Bezpieczeństwo Aplikacji WEB

Zadanie 3

Prowadzący:
Mgr inż. Przemysław Świercz

wtorek 8:15
Miłosz Jagodziński CBE

Nginx

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, Nginx

Przebieg ćwiczenia Nginx

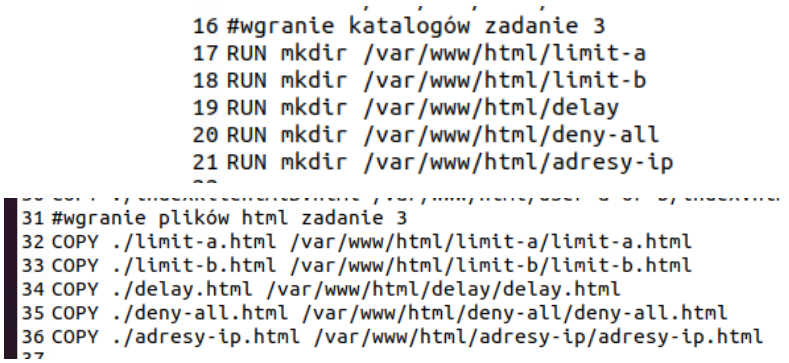
Ostatnie zadanie z konfiguracji nginx, gdzie skonfigurowano limity zapytań globalne i dla użytkownika.

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.

Konfiguracje Dockerfile

Pierwszy krok to utworzenie w moim dockerfile katalogu limit-a dla celu zdania i pod ścieżki do niego limit-a.html. Również utworzono dodatkowe katalogi i pod ścieżki, aby pokazać dodatkowe funkcjonalności.



```
16 #wgranie katalogów zadanie 3
17 RUN mkdir /var/www/html/limit-a
18 RUN mkdir /var/www/html/limit-b
19 RUN mkdir /var/www/html/delay
20 RUN mkdir /var/www/html/deny-all
21 RUN mkdir /var/www/html/adresy-ip
31 #wgranie plików html zadanie 3
32 COPY ./limit-a.html /var/www/html/limit-a/limit-a.html
33 COPY ./limit-b.html /var/www/html/limit-b/limit-b.html
34 COPY ./delay.html /var/www/html/delay/delay.html
35 COPY ./deny-all.html /var/www/html/deny-all/deny-all.html
36 COPY ./adresy-ip.html /var/www/html/adresy-ip/adresy-ip.html
```

Konfiguracja nginx.conf

Następnie zaczęto konfigurować plik nginx.conf, aby wstawić reguły dla rate limiting. Wpierw ustalono na samym początku pliku nginx.conf w bloku http, dyrektywy limit_req_zone.

```
limit_req_zone $binary_remote_addr zone=perip:10m rate=10r/m;
limit_req_zone $server_name zone=perserver:10m rate=100r/m;
# 10 żądań na minutę dla użytkownika rate=10r/m;
# 1000 żądań globalnie w ciągu 60 sekund rate=100r/m;
```

,gdzie

1. limit_req_zone - definiuje parametry żądań i szybkość odpowiedzi
2. $ - oznacza się charakterystykę żądania, do której stosuje się reguły. W tej regule binary_remote_addr oznacza ograniczenie, każdego unikalnego adresu IP oraz ilość żądań od danego adresu IP.
3. Zone - to obszar pamięci współdzielonej, w której przechowuje się stan, każdego adresu IP. 1 megabajt przechowuję 16 000 adresów IP, a więc w tym przykładzie może być aż 160 000 adresów IP. Jeśli pamięć zostanie wyczerpana, gdy NGINX musi dodać nowy wpis, usuwa najstarszy wpis. Jeśli zwolnione miejsce nadal nie wystarcza, aby pomieścić nowy rekord, NGINX zwraca kod stanu. Ponadto, aby zapobiec wyczerpaniu się pamięci, za każdym razem, gdy NGINX tworzy nowy wpis, usuwa maksymalnie dwa wpisy, które nie były używane w ciągu ostatnich 60 sekund.503 (Service Temporarily Unavailable)
4. Rate - jest to maksymalna stawka żądań, wyrazić można ją w żądaniu na sekundę 1r/s lub na minutę 1r/m to administrator ustala liczbę żądań. Jeżeli żądanie nadejdzie z powrotem po wcześniejszym dozwolonym bardzo szybko to zostanie zablokowane.

Uwaga szybkość żądań ogranicza się w określonym bloku server, location.

limit-a

```
            location /limit-a {
#           root /var/www/html/limit-a;
#           autoindex on;
            limit_req zone=perip burst=10 nodelay;
            limit_req zone=perserver burst=10;
            }
```

W tym przykładzie pojawia się parametr `burst` i `nodelay`. `Burst` jest buforem/kolejkomatem w przypadku, gdy klient może wykonuje żądania, przekraczając szybkość określoną przez strefę. Żądanie, które przychodzi wcześniej po poprzednim, jest umieszczane w kolejce, a tutaj ustawiono rozmiar kolejki na 10.

Natomiast `nodelay` przydziela miejsca w kolejce zgodnie z parametrem burst i narzuca skonfigurowany limit szybkości, ale nie przez odstępy w przesyłaniu żądań w kolejce. Zamiast tego, gdy żądanie nadejdzie „zbyt wcześnie", NGINX przekazuje je natychmiast, o ile w kolejce jest dla niego wolne miejsce.

Działanie limit-a

Widzimy podstronę limit-a.html.

Jednak po wysłaniu niezgodnej liczby żądań otrzymujemy następującą odpowiedź 503.

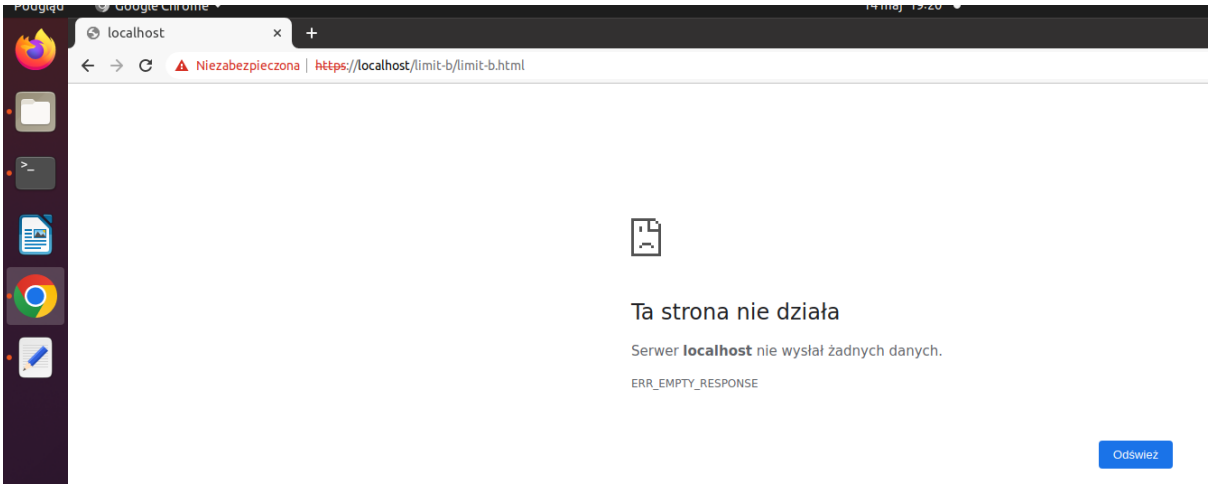
limit-b

Limit-b utworzono, aby pokazać parametr `limit_req_status 444` w przypadku zbyt dużej liczby żądań zamiast 503.

```
            location /limit-b {
#           root /var/www/html/limit-b;
#           autoindex on;
            limit_req zone=perserver burst=10 nodelay;
```

```
        limit_req_status 444;
        }
```

Wywołując zbyt dużo żądań dostaniemy nie dostaniemy odpowiedzi 503 tylko serwer nie będzie chciał się z nami łączyć przez jakiś czas.

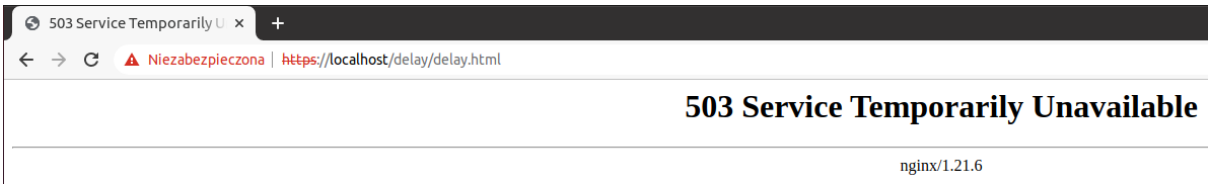


delay

Kolejny dodany parametr to delay jest on przeciwieństwem nodelay. W skrócie można nazwać go dwu stopniowym ograniczaniem szybkości. Na poniższym przykładzie opóźnienie dodawane jest po 6 nadmiernych żądań.

```
        location /delay {
#       root /var/www/html/delay;
#       autoindex on;
        limit_req zone=perip burst=10 delay=6;
        limit_req zone=perserver burst=10;
        }
```

Przykład działania delay 6.

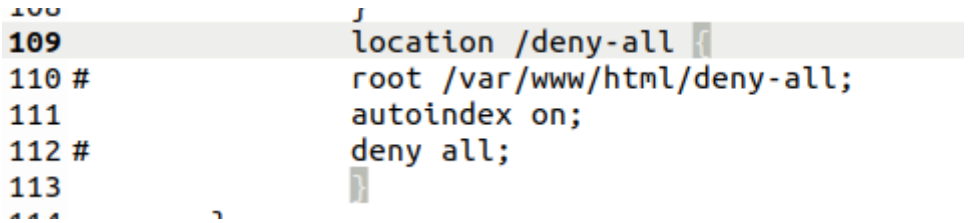


deny-all

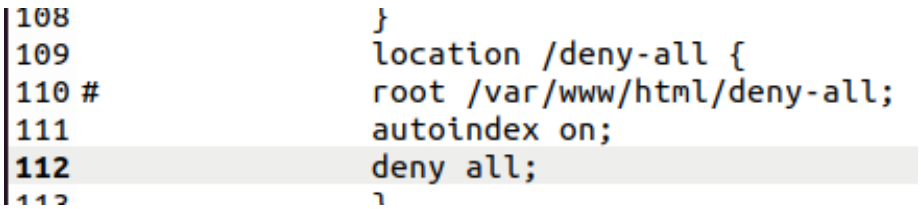
Ostatnie działanie jest niepraktyczne, ale umożliwia odrzucanie wszystkich żądań jest to `deny all`.

```
            location /deny-all {
#           root /var/www/html/deny-all;
#           autoindex on;
            deny all;
            }
```

Przed dodaniem deny-all.



Po dodaniu deny all. Jak widać poniżej mamy brak dostępu tylko nie występuje błąd 503 tylko 403.

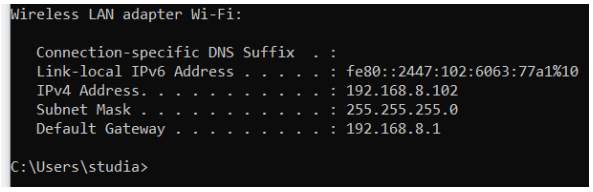


Adresy IP

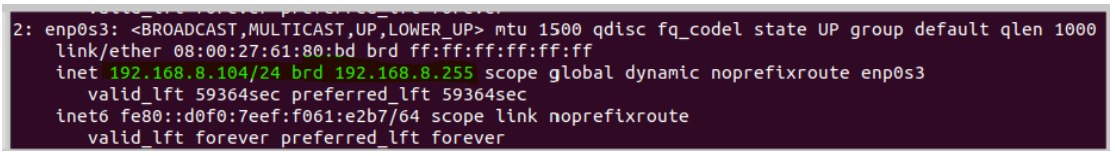
Kolejny przykład to określenie dozwolonych adresów IP. Jako, że moja podsieć to 192.168.8.0/24 to adres gospodarza 192.168.8.102 ustalono ją jako dozwolony, aby pokazać działanie blokady. Adres gościa virtualboxa 192.168.8.104 niedozwolony. Tworzymy zmienną, gdzie 0 oznacza pozwolenie, a 1 niedozwolone adresy IP.

Adresy IP hostów

Gospodarz:



Gość:




```
geo $limit {
          default 1;
          192.168.8.102 0;
          192.168.8.104 1;
}
```

Następnie zmapowano adresy IP.

```
map $limit $limit_key {
   0 "";
    1 $binary_remote_addr;
}
```

,gdzie

0, $limit_key jest to ustawiony na pusty ciąg

1, $limit_key jest to ustawiony na adres IP klienta w formacie binarnym

```
geo $limit {
        default 1;
        192.168.8.102 0;
        192.168.8.104 1;
}


map  $ssl_client_fingerprint $ssl_reject {
    default 1;
    7e0d6f4ec83191db769c5c0a844bbcac4794f481 0;
    bbff4b5f32d33270fb7cafaac3ebc1666ab2dd4b 0;
    }

map $limit $limit_key {
        0 "";
        1 $binary_remote_addr;
}


limit_req_zone $binary_remote_addr zone=perip:10m rate=10r/m;
limit_req_zone $server_name zone=perserver:10m rate=100r/m;

limit_req_zone $limit_key zone=req_zone:10m rate=10r/m;
```

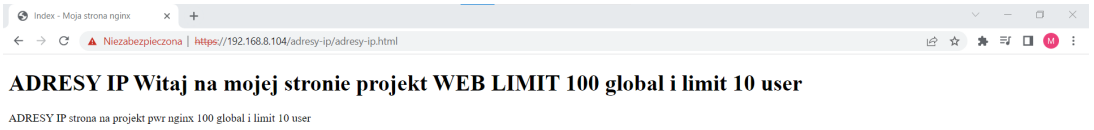
Stworzono nową dyrektywę limit_key dla 10 żądań na minutę dla zdefiniowanego adresu IP.

```
limit_req_zone $limit_key zone=req_zone:10m rate=10r/m;
```

Potem wstawiono do lokacji adresy-ip.

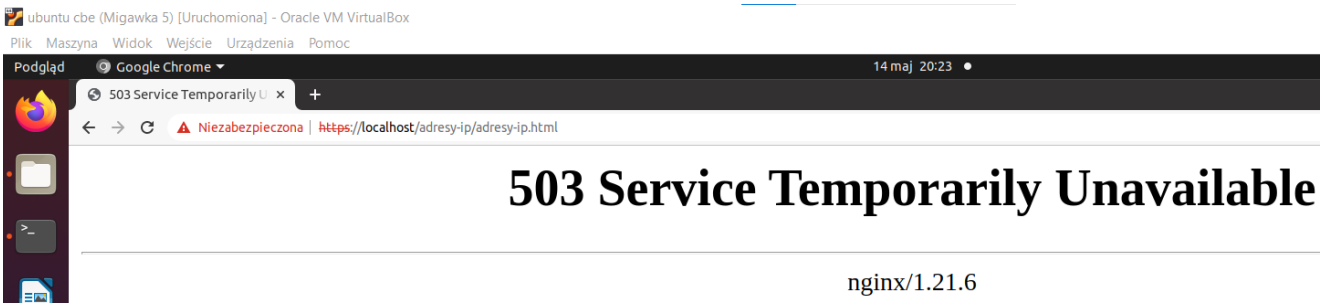
```
        location /adresy-ip {
#       root /var/www/html/adresy-ip;
        autoindex on;
        limit_req zone=req_zone burst=10 nodelay;
        limit_req zone=perserver burst=10 nodelay;
        }
```

Gospodarz pozwolenie 192.168.8.102



Gość virtualbox niepozwolenie 192.168.8.104

Po określonej liczbie żądań wstrzymanie.

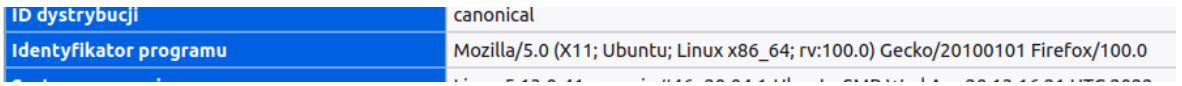


User-agent

Ostatni element to określenie limitu dla user agentów. Bardzo przydatna opcja na boty. Trzeba było utworzyć zmienne i zmapować je, tak aby zależały od innych zmiennych.

```
map $http_user_agent $isbot_ua {
     default 0;
     ~*(Mozilla) 1;
}
```

Ustawiono Mozilla, ponieważ taką nazwę ma user agent w tej przeglądarce w informacjach.



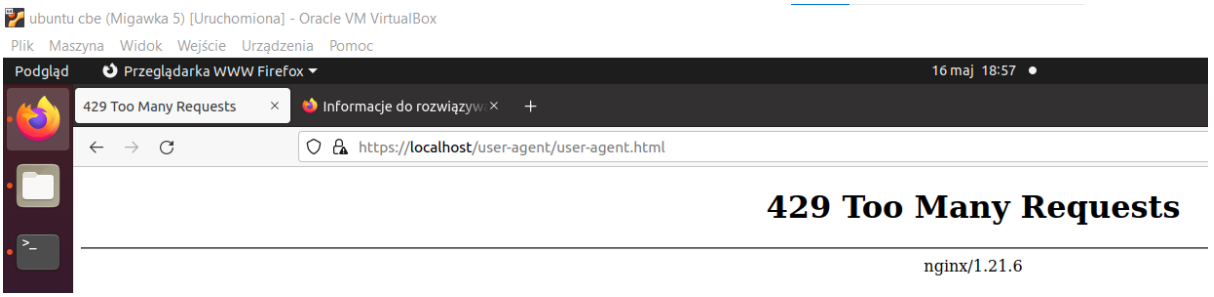

```
map $isbot_ua $limit_bot {
     0       "";
     1       $remote_addr;
}
```

Ostatni krok to skonfigurowanie lokacji user-agenta w pliku konfiguracyjnym nginx.conf

```
     location /user-agent {
```

```
#           root /var/www/html/user-agent;
            autoindex on;
            limit_req zone=bots burst=10 nodelay;
            limit_req zone=boty_globalnie burst=10 nodelay;
            limit_req_status 429;
            }
```

Dodano odpowiedź 429 w przypadku zbyt dużej ilości żądań wysyłanych przez klienta. Działanie w user-agent.



Podsumowanie

Dotychczas najłatwiej konfigurowało się Apache, ale rate limiting w nginx poszedł dość łatwo i szybko. Wynika to z tego, że parametry dodajemy lub edytujemy w pliku nginx.conf. Natomiast w apache trzeba dodawać mody. Rate limiting pozwala chronić serwer przez przeciążeniem i atakami DDoS.