

Instalacja i konfiguracja serwera WWW na Debian Linux

Spis treści

1. Wprowadzenie
2. Wymagania systemowe
3. Instalacja pakietów serwera WWW (Apache)
4. Podstawowa konfiguracja Apache
5. Konfiguracja VirtualHost
6. Zaawansowane parametry konfiguracyjne
7. Zarządzanie modułami Apache
8. Logowanie i monitoring
9. Zabezpieczenia serwera WWW
10. Przeciwdziałanie atakom w warstwach sieciowych
11. Podsumowanie

1. Wprowadzenie

Ten dokument opisuje proces instalacji oraz konfiguracji serwera WWW na systemie Debian GNU/Linux przy użyciu serwera Apache2. Instrukcja zawiera zarówno podstawowe kroki instalacyjne, jak i szczegółowe omówienie parametrów konfiguracyjnych oraz zagadnienia związane z bezpieczeństwem.

2. Wymagania systemowe

- System operacyjny: Debian 10/11/12
- Uprawnienia administratora (root lub sudo)
- Dostęp do Internetu w celu pobrania pakietów

3. Instalacja pakietów serwera WWW (Apache)

Aktualizacja systemu

```
sudo apt update
```

```
sudo apt upgrade -y
```

Instalacja Apache2

```
sudo apt install apache2 -y
```

Uruchomienie i weryfikacja

```
systemctl status apache2
```

```
systemctl enable apache2
```

```
systemctl start apache2
```

Strona testowa powinna być dostępna pod adresem: `http://<adres_IP>`

4. Podstawowa konfiguracja Apache

Główne pliki konfiguracyjne

- `/etc/apache2/apache2.conf` – plik główny
- `/etc/apache2/ports.conf` – konfiguracja portów
- `/etc/apache2/sites-available/` – dostępne konfiguracje VirtualHost
- `/etc/apache2/sites-enabled/` – aktywne konfiguracje

Konfiguracja portów

Domyślnie Apache nasłuchuje na porcie 80:

```
Listen 80
```

Dla HTTPS dodaj:

```
Listen 443
```

Parametry podstawowe

Timeout

Określa maksymalny czas oczekiwania serwera na odpowiedź:

```
Timeout 300
```

KeepAlive

Włącza utrzymywanie połączeń:

```
KeepAlive On
```

```
MaxKeepAliveRequests 100
```

```
KeepAliveTimeout 5
```

5. Konfiguracja VirtualHost

Plik konfiguracyjny:

```
/etc/apache2/sites-available/example.conf
```

Przykład:

```
<VirtualHost *:80>
```

ServerName example.com

ServerAlias www.example.com

DocumentRoot /var/www/example

ErrorLog \${APACHE_LOG_DIR}/example_error.log

CustomLog \${APACHE_LOG_DIR}/example_access.log combined

<Directory /var/www/example>

AllowOverride All

Options Indexes FollowSymLinks

Require all granted

</Directory>

</VirtualHost>

Aktywacja:

sudo a2ensite example.conf

sudo systemctl reload apache2

6. Zaawansowane parametry konfiguracyjne

LimitRequestBody

Ograniczenie rozmiaru żądania:

LimitRequestBody 10485760

MPM (Multi-Processing Module)

Sprawdzenie aktywnego MPM:

apache2ctl -V | grep MPM

Konfiguracja (np. event MPM):

<IfModule mpm_event_module>

StartServers 3

MinSpareThreads 25

MaxSpareThreads 75

ThreadLimit 64

ThreadsPerChild 25

MaxRequestWorkers 400

MaxConnectionsPerChild 0

</IfModule>

Headers

Dodawanie nagłówków bezpieczeństwa:

Header always set X-Frame-Options "DENY"

Header always set X-XSS-Protection "1; mode=block"

Header always set X-Content-Type-Options "nosniff"

7. Zarządzanie modułami Apache

Lista modułów:

apache2ctl -M

Włączanie modułów:

sudo a2enmod rewrite

sudo a2enmod ssl

Wyłączanie:

sudo a2dismod autoindex

8. Logowanie i monitoring

Lokalizacja logów

- /var/log/apache2/access.log
- /var/log/apache2/error.log

Format logów

LogFormat "%h %l %u %t \"%r\" %>s %b" common

Monitoring w czasie rzeczywistym

tail -f /var/log/apache2/access.log

9. Zabezpieczenia serwera WWW

Ustawienie firewall (UFW)

sudo apt install ufw

sudo ufw allow 80/tcp

```
sudo ufw allow 443/tcp
```

```
sudo ufw enable
```

Instalacja certyfikatu SSL (Let's Encrypt)

```
sudo apt install certbot python3-certbot-apache
```

```
sudo certbot --apache
```

Odnawianie certyfikatu:

```
sudo certbot renew --dry-run
```

Ograniczenie dostępu do katalogów

```
<Directory />
```

```
AllowOverride None
```

```
Require all denied
```

```
</Directory>
```

Ukrywanie wersji serwera

```
ServerSignature Off
```

```
ServerTokens Prod
```

10. Przeciwdziałanie atakom w warstwach sieciowych

Klasyfikacja ataków sieciowych na serwer WWW

1. Ataki typu DoS/DDoS (Denial of Service / Distributed Denial of Service)

Celem jest uniemożliwienie korzystania z usługi (np. serwera WWW) poprzez:

- o przeciążenie łącza (wyczerpanie przepustowości),
- o przeciążenie zasobów systemu (CPU, RAM, tablice połączeń),
- o wykorzystanie specyficznych funkcji protokołu (np. TCP, HTTP).

2. Ataki skanowania i rozpoznania (reconnaissance)

Celem jest zebranie informacji o serwerze i infrastrukturze, np.:

- o skanowanie portów (Nmap, Masscan),
- o fingerprinting systemu (OS detection),
- o identyfikacja usług i wersji (bannery, nagłówki HTTP).

3. Ataki na warstwę transportową (L4)

Skupione na mechanizmach TCP/UDP, np.:

- o SYN flood,
- o ACK/RST flood,

- UDP flood.
4. **Ataki na warstwę aplikacyjną (L7)**
Wiedzione przeciwko samej aplikacji HTTP/HTTPS, np.:
- HTTP flood (GET/POST),
 - Slowloris i ataki typu slow HTTP,
 - celowe „ciężkie” zapytania (kosztowne operacje w aplikacji).
5. **Ataki poprzez nadużycie protokołów pomocniczych**
Np. DNS amplification, NTP amplification, memcached amplification – używane jako „wzmacniacze” ruchu przeciwko Twojemu serwerowi.

W dalszej części opiszemy szczegółowo mechanizmy działania wybranych ataków oraz sposoby obrony.

10.2 Ataki DoS/DDoS – mechanizm działania

10.2.1 Pojedynczy DoS (host-based)

Atakujący z jednego hosta generuje duży wolumen ruchu lub specjalnie skonstruowane pakiety, by:

- zająć wszystkie dostępne połączenia serwera (TCP/HTTP),
- wywołać nadmierne zużycie CPU (np. skomplikowane zapytania),
- zablokować kolejkę połączeń (backlog) na gnieździe nasłuchującym.

W przypadku serwera WWW często jest to:

- szybki **HTTP GET/POST flood** (wiele równoległych żądań),
- **TCP SYN flood** (dużo półotwartych połączeń),
- **UDP flood** na port usług towarzyszących.

10.2.2 DDoS rozproszony

W ataku DDoS uczestniczy wiele hostów (botnet), co powoduje:

- rozproszenie źródeł ruchu – trudniej filtrować po IP,
- ogromny wolumen ruchu, często przekraczający przepustowość łącza,
- wykorzystanie technik amplifikacji (DNS, NTP itp.).

Z punktu widzenia administratora serwera WWW istotne jest:

- **odróżnienie przeciążenia łącza od przeciążenia hosta,**
- współpraca z dostawcą usług (ISP, dostawca ochrony DDoS).

10.3 Ataki na TCP – SYN flood i pokrewne

10.3.1 Jak działa TCP handshake

Standardowy handshake TCP:

1. Klient wysyła **SYN** do serwera.
2. Serwer odpowiada **SYN-ACK**.
3. Klient wysyła **ACK** – połączenie jest ustanowione.

Podczas oczekiwania na ACK serwer rezerwuje zasoby (gniazdo półotwarte, wpis w tablicy połączeń). To pole ataku.

10.3.2 Mechanizm ataku SYN flood

Atakujący wysyła:

- dużą liczbę pakietów SYN,
- często z fałszywym (spoofowanym) adresem źródłowym,
- nie odsyła finalnego ACK.

Skutki:

- zapełnienie kolejki półotwartych połączeń,
- odrzucanie nowych, legalnych połączeń,
- wzrost użycia CPU przy obsłudze timeoutów.

10.3.3 Ochrona przed SYN flood

Na poziomie jądra (iptables / nftables, sysctl):

a) SYN cookies – włączenie mechanizmu chroniącego tablicę półotwartych połączeń:

```
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

```
# lub trwale w /etc/sysctl.conf
```

```
net.ipv4.tcp_syncookies = 1
```

b) Ograniczenie liczby półotwartych połączeń i timeoutów:

```
net.ipv4.tcp_max_syn_backlog = 2048
```

```
net.ipv4.tcp_synack_retries = 3
```

```
net.ipv4.tcp_syn_retries = 3
```

c) Filtrowanie pakietów podejrzanych za pomocą iptables:

```
iptables -A INPUT -p tcp --syn --dport 80 -m connlimit --connlimit-above 50 -j DROP
```

```
iptables -A INPUT -p tcp --dport 80 -m state --state NEW -m recent --set
```

```
iptables -A INPUT -p tcp --dport 80 -m state --state NEW -m recent --update --seconds 2 --hitcount 40 -j DROP
```

Wyjaśnienie:

- connlimit ogranicza liczbę jednoczesnych połączeń z jednego IP,
- recent blokuje adresy generujące zbyt dużo nowych połączeń w krótkim czasie.

Na poziomie Apache:

- użycie **Reverse Proxy / Load Balancera** (np. HAProxy, Nginx) przed Apache, który jest lżejszy do obsługi dużej liczby połączeń,
- dostosowanie parametrów MPM, aby zwiększyć odporność na wiele połączeń (ale z głową, patrz rozdział o MPM).

10.4 Ataki UDP flood i ICMP

10.4.1 UDP flood

Atak polega na wysłaniu dużej liczby pakietów UDP na losowe lub konkretne porty serwera.

Skutki:

- przeciążenie pasma,
- obciążenie CPU odpowiedziami ICMP "Port unreachable" (jeśli są generowane),
- możliwe zapchanie kolejek na urządzeniach sieciowych.

Obrona:

- filtrowanie niepotrzebnych portów UDP (firewall),
- ograniczanie natężenia (rate limiting) ICMP i UDP,
- na routerach brzegowych – QoS, policery.

Przykład reguł iptables blokujących niechciany ruch UDP na serwerze WWW (który głównie używa TCP 80/443):

```
iptables -A INPUT -p udp --dport 1:65535 -j DROP
```

lub selektywnie, jeśli używasz np. DNS, VPN, itp.

10.4.2 ICMP flood i ping of death (historycznie)

ICMP flood polega na zasypaniu serwera dużą liczbą pakietów ICMP Echo Request (ping). Choć współczesne systemy są odporne na klasyczny „ping of death”, duże natężenie ruchu ICMP nadal może:

- zużyć pasmo,
- obciążyć CPU przy przetwarzaniu pakietów.

Obrona:

- ograniczenie ICMP na serwerze (ale nie całkowite wyłączenie, bo ICMP jest przydatny diagnostycznie):

Ograniczenie liczby odpowiedzi ICMP:

```
net.ipv4.icmp_ratelimit = 100
```

net.ipv4.icmp_ratemask = 88089

- filtrowanie nadmiarowego ICMP na firewallu lub na routerach brzegowych,
 - w przypadku bardzo wrażliwych serwerów – ograniczenie ICMP Echo Request z Internetu.
-

10.5 Ataki HTTP flood (L7)

Ataki na warstwę aplikacyjną niekoniecznie generują ogromny wolumen ruchu, ale są ukierunkowane na logikę aplikacji i serwera WWW.

10.5.1 HTTP GET/POST flood

Mechanizm:

- atakujący generuje wiele równoległych żądań HTTP GET lub POST,
- mogą to być żądania do zasobów „ciężkich” (np. dynamiczne raporty, generowanie PDF, złożone zapytania do bazy),
- często z zachowaniem poprawnej składni HTTP – trudniejsze do wykrycia jako „nietypowe”.

Skutki:

- wysokie wykorzystanie CPU i RAM,
- przeciążenie bazy danych lub serwisów backend,
- wzrost czasu odpowiedzi dla legalnych użytkowników.

Obrona (Apache i infrastruktura):

1. **mod_evasive** – moduł Apache do prostego rate limiting:

```
sudo apt install libapache2-mod-evasive
```

Przykładowa konfiguracja (np. /etc/apache2/mods-enabled/evasive.conf):

```
<IfModule mod_evasive20.c>
```

```
DOSHashTableSize 3097
```

```
DOSPageCount 20
```

```
DOSPageInterval 1
```

```
DOSSiteCount 300
```

```
DOSSiteInterval 1
```

```
DOSBlockingPeriod 10
```

```
DOSEmailNotify admin@example.com
```

```
DOSLogDir "/var/log/mod_evasive"
```

```
</IfModule>
```

Wyjaśnienie kluczowych parametrów:

- DOSPageCount / DOSPageInterval – ile żądań do tej samej strony na sekundę jest akceptowalne,
- DOSSiteCount / DOSSiteInterval – ile ogólnie żądań z 1 IP na sekundę,
- DOSBlockingPeriod – czas blokady IP (w sekundach).

2. Reverse proxy / WAF (Web Application Firewall)

- Nginx, HAProxy, czy dedykowany WAF (np. ModSecurity) przed Apache,
- filtracja żądań na podstawie reguł (np. OWASP Core Rule Set),
- możliwość łatwiejszego skalowania horyzontalnego (większa liczba serwerów backend).

3. Rate limiting na poziomie sieci (iptables):

```
iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstate NEW -m limit --limit 50/second --limit-burst 200 -j ACCEPT
```

```
iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstate NEW -j DROP
```

- 4. **Ograniczenie kosztownych operacji w samej aplikacji** – optymalizacja zapytań do bazy, cache (Varnish, Redis), CDN.

10.5.2 Slowloris i slow HTTP attacks

Mechanizm ataku Slowloris:

- atakujący nawiązuje wiele połączeń HTTP,
- wysyła nagłówki **bardzo wolno** (utrzymując połączenie przy życiu),
- serwer trzyma zasoby (wątki/procesy) zarezerwowane dla tych połączeń,
- legalne połączenia nie mogą być obsłużone – wyczerpanie puli workerów.

Obrona w Apache:

1. Ograniczenie czasu na otrzymanie żądania:

```
RequestReadTimeout header=20-40,MinRate=500 body=20,MinRate=500
```

2. Ograniczenie liczby równoczesnych połączeń z jednego IP (np. mod_reqtimeout, mod_qos, konfiguracja MPM):

```
<IfModule mpm_event_module>
```

```
MaxRequestWorkers 400
```

```
ServerLimit 16
```

```
</IfModule>
```

3. Wykorzystanie **reverse proxy** (Nginx/HAProxy), które lepiej radzą sobie z dużą liczbą wolnych połączeń.

4. Moduł `mod_evasive` lub `mod_security` z regułami dla slow HTTP.
-

10.6 Skanowanie i rozpoznanie (reconnaissance)

Atakujący przed właściwym atakiem zazwyczaj wykonuje etap rozpoznania:

- skanuje zakresy adresów IP i porty,
- identyfikuje, które porty są otwarte (np. 22, 80, 443),
- próbuje wykryć system operacyjny i wersje usług.

Typowe narzędzia: Nmap, Masscan, ZMap.

Skutki dla obrońcy:

- duża liczba połączeń do wielu portów,
- nietypowe flagi TCP,
- próby łączenia się do portów, które nie powinny być publicznie dostępne.

Obrona:

1. Zasada minimalnej ekspozycji (least exposure)

- Udostępniaj na zewnątrz tylko niezbędne porty (80/443, ewentualnie 22 z dodatkowymi zabezpieczeniami),
- pozostałe blokuj na firewallu (UFW/iptables/nftables).

2. Firewall z domyślną polityką DROP:

```
iptables -P INPUT DROP
```

```
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

```
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

```
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
```

3. Wykrywanie skanów portów:

- IDS/IPS (np. Snort, Suricata) z regułami wykrywającymi nietypowe kombinacje flag TCP,
- monitoring logów systemowych i logów firewall.

4. Ukrywanie informacji o usługach:

- w Apache: `ServerTokens Prod`, `ServerSignature Off` (opisane wcześniej),
- nie eksponuj banerów usług (np. w SSH, FTP) lub ogranicz ich szczegółowość.

10.7 Fail2Ban – automatyczne blokowanie podejrzanych adresów IP

Fail2Ban analizuje logi (np. Apache, SSH) i blokuje adresy IP, które wykazują wzorzec złośliwych działań.

10.7.1 Instalacja

```
sudo apt install fail2ban
```

10.7.2 Konfiguracja dla Apache

Plik główny konfiguracyjny: `/etc/fail2ban/jail.conf` (nie modyfikujemy bezpośrednio – lepiej utworzyć `/etc/fail2ban/jail.local`).

Przykładowa konfiguracja w `/etc/fail2ban/jail.local`:

```
[DEFAULT]
```

```
bantime = 600
```

```
findtime = 600
```

```
maxretry = 5
```

```
backend = systemd
```

```
[apache-auth]
```

```
enabled = true
```

```
port = http,https
```

```
filter = apache-auth
```

```
logpath = /var/log/apache2/error.log
```

```
maxretry = 5
```

```
[apache-badbots]
```

```
enabled = true
```

```
port = http,https
```

```
filter = apache-badbots
```

```
logpath = /var/log/apache2/access.log
```

```
maxretry = 10
```

```
[apache-noscript]
```

```
enabled = true
```

```
port = http,https
```

```
filter = apache-noscript
```

```
logpath = /var/log/apache2/error.log
```

```
maxretry = 5
```

Wyjaśnienie parametrów:

- bantime – czas blokady IP (w sekundach),
- findtime – okres, w którym zliczane są nieudane próby,
- maxretry – liczba dozwolonych naruszeń przed nałożeniem bana,
- filter – nazwa filtra określającego wzorce logów.

Uruchomienie i sprawdzenie:

```
sudo systemctl enable fail2ban
```

```
sudo systemctl start fail2ban
```

```
sudo fail2ban-client status
```

```
sudo fail2ban-client status apache-auth
```

10.8 Ochrona na poziomie iptables / nftables

Firewall jądra Linux jest kluczowym elementem obrony przed atakami sieciowymi.

10.8.1 Podstawowe zasady

1. **Domyślna polityka DROP dla INPUT** – przepuszczaj tylko to, czego potrzebujesz.
2. **Akceptuj połączenia ustanowione** – dzięki modułowi conntrack.
3. **Limituj natężenie nowych połączeń** – chroni przed floodami.
4. **Filtrowanie nietypowych flag TCP** – np. pakiety z wszystkimi flagami ustawionymi lub bez flag (NULL scan).

10.8.2 Przykładowa konfiguracja bazowa iptables dla serwera WWW

```
# Wyczyść istniejące reguły
```

```
iptables -F
```

```
iptables -X
```

```
# Domyślne polityki
```

```
iptables -P INPUT DROP
```

```
iptables -P FORWARD DROP
```

```
iptables -P OUTPUT ACCEPT
```

Zezwól na ruch lokalny

```
iptables -A INPUT -i lo -j ACCEPT
```

Zezwól na połączenia już ustanowione

```
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

Zezwól na SSH (opcjonalnie ogranicz po IP)

```
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

Zezwól na HTTP i HTTPS

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

```
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
```

Filtrowanie anomalii TCP

```
iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP
```

```
iptables -A INPUT -p tcp --tcp-flags ALL ALL -j DROP
```

Ograniczenie ICMP

```
iptables -A INPUT -p icmp -m limit --limit 1/second -j ACCEPT
```

Logowanie odrzuconych pakietów (ostrożnie, by nie zalogować zbyt wiele)

```
iptables -A INPUT -m limit --limit 5/min -j LOG --log-prefix "iptables-denied: " --log-level 7
```

Analogiczną logikę można zastosować w **nftables**, które jest następcą iptables w nowoczesnych dystrybucjach Debiana.

10.9 IDS/IPS oraz monitoring ruchu (Suricata, Snort, NetFlow)

10.9.1 IDS vs IPS

- **IDS (Intrusion Detection System)** – wykrywa podejrzane zdarzenia i generuje alerty,
- **IPS (Intrusion Prevention System)** – dodatkowo może automatycznie blokować atak (integracja z firewallem).

Narzędzia takie jak **Suricata** czy **Snort** analizują ruch sieciowy w czasie rzeczywistym w oparciu o:

- sygnatury znanych ataków,
- statystyczną analizę anomalii,
- reguły tworzone przez administratorów.

10.9.2 Monitoring NetFlow / sFlow

Jeśli operujesz większą infrastrukturą, warto:

- zbierać dane NetFlow/sFlow z routerów i przełączników,
- analizować wolumen ruchu, źródła, docelowe porty,
- wykrywać nagłe skoki ruchu charakterystyczne dla DDoS.

Popularne narzędzia: nfdump, ntopng, Grafana + Prometheus, ELK/Opensearch Stack.

10.10 Współpraca z ISP i usługami ochrony DDoS

W przypadku dużych ataków DDoS, których **nie da się obsłużyć na pojedynczym serwerze**, kluczowa jest współpraca:

- z dostawcą Internetu (ISP), który może filtrować ruch "bliżej źródła", zanim dotrze do Twojego łącza,
- z zewnętrznymi usługami ochrony DDoS / CDN (np. dostawcy chmurowi, usługi typu reverse proxy w chmurze).

Techniki stosowane na poziomie ISP:

- blackholing / null routing dla atakowanego IP,
 - scrubbing center – czyszczenie ruchu i przekazywanie tylko legalnego,
 - filtrowanie ruchu na podstawie BGP (RTBH – Remotely Triggered Black Hole).
-

11. Podsumowanie

Serwer WWW w Debianie można szybko uruchomić, ale jego poprawna konfiguracja i zabezpieczenie wymaga zrozumienia wielu parametrów i praktyk administracyjnych. Poprawnie skonfigurowany serwer minimalizuje ryzyko ataków oraz zapewnia stabilną i bezpieczną pracę.