Bezpieczeństwo Aplikacji WEB

Zadanie 3

Prowadzący:
Mgr inż. Przemysław Świercz

wtorek 8:15
Miłosz Jagodziński CBE

Apache

Zadanie

Cel:

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

Zadanie 3: (kontynuacja zadania 1 i 2)

Proszę przygotować rozwiązanie zapewniające limitowanie liczby zapytań (rate limitting), tak aby Dla wybranej ścieżki można było ustawić osobny globalny (czyli niezależny od tego kto wykonuje zapytanie) limit. Limit można było zdefiniować dla danej ścieżki dla wszystkich zapytań, albo osobny limit dla zapytań z wybranym nagłówkiem HTTP (np. user agent). Np ścieżka /limit-a pozwala na 100 zapytań na minutę globalnie, ale tylko 10 zapytań na minutę dla danego user-agenta.

Środowisko pracy:

Ubuntu 20.04 Virtualbox, Docker, Apache

Przebieg ćwiczenia Apache

Ostatnie zadanie z konfiguracji Apache, gdzie skonfigurowano limity zapytań globalne i dla użytkownika. Trzeba było wykorzystać dodatkowe moduły np. mod_evasive, mod_security lub mod_qos itp.. W tej konfiguracji wykorzystano mod_qos.

Opis Rate Limiting

Rate Limiting ma pomagać w ochronie przed atakami DDoS i zwiększyć bezpieczeństwo. Opcja ta ogranicza ilość żądań http wysyłanych przez klienta po drugiej stronie. Podsumowując rozwiązanie to ma chronić przed przeciążeniem serwera w przypadku dużej ilości żądań.

Opis modułu mod_qos

Moduł ten zapewnia ochronę dla serwera przed przeciążeniami w przypadku, gdy do serwera wysyłanych jest zbyt wiele żądań. Mod_qos umożliwia wprowadzenie wiele mechanizmów kontrolnych, aby określić ilość dozwolonych

żądań od klienta. Według twórców mod_qos oferuje następujące możliwości, ale w zadaniu wykorzystano tylko ułamek:”

- Maksymalna liczba jednoczesnych żądań do lokalizacji/zasobu (URL) lub hosta wirtualnego.
- Ograniczenie przepustowości , takie jak maksymalna dozwolona liczba żądań na sekundę do adresu URL lub maksymalna/minimalna liczba pobranych kilobajtów na sekundę.
- Ogranicza liczbę zdarzeń żądania na sekundę (specjalne warunki żądania).
- Ogranicza liczbę zdarzeń żądań w określonym przedziale czasu .
- Może również wykryć bardzo ważne osoby (VIP), które mogą uzyskać dostęp do serwera WWW bez ograniczeń lub z mniejszymi ograniczeniami.
- Ogólny filtr wiersza żądania i nagłówka w celu odrzucenia nieautoryzowanych operacji.
- Ograniczenie i filtrowanie danych treści żądania (wymaga mod_parp).
- Ogranicza liczbę zdarzeń żądań dla poszczególnych klientów (IP).
- Ograniczenia na poziomie połączenia TCP, np. maksymalna liczba dozwolonych połączeń z pojedynczego adresu źródłowego IP lub dynamiczna kontrola utrzymywania aktywności.
- Preferuje znane adresy IP, gdy serwerowi zabraknie wolnych połączeń TCP.
- Serializacja wniosków.”

Konfiguracja Dockerfile

Na samym początku skonfigurowano dockerfile, gdzie zainstalowano moduł i utworzono odpowiednie katalogi i podstrony. Trzeba pamiętać o parametrze -y, aby instalacja została zaakceptowana automatycznie.

`RUN apt-get install -y libapache2-mod-qos` - instalacja mod_qos

Następnie dodano katalogi i podstrony.

```
RUN mkdir /usr/local/apache2/htdocs/user-agent
RUN mkdir /usr/local/apache2/htdocs/limit-a
COPY ./user-agent.html /usr/local/apache2/htdocs/user-agent/user-agent.html
COPY ./limit-a.html /usr/local/apache2/htdocs/limit-a/limit-a.html
```

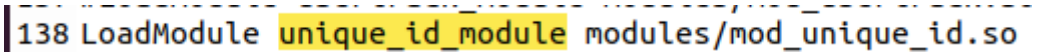
Dockerfile został skonfigurowany.

Działa dalej!!!!!!!!
Poniżej pokazano nieudane kroki.

Konfiguracja httpd.conf Uwaga to nie działało!!!!!!!!

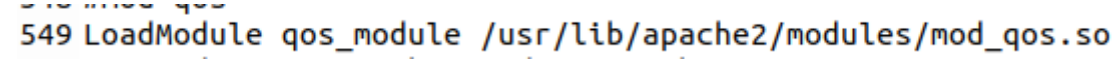
Potem do httpd.conf dodano nowe linijki, aby wprowadzić parametry mod_qos. Wpierw trzeba było odznaczyć moduł unique_id_module, ponieważ bez tego występował następujacy błąd:

Zdekomentowanie modułu id:




```
LoadModule unique_id_module modules/mod_unique_id.so
```

Po zdekomentowaniu unique_id_module trzeba było wskazać na moduł qos. Dlatego trzeba było ręcznie wpisać ścieżkę do niego, gdzie się wskazuje.



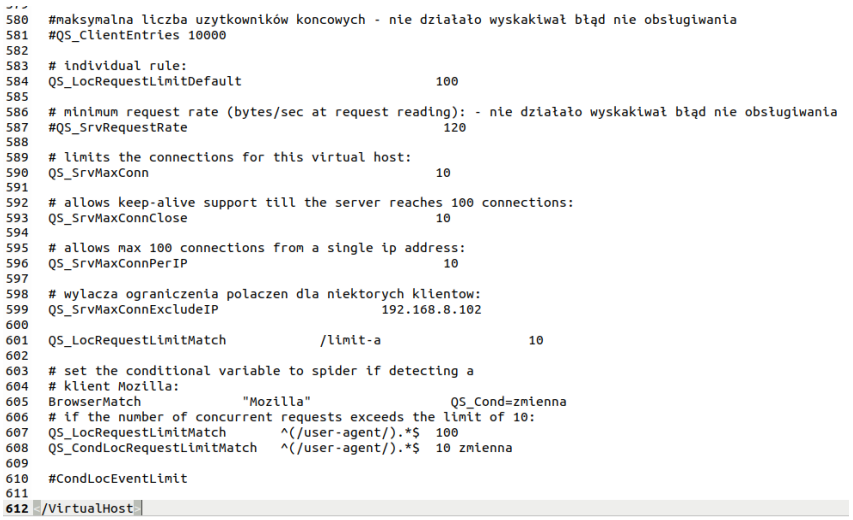

```
LoadModule qos_module /usr/lib/apache2/modules/mod_qos.so
```

W httpd.conf można parametrem include wskazać na ścieżkę pliku konfiguracyjnego qos, jednak skonfigurowano w virtualhost.

Następnie przed virtualhost dodano odpowiedź, która ma zostać pokazana dla klienta w przypadku przeciążenia.


```
QS_ErrorResponseCode 429
```

Dalej wprowadzono parametry mod_qos:



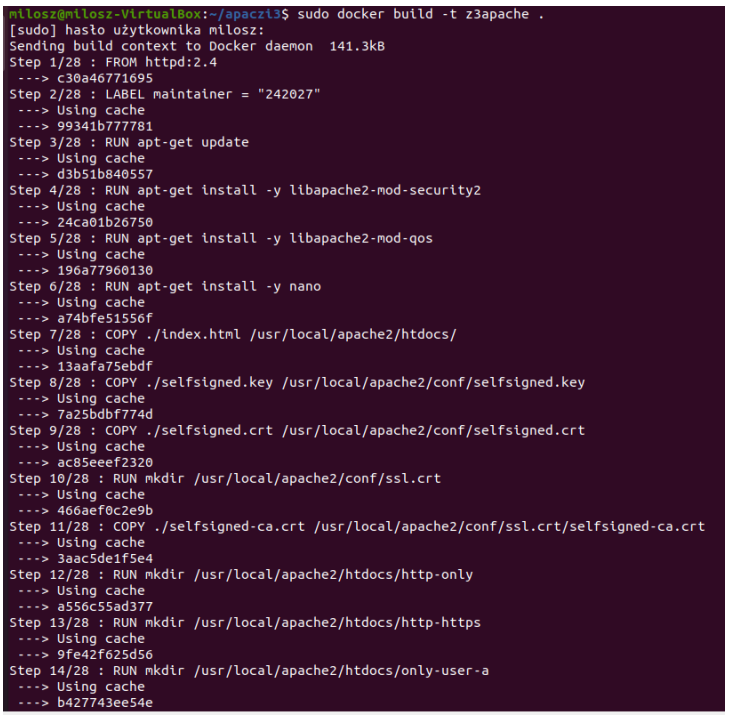
```
#maksymalna liczba uzytkowników koncowych - nie działało wyskakiwał błąd nie obsługiwania
#QS_ClientEntries 10000

# individual rule:
QS_LocRequestLimitDefault                       100

# minimum request rate (bytes/sec at request reading): - nie działało wyskakiwał błąd nie obsługiwania
#QS_SrvRequestRate                               120

# limits the connections for this virtual host:
QS_SrvMaxConn                                   10

# allows keep-alive support till the server reaches 100 connections:
QS_SrvMaxConnClose                              10

# allows max 100 connections from a single ip address:
QS_SrvMaxConnPerIP                              10

# wylacza ograniczenia polaczen dla niektorych klientow:
QS_SrvMaxConnExcludeIP                  192.168.8.102

QS_LocRequestLimitMatch           /limit-a                  10

# set the conditional variable to spider if detecting a
# klient Mozilla:
BrowserMatch            "Mozilla"                QS_Cond=zmienna
# if the number of concurrent requests exceeds the limit of 10:
QS_LocRequestLimitMatch       ^(/user-agent/).*$  100
QS_CondLocRequestLimitMatch   ^(/user-agent/).*$  10 zmienna

#CondLocEventLimit

</VirtualHost>
```

```
QS_RequestHeaderFilter on

#maksymalna liczba uzytkowników koncowych - nie działało wyskakiwał błąd nie obsługiwania
#QS_ClientEntries 10000 - ilość maksymalna użytkowników, ale w trakcie budowania obrazu wyskakiwał błąd, więc zdementowano

# individual rule: - domyślny limit dla klientów 100
QS_LocRequestLimitDefault                       100

# minimalna szybkość żądania (B/s przy odczycie żądania): - nie działało wyskakiwał błąd nie obsługiwania, mi
#QS_SrvRequestRate                               120

# limits the connections for this virtual host: - limit dla wirtualnych hostów
QS_SrvMaxConn                                   100

# allows keep-alive support till the server reaches 100 connections: - dla serwera
QS_SrvMaxConnClose                              100

# allows max 10 connections from a single ip address: - ilość żądań od pojedynczego adresu IP
QS_SrvMaxConnPerIP                              10

# wylacza ograniczenia połączeń dla niektórych klientów:- Adres IP gospodarza otrzymuje pozwolenie bez reguł
QS_SrvMaxConnExcludeIP                  192.168.8.102
```

```
  QS_LocRequestLimitMatch              /limit-a
10 - limit dla ścieżki ustawiony na 10

  # set the conditional variable to spider if detecting a
  # klient Mozilla: - poniżej widac jak wskazuję się usera i tworzy zmienną o
nazwie zmienna i wskazuje agenta
  BrowserMatch              "Mozilla"           QS_Cond=zmienna
  # if the number of concurrent requests exceeds the limit of
10: - globalnie 100, ale już dla user agenta 10.
  QS_LocRequestLimitMatch       ^(/user-agent/).*$  100
  QS_CondLocRequestLimitMatch   ^(/user-agent/).*$  10 zmienna
```

Następnie zbudowano obraz i kontener.

```
sudo docker build -t z3apache .
```



```
Step 15/28 : RUN mkdir /usr/local/apache2/htdocs/only-user-b
 ---> Using cache
 ---> 3dfe51b5af58
Step 16/28 : RUN mkdir /usr/local/apache2/htdocs/user-a-or-b
 ---> Using cache
 ---> 11be4b88c020
Step 17/28 : RUN mkdir /usr/local/apache2/htdocs/user-agent
 ---> Using cache
 ---> 4a5459013661
Step 18/28 : RUN mkdir /usr/local/apache2/htdocs/limit-a
 ---> Using cache
 ---> c58300c8989c
Step 19/28 : COPY ./indexhttp.html /usr/local/apache2/htdocs/http-only/index2.index
 ---> Using cache
 ---> f5ba572b8188
Step 20/28 : COPY ./indexhttps.html /usr/local/apache2/htdocs/http-https/index3.index
 ---> Using cache
 ---> 7d7aac6d920e
Step 21/28 : COPY ./indexklientA.html /usr/local/apache2/htdocs/only-user-a/indexklientA.html
 ---> Using cache
 ---> 95cd0c0a5396
Step 22/28 : COPY ./indexklientB.html /usr/local/apache2/htdocs/only-user-b/indexklientB.html
 ---> Using cache
 ---> 92bd291e24f2
Step 23/28 : COPY ./indexklientAiB.html /usr/local/apache2/htdocs/user-a-or-b/indexklientAiB.html
 ---> Using cache
 ---> 841d9bf2760f
Step 24/28 : COPY ./user-agent.html /usr/local/apache2/htdocs/user-agent/user-agent.html
 ---> Using cache
 ---> b9ba9d269459
Step 25/28 : COPY ./limit-a.html /usr/local/apache2/htdocs/limit-a/limit-a.html
 ---> Using cache
 ---> 8b9243341eaf
Step 26/28 : COPY ./httpd.conf /usr/local/apache2/conf/httpd.conf
 ---> 62691351c4f6
Step 27/28 : COPY ./httpd-ssl.conf /usr/local/apache2/conf/extra/httpd-ssl.conf
 ---> 222d2170fac2
Step 28/28 : RUN apachectl
 ---> Running in 7b2f40e3eec9
[Wed May 18 11:35:38.941219 2022] [ssl:warn] [pid 9:tid 140715430632768] AH01909: www.example.com:44
Removing intermediate container 7b2f40e3eec9
 ---> 338a00da0c2a
Successfully built 338a00da0c2a
Successfully tagged z3apache:latest
milosz@milosz-VirtualBox:~/apaczi3$
```

Występuje ostrzeżenie, ale wynika ono z powodu nie potwierdzenia certyfikatów ssl.

Potem odpalono kontener.

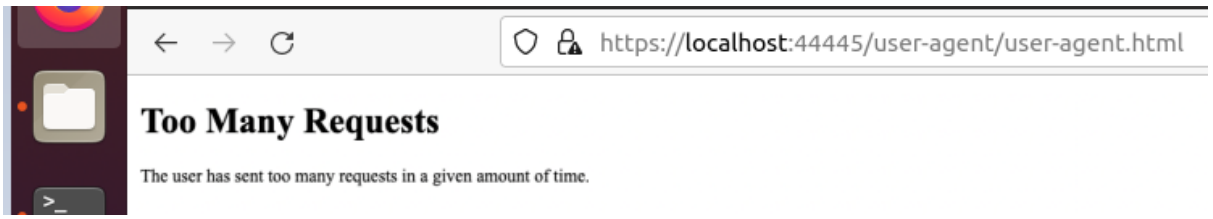
```
sudo docker run -dit --name kontener -p 8082:80 -p 44445:443 z3apache
```

```
milosz@milosz-VirtualBox:~/apaczi3$ sudo docker run -dit --name kontener -p 8082:80 -p 44445:443 z3apache
d0aa4ff159dfecdfccdc65412472d6853035600b9f1c55498224c20add2ddec1
milosz@milosz-VirtualBox:~/apaczi3$ sudo docker ps
CONTAINER ID   IMAGE      COMMAND              CREATED         STATUS         PORTS                                                                                   NAMES
d0aa4ff159df   z3apache   "httpd-foreground"   9 seconds ago   Up 7 seconds   0.0.0.0:8082->80/tcp, :::8082->80/tcp, 0.0.0.0:44445->443/tcp, :::44445->443/tcp   konter
```

User-agent

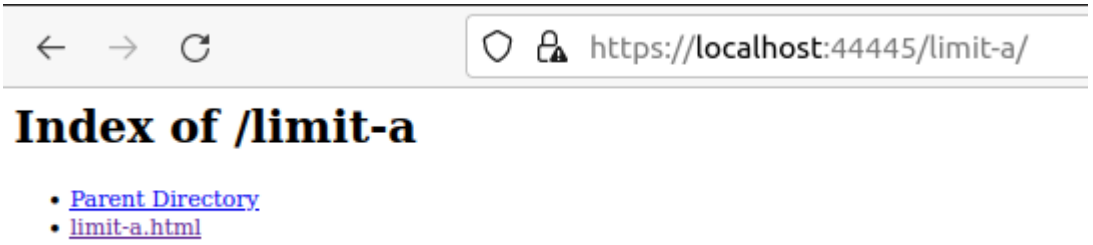
Działanie user-agent.

Przy wykonaniu zbyt wielu żądań następująca odpowiedź.



Limit-a

Na dole widać limit-a.



Błąd przy wysyłaniu dużej liczby żądań w limit-a.

Udane kroki

Gdy to nie wyszło spróbowano kolejnych działań. Usunięto ścieżki user-agent, adresy-ip.

Konfiguracja httpd.conf

Do pliku httpd. conf dodano poniższe komendy, a te wcześniejsze usunięto. Ilość 100 oznacza ilość globalną w 60 sekundach, czyli w minucie. Wskazano ścieżkę limit-a i wysłanie żądania w przypadku zbyt dużej liczby.

`QS_ErrorResponseCode 429` - zostawiono to

`SetEnvIf Request_URI ^/limit-a reguła`

`QS_EventLimitCount reguła 100 60` - 100 żądań i 60 sekund

`SetEnvIf User-Agent curl QS_COND=curl`
`QS_CondClientEventLimitCount 10 60 reguła curl` - 10 żądań klientów

```
559
560 QS_ErrorResponseCode 429
561
562 SetEnvIf Request_URI ^/limit-a reguła
563
564 QS_EventLimitCount reguła 100 60
565
566 SetEnvIf User-Agent curl QS_COND=curl
567 QS_CondClientEventLimitCount 10 60 reguła curl
568
569 <VirtualHost *:80>
```

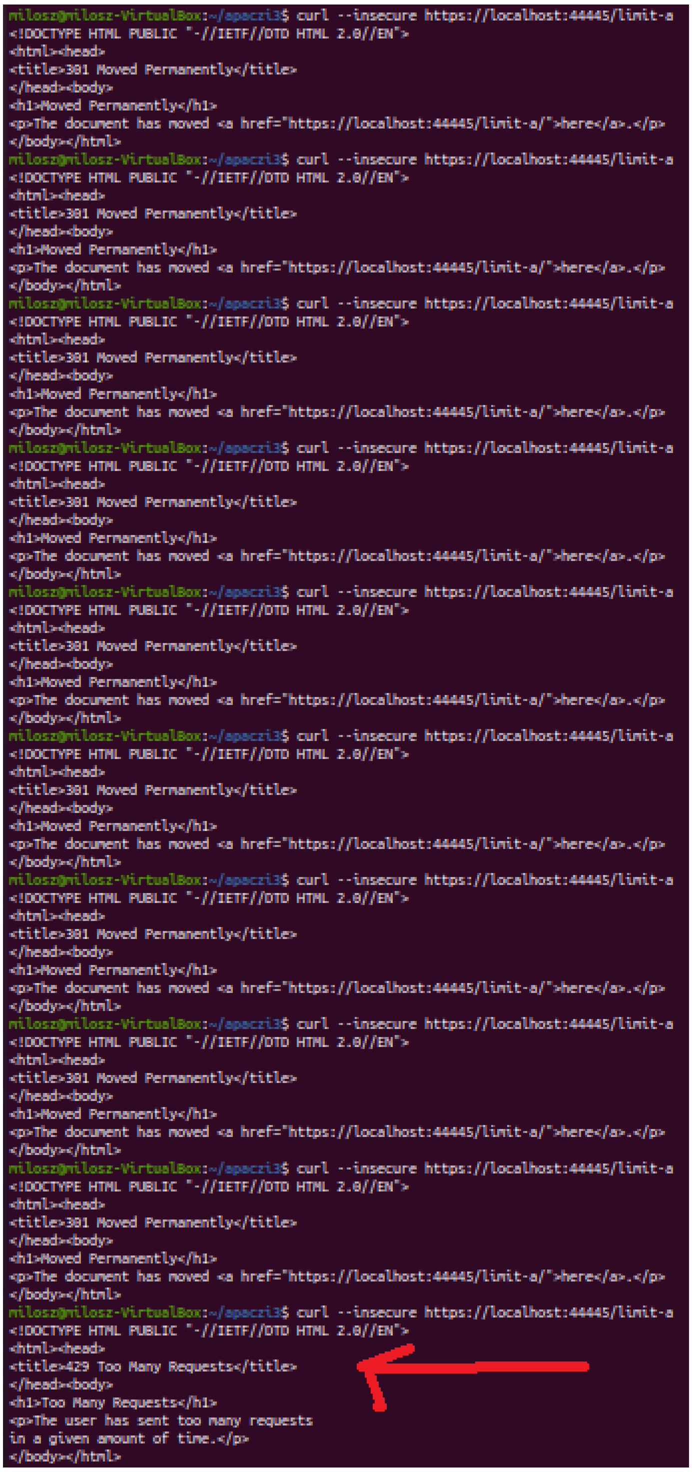
Poniżej widać działanie, które sprawdzono za pomocą polecenia:

`curl --insecure https://localhost:44445/limit-a/limit-a.html`

i podścieżka

`curl --insecure https://localhost:44445/limit-a/limit-a.html`

Działanie limit-a



1

2

3

4

5

6

7

8

9

10 -błąd

Żądanie za żądaniem

```
milosz@milosz-VirtualBox:~/apaczi3$ curl --insecure https://localhost:44445/limit-a/limit-a.html
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>429 Too Many Requests</title>
</head><body>
<h1>Too Many Requests</h1>
<p>The user has sent too many requests
in a given amount of time.</p>
</body></html>
milosz@milosz-VirtualBox:~/apaczi3$ curl --insecure https://localhost:44445/limit-a/limit-a.html
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>429 Too Many Requests</title>
</head><body>
<h1>Too Many Requests</h1>
<p>The user has sent too many requests
in a given amount of time.</p>
</body></html>
milosz@milosz-VirtualBox:~/apaczi3$ curl --insecure https://localhost:44445/limit-a
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>429 Too Many Requests</title>
</head><body>
<h1>Too Many Requests</h1>
<p>The user has sent too many requests
in a given amount of time.</p>
</body></html>
milosz@milosz-VirtualBox:~/apaczi3$ curl --insecure https://localhost:44445/limit-a
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>429 Too Many Requests</title>
</head><body>
<h1>Too Many Requests</h1>
<p>The user has sent too many requests
in a given amount of time.</p>
</body></html>
milosz@milosz-VirtualBox:~/apaczi3$ curl --insecure https://localhost:44445/limit-a
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>429 Too Many Requests</title>
</head><body>
<h1>Too Many Requests</h1>
<p>The user has sent too many requests
in a given amount of time.</p>
</body></html>
milosz@milosz-VirtualBox:~/apaczi3$
```

Działanie limit-a podścieżka

```
</html>
milosz@milosz-VirtualBox:~/apaczi3$ curl --insecure https://localhost:44445/limit-a/limit-a.html
<!doctype html>
<html lang="pl">

<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="style.css">
    <title>Index - Moja strona APACHE</title>
</head>

<body>

    <h1>Witaj na mojej stronie projekt WEB LIMIT 100 global i limit 10 user</h1>
        <p>strona na projekt pwr nginx 100 global i limit 10 user<<p>

</body>

</html>
milosz@milosz-VirtualBox:~/apaczi3$ curl --insecure https://localhost:44445/limit-a/limit-a.html
<!doctype html>
<html lang="pl">

<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="style.css">
    <title>Index - Moja strona APACHE</title>
</head>

<body>

    <h1>Witaj na mojej stronie projekt WEB LIMIT 100 global i limit 10 user</h1>
        <p>strona na projekt pwr nginx 100 global i limit 10 user<<p>

</body>

</html>
milosz@milosz-VirtualBox:~/apaczi3$ curl --insecure https://localhost:44445/limit-a/limit-a.html
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>429 Too Many Requests</title>
</head><body>
<h1>Too Many Requests</h1>
<p>The user has sent too many requests
in a given amount of time.</p>
</body></html>
milosz@milosz-VirtualBox:~/apaczi3$
```

Podsumowanie

Wprowadzenie rate limiting było bardzo trudne w Apache, a w nginx łatwe. Wynikać to może z powodu instalacji modułów. Administrator może wybrać odpowiedni moduł od potrzeby. W zadaniu wykorzystano w moduł qos. Do wielu rzeczy dochodzono metodą prób i błędów. Dużo skomplikował błąd z powodu braku modułu id, ponieważ bardzo długo szukano przyczyny. Jednak najgorsza była pierwsza konfiguracja na szczęście znaleziono inne rozwiązania, aby klienci byli blokowani przy 10 żądaniach.