Bezpieczeństwo Aplikacji WEB

Zadanie 1

Prowadzący:
Mgr inż. Przemysław Świercz

wtorek 8:15
Miłosz Jagodziński CBE

Apache

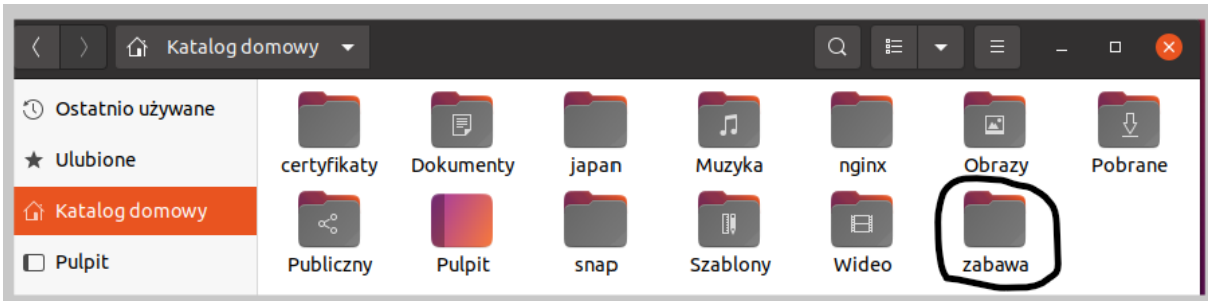
Zadanie
Bezpieczna konfiguracja serwera webowego
Proszę przygotować konfigurację dla serwera Apache ORAZ Nginx spełniające następujące kryteria:

Zadanie 1:
Proszę wygenerować self-signed certyfikat oraz skonfigurować obsługę HTTPS
Ścieżka /http-only (wraz ze wszystkimi podścieżkami) ma działać wyłącznie w trybie HTTP (nieszyfrowanym)Ścieżka /http-https (wraz ze wszystkimi podścieżkami) ma działać w oby trybach, HTTP i HTTPS Wszystkie pozostałe ścieżki mają mieć automatyczne przekierowanie na tryb HTTPS, czyli np. kiedy przyjdzie zapytanie po HTTP dla /inna-sciezka ma zostać wykonane przekierowanie na wersję HTTPS dla tej samej ścieżki

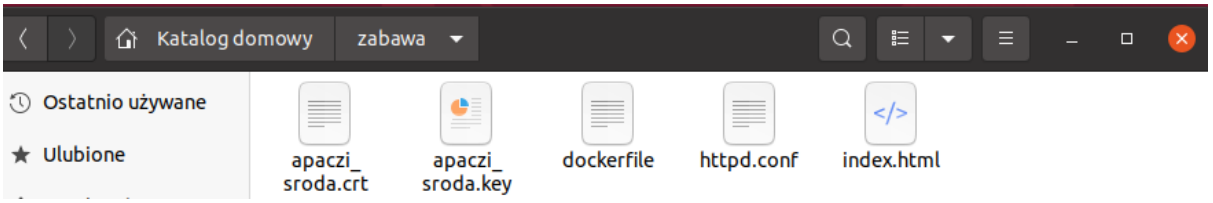
Środowisko pracy:
Ubuntu 20.04 Virtualbox

Utworzyłem folder roboczy, aby przechowywać potrzebne pliki. Uwaga nie można przetrzymywać wszystkich plików w jednym miejscu w tym certyfikatów, ale jest o wersja robocza i do nauki, dlatego trzymano te rzeczy w jednym miejscu.

cd mkdir zabawa



W folderze tym zamieszczono potrzebne pliki. Potem dodano kolejne pliki.



Na mojej wirtualnej maszynie zainstalowałem Dokcera, aby móc wykonać zadania wygenerowania certyfikatów SSL i stworzenia przekierowań.

Certyfikaty wygenerowano za pomocą polecenia:
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /home/milosz/certyfikaty/apache-selfsigned.key -out /home/milosz/certyfikaty/apache-selfsigned.crt

Potem zrobiono Dockerfile, aby można było go wgrać przy budowaniu obrazu i kontenera
Zawartość Dockerfile

Wygląd dockerfile na tym etapie

```
FROM httpd:2.4
        #pobranie wezwanie apacza, który będzie naszą bazą do dalszych działań
LABEL maintainer = "242027"
        # nazwa wlasciciela
COPY ./index.html /usr/local/apache2/htdocs/
        #ścieżka do mojej stronki
COPY ./apaczi_sroda.key /usr/local/apache2/conf/server.key
COPY ./apaczi_sroda.crt /usr/local/apache2/conf/server.crt
        #wgranie certyfikatów
```

Uwaga certyfikaty muszą się nazywać server.crt i server.key w innym przypadku nie działało.

Następnie utworzono obraz w Docker wykorzystując polecenie:
sudo docker build apaczik .

```
milosz@milosz-VirtualBox:~/zabawa$ sudo docker build -t apaczii .
Sending build context to Docker daemon   29.7kB
Step 1/5 : FROM httpd:2.4
 ---> c30a46771695
Step 2/5 : LABEL maintainer = "242027"
 ---> Using cache
 ---> 6b4c70d2b907
Step 3/5 : COPY ./index.html /usr/local/apache2/htdocs/
 ---> Using cache
 ---> 4d2e5227aff9
Step 4/5 : COPY ./apaczi_sroda.key /usr/local/apache2/conf/server.key
 ---> 203062ffde5c
Step 5/5 : COPY ./apaczi_sroda.crt /usr/local/apache2/conf/server.crt
 ---> 4b63d1b9d197
Successfully built 4b63d1b9d197
Successfully tagged apaczii:latest
```

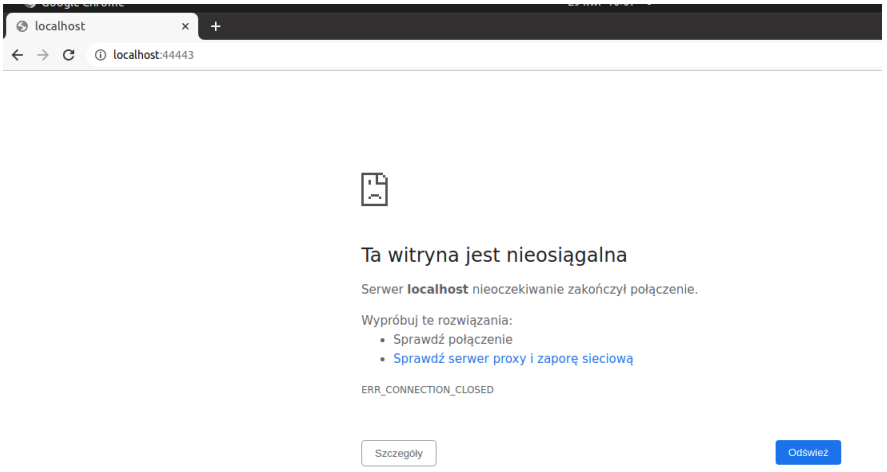
Kolejno utworzono kontener
sudo docker run -dit --name xd1 -p 8080:80 -p 44443:443 sroda

```
milosz@milosz-VirtualBox:~/zabawa$ sudo docker run -dit --name apaczik -p 8080:80 -p 44443:443 apaczii
acc5fd1f862dc670a8fb26e211e9464f65d7c5abc045e4c83caa57e63cb7f3bd
milosz@milosz-VirtualBox:~/zabawa$
```

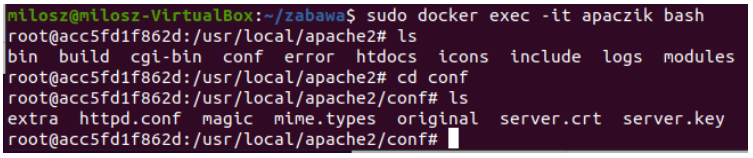
```
milosz@milosz-VirtualBox:~/zabawa$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED              STATUS              PORTS
                  NAMES
acc5fd1f862d   apaczii   "httpd-foreground"   About a minute ago   Up About a minute   0.0.0.0:8080->80/tcp, :::8080->80/tcp, 0.0.0.0:44443->443/tcp,
 :::44443->443/tcp   apaczik
```

Jak widać poniżej strona jest dostępna lokalnie przez port 8080.

Jednak przez https nie można wejść.



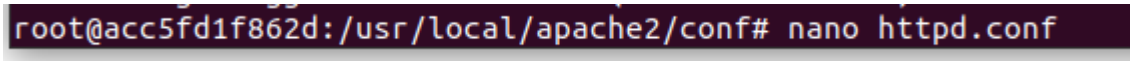
Kolejny krok to wejście do kontenera, aby zmodyfikować plik httpd.conf. Również widać tu wgrane certyfikaty.
sudo docker exec -it nazwa bash



Zdekomentowano następujące linijki tego plik tak jak jest w instrukcji httpd na oficjalnej stronie Dockera.

...
#LoadModule socache_shmcb_module modules/mod_socache_shmcb.so
...
#LoadModule ssl_module modules/mod_ssl.so
...
#Include conf/extra/httpd-ssl.conf

Jednak wpierw trzeba było apt-get update i apt-get install nano.



nano httpd.conf

Zdekomentowanie:
1

```
#LoadModule cache_socache_module modules/mod_cache_socache.so
LoadModule socache_shmcb_module modules/mod_socache_shmcb.so
#LoadModule socache_dbm_module modules/mod_socache_dbm.so
```

2

```
LoadModule ssl_module modules/mod_ssl.so
```

3

```
# Secure (SSL/TLS) connections
Include conf/extra/httpd-ssl.conf
#
```

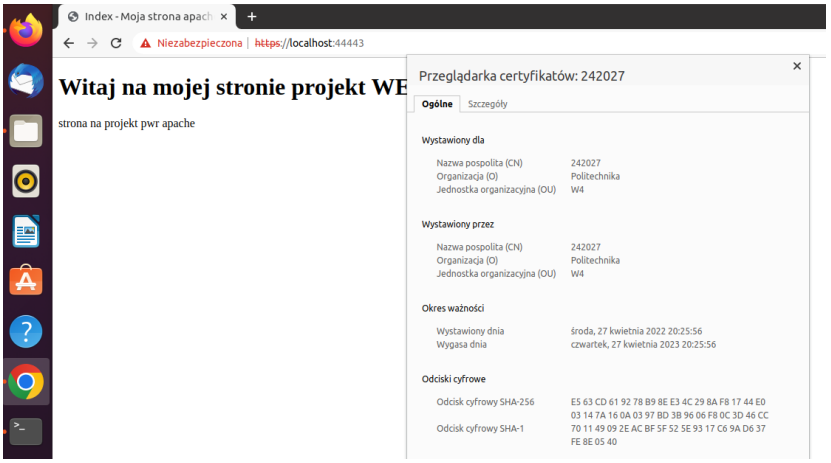
Przy okazji zmieniono ServerName na localhost.

```
#
ServerName localhost:80
```

Skonfigurowany plik zapisano i zrestartowano Apache za pomocą komendy na dole.
apachectl restart

Możemy wejść od tego momentu po https, ale certyfikat nie jest zgłoszony, dlatego nie ma zielonej kłódki.

Poniżej widać, że nasz certyfikat jest nieważny, ale istnieje.



Przekierowywanie
Po wgraniu certyfikatów ssl przystąpiono do dalszej części zadania. Utworzono w folderze htdocs foldery http-only i http-https.

```
root@84f0ba02c15c:/usr/local/apache2/htdocs/http-https# nano index1.html
root@84f0ba02c15c:/usr/local/apache2/htdocs/http-https# nano index1.html
root@84f0ba02c15c:/usr/local/apache2/htdocs/http-https# cd ..
root@84f0ba02c15c:/usr/local/apache2/htdocs# ls
http-https  http-only  index.html
root@84f0ba02c15c:/usr/local/apache2/htdocs# cd http-only
root@84f0ba02c15c:/usr/local/apache2/htdocs/http-only# ls
root@84f0ba02c15c:/usr/local/apache2/htdocs/http-only# nano index2.html
root@84f0ba02c15c:/usr/local/apache2/htdocs/http-only# ls
index2.html
```

Kolejnym krokiem była znowu edycja w pliku httpd.conf. Odekomentowano tą frazę LoadModule rewrite_module modules/mod_rewrite.so .

Następnie wprowadzono poniższe wiersze:
<VirtualHost *:80>

```
  #ServerName localhost:8082
  #ServerAlias localhost:8082*
  #Redirect permanent / https://localhost:44445
  RewriteEngine On
  RewriteCond %{HTTPS} off
  RewriteRule ^(?!.*http-only|http-https/).* https://localhost:44445%{REQUEST_URI}
  #RewriteRule ^(?!.*/http-https/).* https://localhost:44445%{REQUEST_URI}
```

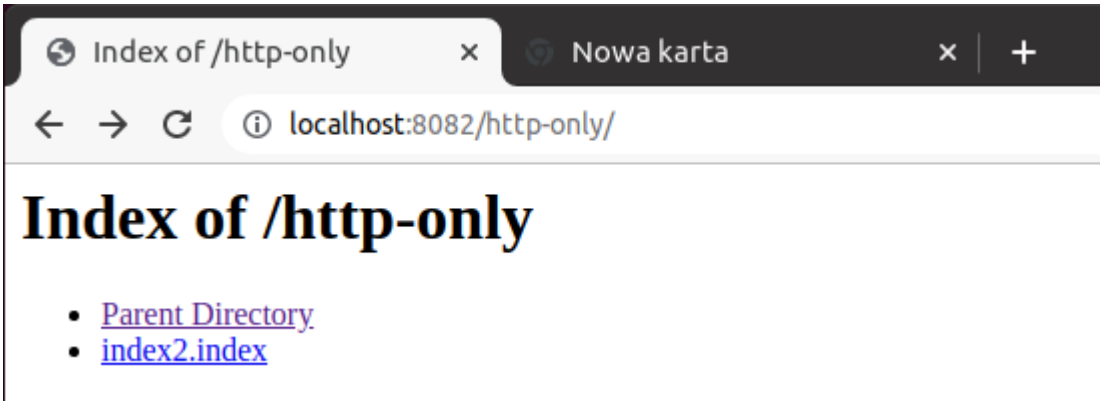
</VirtualHost>

Ostatnią rzeczą było dodanie w pliku /usr/local/apache/conf/extra/httpd-ssl.conf:

```
RewriteEngine On
RewriteCond %{HTTPS} on
RewriteRule (.*http-only*) http://localhost:8080%{REQUEST_URI}
```

Trzeba było zrestartować Apache:
apachectl restart

http-only



http-https

Podsumowanie

Konfigurowanie Apache i zrobienie tych zadań było bardzo wymagające. Nauczyłem się generować certyfikaty ssl i zapoznałem się z plikami konfiguracyjnymi Apache. Trzeba było wykorzystywać różne parametry.

Nginx

Zadanie
Bezpieczna konfiguracja serwera webowego
Proszę przygotować konfigurację dla serwera Apache ORAZ Nginx spełniające następujące kryteria:

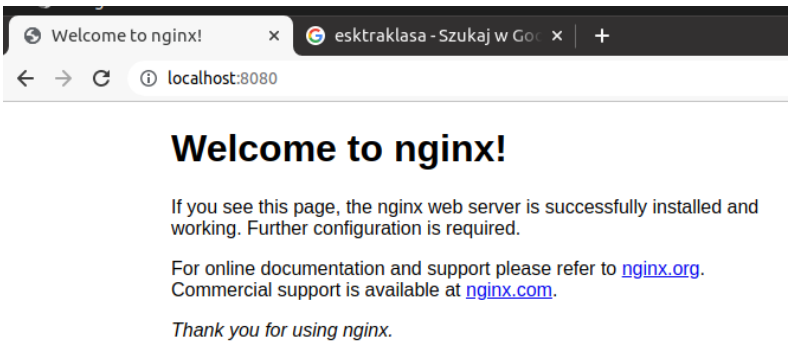
Zadanie 1:
Proszę wygenerować self-signed certyfikat oraz skonfigurować obsługę HTTPS
Ścieżka /http-only (wraz ze wszystkimi podścieżkami) ma działać wyłącznie w trybie HTTP (nieszyfrowanym)Ścieżka /http-https (wraz ze wszystkimi podścieżkami) ma działać w oby trybach, HTTP i HTTPS Wszystkie pozostałe ścieżki mają mieć automatyczne przekierowanie na tryb HTTPS, czyli np. kiedy przyjdzie zapytanie po HTTP dla /inna-sciezka ma zostać wykonane przekierowanie na wersję HTTPS dla tej samej ścieżki.

Środowisko pracy:
Ubuntu 20.04 Virtualbox

W tym zadaniu wykorzystałem Alpine, aby na nim postawić Nginx. Alpine jest bardzo lekką dystrybucją Linuxa, aby wykorzystywał tylko najpotrzebniejsze zasoby.

```
milosz@milosz-VirtualBox:~$ sudo docker run -it -p 8080:80 -p 4444:443 --name nginx-alpy
alpine /bin/sh;
/ #
```

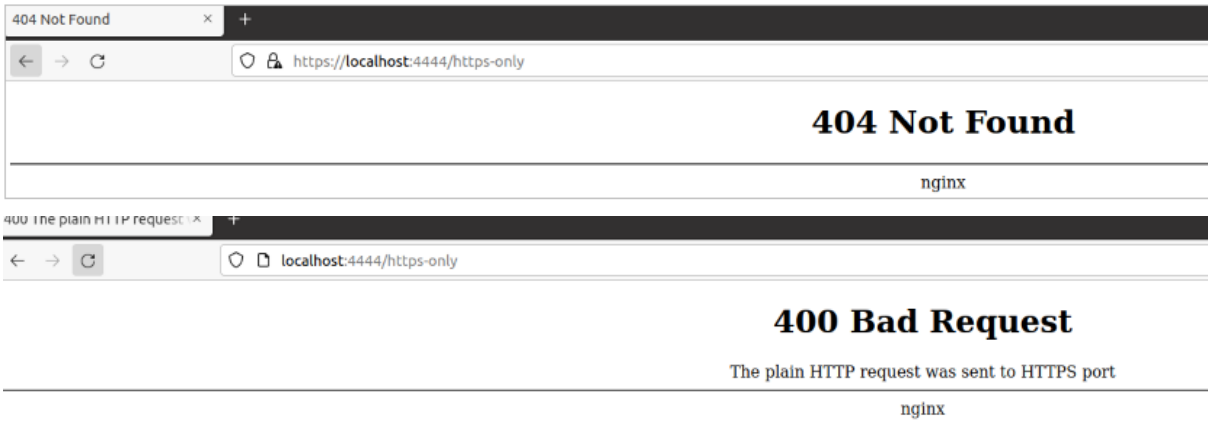
Poniżej widać jak działa stronka na ten moment lokalnie i http.



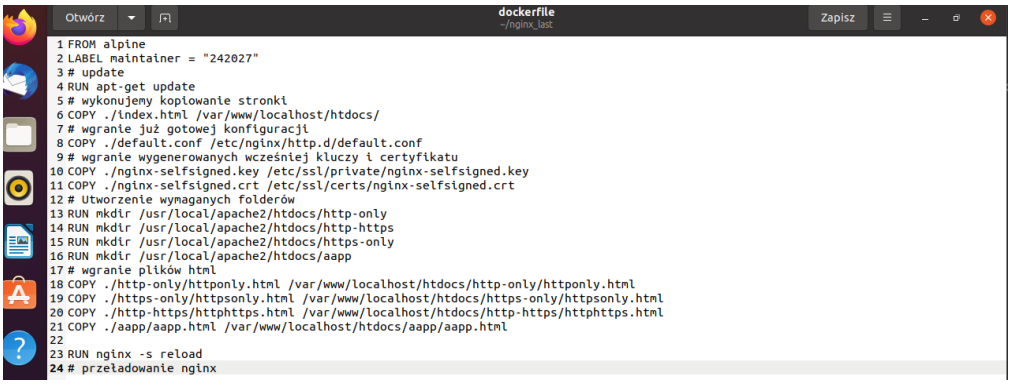
Kolejno zmodyfikowano plik default.conf. Jak widać poniżej.

```
server {

        listen 80 default_server;
        listen [::]:80 default_server;
           rewrite ^(?!.*http-only|http-https/).* https://localhost$request_uri;
        # New root location
        location / {
                root /var/www/localhost/htdocs;
                # return 404;
        }
        # You may need this to prevent return 404 recursion.
        location = /404.html {
                internal;
           }
}

server {
        listen 443 ssl http2 default_server;
        listen [::]:443 ssl http2 default_server;
        ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;

        ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;
           rewrite .*http-only* http://localhost$request_uri;
        # New root location
        location / {
                root /var/www/localhost/htdocs;
                # return 404;
        }
        # You may need this to prevent return 404 recursion.
        location = /404.html {
                internal;
        }
}
```

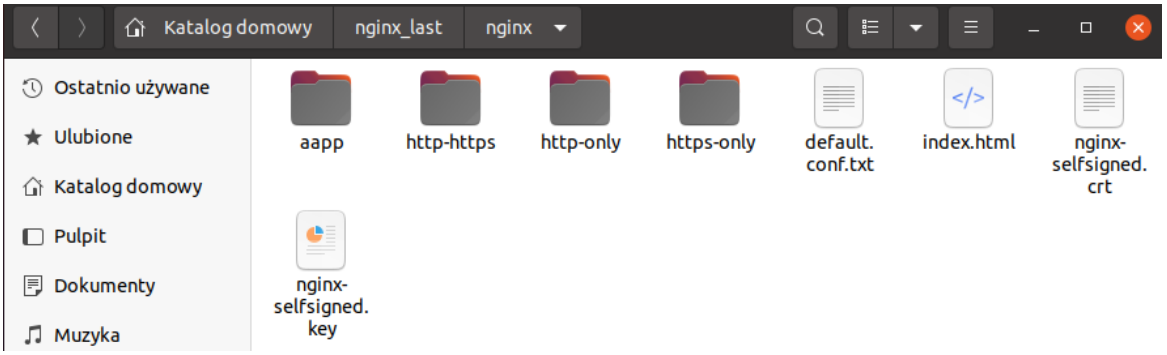
Na dole widać np. jak działa https-only.



Końcowy dockerfile wygląda następująco:



Natomiast folder końcowy zawiera aż 9 plików.



Podsumowanie

Przy konfigurowaniu nginx trzeba było postawić Alpine, ponieważ jest to jedno z rozwiązań. Można konfigurować od razu z FROM nginx, ale przy pierwszych próbach nie wychodziło. Tak samo wygenerowano certyfikaty jak przy Apache. Konfiguracja przekierowania jednak trochę się różni, ponieważ są to inne rzeczy, a więc trzeba było konfigurować inne pliki konfiguracyjne i używać innych parametrów.