSSL)       Forward Secrecy
--------------------------------------------------------------------------------------------
  Android 7.0 (native)         TLSv1.2   ECDHE-RSA-AES128-GCM-SHA256       256 bit ECDH
(P-256)
  Android 8.1 (native)         TLSv1.2   ECDHE-RSA-AES128-GCM-SHA256       253 bit ECDH
(X25519)
  Android 9.0 (native)         TLSv1.3   TLS_AES_128_GCM_SHA256            253 bit ECDH
(X25519)
  Android 10.0 (native)        TLSv1.3   TLS_AES_128_GCM_SHA256            253 bit ECDH
(X25519)
  Android 11/12 (native)       TLSv1.3   TLS_AES_128_GCM_SHA256            253 bit ECDH
(X25519)
  Android 13/14 (native)       TLSv1.3   TLS_AES_128_GCM_SHA256            253 bit ECDH
(X25519)
  Android 15 (native)          TLSv1.3   TLS_AES_128_GCM_SHA256            X25519MLKEM768
  Chrome 101 (Win 10)          TLSv1.3   TLS_AES_128_GCM_SHA256            253 bit ECDH
(X25519)
  Chromium 137 (Win 11)        TLSv1.3   TLS_AES_128_GCM_SHA256            X25519MLKEM768
  Firefox 100 (Win 10)         TLSv1.3   TLS_AES_128_GCM_SHA256            253 bit ECDH
(X25519)
  Firefox 137 (Win 11)         TLSv1.3   TLS_AES_128_GCM_SHA256            X25519MLKEM768
  IE 8 Win 7                   No connection
  IE 11 Win 7                  TLSv1.2   ECDHE-RSA-AES256-SHA384           256 bit ECDH
(P-256)
  IE 11 Win 8.1                TLSv1.2   ECDHE-RSA-AES256-SHA384           256 bit ECDH
(P-256)
  IE 11 Win Phone 8.1          TLSv1.2   ECDHE-RSA-AES128-SHA256           256 bit ECDH
(P-256)
  IE 11 Win 10                 TLSv1.2   ECDHE-RSA-AES256-GCM-SHA384       256 bit ECDH
(P-256)
  Edge 15 Win 10               TLSv1.2   ECDHE-RSA-AES256-GCM-SHA384       253 bit ECDH
```

```
(X25519)
  Edge 101 Win 10 21H2         TLSv1.3   TLS_AES_128_GCM_SHA256           253 bit ECDH
(X25519)
  Edge 133 Win 11 23H2         TLSv1.3   TLS_AES_128_GCM_SHA256           X25519MLKEM768
  Safari 18.4 (iOS 18.4)       TLSv1.3   TLS_AES_128_GCM_SHA256           253 bit ECDH
(X25519)
  Safari 15.4 (macOS 12.3.1)   TLSv1.3   TLS_AES_128_GCM_SHA256           253 bit ECDH
(X25519)
  Safari 18.4 (macOS 15.4)     TLSv1.3   TLS_AES_128_GCM_SHA256           253 bit ECDH
(X25519)
  Java 7u25                    No connection
  Java 8u442 (OpenJDK)         TLSv1.3   TLS_AES_256_GCM_SHA384           253 bit ECDH
(X25519)
  Java 11.0.2 (OpenJDK)        TLSv1.3   TLS_AES_128_GCM_SHA256           256 bit ECDH
(P-256)
  Java 17.0.3 (OpenJDK)        TLSv1.3   TLS_AES_256_GCM_SHA384           253 bit ECDH
(X25519)
  Java 21.0.6 (OpenJDK)        TLSv1.3   TLS_AES_256_GCM_SHA384           253 bit ECDH
(X25519)
  go 1.17.8                    TLSv1.3   TLS_AES_128_GCM_SHA256           253 bit ECDH
(X25519)
  LibreSSL 3.3.6 (macOS)       TLSv1.3   TLS_CHACHA20_POLY1305_SHA256     253 bit ECDH
(X25519)
  OpenSSL 1.0.2e               TLSv1.2   ECDHE-RSA-AES256-GCM-SHA384      256 bit ECDH
(P-256)
  OpenSSL 1.1.1d (Debian)      TLSv1.3   TLS_AES_256_GCM_SHA384           253 bit ECDH
(X25519)
  OpenSSL 3.0.15 (Debian)      TLSv1.3   TLS_AES_256_GCM_SHA384           253 bit ECDH
(X25519)
  OpenSSL 3.5.0 (git)          TLSv1.3   TLS_AES_256_GCM_SHA384           X25519MLKEM768
  Apple Mail (16.0)            TLSv1.2   ECDHE-RSA-AES256-GCM-SHA384      256 bit ECDH
(P-256)
  Thunderbird (91.9)           TLSv1.3   TLS_AES_128_GCM_SHA256           253 bit ECDH
(X25519)


  Rating (experimental)

  Rating specs (not complete)  SSL Labs's 'SSL Server Rating Guide' (version 2009r from
2025-05-16)
  Specification documentation  https://github.com/ssllabs/research/wiki/SSL-Server-Rating-
Guide
  Protocol Support (weighted)  100 (30)
  Key Exchange     (weighted)  100 (30)
  Cipher Strength  (weighted)  90 (36)
  Final Score                  96
  Overall Grade                A+

  Done 2025-12-15 23:37:41 [  85s] -->> 127.0.0.1:443 (localhost) <<--


----------------------------------------------------------------------------------------------------
  Done testing now all IP addresses (on port 443): 127.0.0.1
```

Listing 22: Output of testssl.sh against the configured HTTPS server.

4. Appendix 2

```
$ ./testssl.sh --full --add-ca tls/nginx_test.crt localhost:443

  ###########################################################################
    testssl.sh version 3.3dev from https://testssl.sh/dev/
    (1250d6f8 2025-11-29 22:38:18)

    This program is free software. Distribution and modification under
    GPLv2 permitted. USAGE w/o ANY WARRANTY. USE IT AT YOUR OWN RISK!

    Please file bugs @ https://testssl.sh/bugs/
  ###########################################################################

    Using OpenSSL 1.0.2-bad (Mar 28 2025)  [~179 ciphers]
    on framework:./bin/openssl.Linux.x86_64

  Testing all IPv4 addresses (port 443): 127.0.0.1
------------------------------------------------------------------------------------------------------------
  Start 2025-12-15 23:52:48        -->> 127.0.0.1:443 (localhost) <<--

  A record via:           /etc/hosts
  rDNS (127.0.0.1):       localhost.
  Service detected:       HTTP

  Testing protocols via sockets except NPN+ALPN

  SSLv2      not offered (OK)
  SSLv3      not offered (OK)
  TLS 1      not offered
  TLS 1.1    not offered
  TLS 1.2    offered (OK)
  TLS 1.3    offered (OK): final
  QUIC       Local problem: No OpenSSL QUIC support
  NPN/SPDY   not offered
  ALPN/HTTP2 http/1.1 (offered)

  Testing for server implementation bugs

  No bugs found.

  Testing cipher categories

  NULL ciphers (no encryption)                      not offered (OK)
  Anonymous NULL Ciphers (no authentication)        not offered (OK)
  Export ciphers (w/o ADH+NULL)                     not offered (OK)
  LOW: 64 Bit + DES, RC[2,4], MD5 (w/o export)      not offered (OK)
  Triple DES Ciphers / IDEA                         not offered
  Obsoleted CBC ciphers (AES, ARIA etc.)            not offered
  Strong encryption (AEAD ciphers) with no FS       not offered
  Forward Secrecy strong encryption (AEAD ciphers)  offered (OK)


  Testing server's cipher preferences

  Hexcode  Cipher Suite Name (OpenSSL)       KeyExch.   Encryption  Bits     Cipher Suite
Name (IANA/RFC)
```

```
--------------------------------------------------------------------------------------------------------
  SSLv2
  -
  SSLv3
  -
  TLSv1
  -
  TLSv1.1
  -
  TLSv1.2 (server order)
  xc030   ECDHE-RSA-AES256-GCM-SHA384       ECDH 384   AESGCM      256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
  TLSv1.3 (server order)
  x1302   TLS_AES_256_GCM_SHA384            ECDH 384   AESGCM      256
TLS_AES_256_GCM_SHA384
  x1303   TLS_CHACHA20_POLY1305_SHA256      ECDH 384   ChaCha20    256
TLS_CHACHA20_POLY1305_SHA256

  Has server cipher order?     yes (OK) -- TLS 1.3 and below


  Testing robust forward secrecy (FS) -- omitting Null Authentication/Encryption, 3DES,
RC4

  FS is offered (OK)           TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 ECDHE-
RSA-AES256-GCM-SHA384
  Elliptic curves offered:     secp384r1
  TLS 1.2 sig_algs offered:    RSA-PSS-RSAE+SHA256 RSA-PSS-RSAE+SHA384 RSA-PSS-RSAE+SHA512
RSA+SHA256 RSA+SHA384 RSA+SHA512 RSA+SHA224
  TLS 1.3 sig_algs offered:    RSA-PSS-RSAE+SHA256 RSA-PSS-RSAE+SHA384 RSA-PSS-RSAE+SHA512

  Testing server defaults (Server Hello)

  TLS extensions (standard)    "server name/#0" "max fragment length/#1" "EC point
formats/#11" "application layer protocol negotiation/#16" "extended master secret/#23"
                               "supported versions/#43" "key share/#51" "renegotiation
info/#65281"
  Session Ticket RFC 5077 hint no -- no lifetime advertised
  SSL Session ID support       yes
  Session Resumption           tickets no, ID: yes
  TLS 1.3 early data support   no early data offered
  TLS clock skew               Random values, no fingerprinting possible
  Certificate Compression      none
  Client Authentication        none
  Signature Algorithm          SHA256 with RSA
  Server key size              RSA 4096 bits (exponent is 65537)
  Server key usage             --
  Server extended key usage    --
  Serial                       2CAD8D45D7ACEF3F00339C5BB56A42AE404F23B7 (OK: length 20)
  Fingerprints                 SHA1 04CD6B99F36DAC7F4DCCDB1B6C258EB2D53EC793
                               SHA256
478AF59EAF01A4A237EE8571E8F1889D4991EB29675460E94ED4932F1D3B0978
  Common Name (CN)             localhost
  subjectAltName (SAN)         missing (NOT ok) -- Browsers are complaining
  Trust (hostname)             via CN only -- Browsers are complaining (same w/o SNI)
  Chain of trust               Ok
  EV cert (experimental)       no
  Certificate Validity (UTC)   364 >= 60 days (2025-12-15 22:07 --> 2026-12-15 22:07)
  ETS/"eTLS", visibility info  not present
```

```
  Certificate Revocation List  --
  OCSP URI                     --
                                NOT ok -- neither CRL nor OCSP URI provided
  OCSP stapling                not offered
  OCSP must staple extension   --
  DNS CAA RR (experimental)    not offered
  Certificate Transparency     --
  Certificates provided        1
  Issuer                       localhost (HCW from AU)
  Intermediate Bad OCSP (exp.) Ok


  Testing HTTP header response @ "/"

  HTTP Status Code             200 OK
  HTTP clock skew              0 sec from localtime
  Strict Transport Security    365 days=31536000 s, includeSubDomains, preload
  Public Key Pinning           --
  Server banner                nginx/1.29.4
  Application banner           --
  Cookie(s)                    (none issued at "/")
  Security headers             --
  Reverse Proxy banner         --


  Testing vulnerabilities

  Heartbleed (CVE-2014-0160)                not vulnerable (OK), no heartbeat extension
  CCS (CVE-2014-0224)                       not vulnerable (OK)
  Ticketbleed (CVE-2016-9244), experiment.  not vulnerable (OK), no session ticket
extension
  Opossum (CVE-2025-49812)                  not vulnerable (OK)
  ROBOT                                     Server does not support any cipher suites that
use RSA key transport
  Secure Renegotiation (RFC 5746)           supported (OK)
  Secure Client-Initiated Renegotiation     not vulnerable (OK)
  CRIME, TLS (CVE-2012-4929)                not vulnerable (OK)
  BREACH (CVE-2013-3587)                    no gzip/deflate/compress/br HTTP compression
(OK)  - only supplied "/" tested
  POODLE, SSL (CVE-2014-3566)               not vulnerable (OK), no SSLv3 support
  TLS_FALLBACK_SCSV (RFC 7507)              No fallback possible (OK), no protocol below
TLS 1.2 offered
  SWEET32 (CVE-2016-2183, CVE-2016-6329)    not vulnerable (OK)
  FREAK (CVE-2015-0204)                     not vulnerable (OK)
  DROWN (CVE-2016-0800, CVE-2016-0703)      not vulnerable on this host and port (OK)
                                            make sure you don't use this certificate
elsewhere with SSLv2 enabled services, see
                                            https://search.censys.io/search?resource=
hosts&virtual_hosts=INCLUDE&q=478AF59EAF01A4A237EE8571E8F1889D4991EB29675460E94ED4932F1D3B
0978
  LOGJAM (CVE-2015-4000), experimental      not vulnerable (OK): no DH EXPORT ciphers, no
DH key detected with <= TLS 1.2
  BEAST (CVE-2011-3389)                     not vulnerable (OK), no SSL3 or TLS1
  LUCKY13 (CVE-2013-0169), experimental     not vulnerable (OK)
  Winshock (CVE-2014-6321), experimental    not vulnerable (OK)
  RC4 (CVE-2013-2566, CVE-2015-2808)        no RC4 ciphers detected (OK)


  Running client simulations (HTTP) via sockets
```

Browser	Protocol	Cipher Suite Name (OpenSSL)	Forward Secrecy
Android 7.0 (native)	No connection		
Android 8.1 (native)	TLSv1.2	ECDHE-RSA-AES256-GCM-SHA384	384 bit ECDH (P-384)
Android 9.0 (native)	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
Android 10.0 (native)	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
Android 11/12 (native)	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
Android 13/14 (native)	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
Android 15 (native)	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
Chrome 101 (Win 10)	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
Chromium 137 (Win 11)	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
Firefox 100 (Win 10)	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
Firefox 137 (Win 11)	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
IE 8 Win 7	No connection		
IE 11 Win 7	No connection		
IE 11 Win 8.1	No connection		
IE 11 Win Phone 8.1	No connection		
IE 11 Win 10	TLSv1.2	ECDHE-RSA-AES256-GCM-SHA384	384 bit ECDH (P-384)
Edge 15 Win 10	TLSv1.2	ECDHE-RSA-AES256-GCM-SHA384	384 bit ECDH (P-384)
Edge 101 Win 10 21H2	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
Edge 133 Win 11 23H2	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
Safari 18.4 (iOS 18.4)	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
Safari 15.4 (macOS 12.3.1)	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
Safari 18.4 (macOS 15.4)	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
Java 7u25	No connection		
Java 8u442 (OpenJDK)	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
Java 11.0.2 (OpenJDK)	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
Java 17.0.3 (OpenJDK)	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
Java 21.0.6 (OpenJDK)	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
go 1.17.8	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
LibreSSL 3.3.6 (macOS)	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH (P-384)
OpenSSL 1.0.2e	TLSv1.2	ECDHE-RSA-AES256-GCM-SHA384	384 bit ECDH (P-384)
OpenSSL 1.1.1d (Debian)	TLSv1.3	TLS_AES_256_GCM_SHA384	384 bit ECDH

```
(P-384)
  OpenSSL 3.0.15 (Debian)      TLSv1.3   TLS_AES_256_GCM_SHA384            384 bit ECDH
(P-384)
  OpenSSL 3.5.0 (git)          TLSv1.3   TLS_AES_256_GCM_SHA384            384 bit ECDH
(P-384)
  Apple Mail (16.0)            TLSv1.2   ECDHE-RSA-AES256-GCM-SHA384       384 bit ECDH
(P-384)
  Thunderbird (91.9)           TLSv1.3   TLS_AES_256_GCM_SHA384            384 bit ECDH
(P-384)


  Rating (experimental)

  Rating specs (not complete)  SSL Labs's 'SSL Server Rating Guide' (version 2009r from
2025-05-16)
  Specification documentation  https://github.com/ssllabs/research/wiki/SSL-Server-Rating-
Guide
  Protocol Support (weighted)  100 (30)
  Key Exchange     (weighted)  100 (30)
  Cipher Strength  (weighted)  100 (40)
  Final Score                  100
  Overall Grade                A+

  Done 2025-12-15 23:54:18 [  94s] -->> 127.0.0.1:443 (localhost) <<--


--------------------------------------------------------------------------------------------------------
  Done testing now all IP addresses (on port 443): 127.0.0.1
```

Listing 23: Output of testssl.sh against the configured HTTPS server (hardened).