

Projekt: VPN-miljö på Debian 13 med ZeroTier, NGINX och WordPress

1. Inledning

I detta grupparbete har vi byggt en säker och flexibel VPN-miljö på **Debian 13** i en virtuell miljö.

Syftet var att skapa ett nätverk med flera servrar (brandvägg, WordPress-server och backup-server) som kommunicerar via ett virtuellt privat nätverk.

Projektet fokuserade på **nätwerkssäkerhet, systemadministration och automatisering**.

2. Val av plattform

Vi valde **Debian 13** som bas eftersom:

- det är **stabilt, säkert och öppet**,
- det kräver lite systemresurser,
- det ger full kontroll över servermiljön,
- och det lämpar sig väl för server- och VPN-tjänster utan licenskostnader.

Debian används ofta i produktion för att driva webb-, databas- och säkerhetstjänster, vilket gör det idealiskt för den här uppgiften.

3. VPN-lösning: ZeroTier

Vi implementerade ZeroTier som VPN-lösning.

ZeroTier gör det enkelt att skapa ett **virtuellt, krypterat nätverk över Internet** utan komplicerad manuell konfiguration.

Alla servrar anslöts till samma ZeroTier-nätverk så att de kunde kommunicera som om de låg på ett gemensamt lokalt nätverk.

Installation och anslutning

```
curl -s https://install.zerotier.com | sudo bash
```

```
sudo zerotier-cli join 68BEA79ACF8CB3B0
```

Efter anslutningen godkändes noderna i ZeroTier-kontrollpanelen.

Varför inte Tailscale eller Azure?

Vi valde bort Tailscale och molnlösningar som Azure eftersom:

- ZeroTier ger **full lokal kontroll** och kräver ingen extern molninloggning,
- installationen är **snabbare och enklare**,
- det fungerar utmärkt i en **klassrums- eller labbmiljö** utan extra kostnader.

4. Nätverksdesign

VPN-nätverket bestod av tre centrala delar:

Server	Funktion	Anslutning
Brandväggs-server	Hanterar trafik mellan VPN-noderna och filtrerar anslutningar	Mellan ZeroTier och WordPress
WordPress-server	Webbserver (Apache/NGINX) med WordPress och MariaDB	Bakom brandväggen
Backup-server	Tar emot säkerhetskopior från WordPress-servern	Direktansluten till WordPress-servern

Brandväggen fungerar som skyddande mellanlager mellan externa kunder och interna tjänster.

5. Säkerhets- och nätverkskonfiguration

5.1 Installation av grundläggande verktyg

För att kunna konfigurera nätverket och säkra servrarna installerade vi:

```
sudo apt install curl ufw procps -y
```

5.2 IP-forwarding och UFW

Vi aktiverade IP-forwarding i `/etc/sysctl.conf` och konfigurerade UFW (Uncomplicated Firewall) med anpassade regler i `/etc/ufw/before.rules` för:

- **PREROUTING** och **POSTROUTING** (NAT-trafik),
- filtrering av inkommande och utgående trafik.

5.3 SSH och användarhantering

För fjärradministration aktiverades SSH:

```
sudo apt install openssh-server -y  
sudo useradd <användare>  
sudo usermod -aG sudo <användare>
```

SSH-åtkomst tilläts endast via port 22.

6. NGINX som proxy

Vi använde **NGINX** som **reverse proxy** framför WordPress-servern.

Det gav möjlighet att dirigera trafik, dölja interna adresser och införa last- och åtkomstkontroll.

Exempel på proxy-konfiguration i

`/etc/nginx/sites-available/default:`

```
server {  
  
    server_name proxy.example.com;  
  
    location / {  
  
        proxy_pass http://wordpress.example.local/;  
  
        proxy_set_header Host $host;  
  
        proxy_set_header X-Real-IP $remote_addr;  
  
    }  
  
}
```

Rate-limiting

För att förhindra överbelastning:

```
limit_req_zone $binary_remote_addr zone=one:10m rate=5r/s;
```

```
server {  
    limit_req zone=one burst=10;  
  
    limit_req_status 429;  
}
```

Testades med **siege**-verktyget för att verifiera svarskoder:

404 Not Found, 502 Bad Gateway, 429 Too Many Requests, 503 Service Unavailable.

7. WordPress- och databasserver

På WordPress-servern installerades nödvändiga program:

```
sudo apt install apache2 mariadb-server php php-mysql -y
```

Sedan konfigurerades:

1. Databas och användare i MariaDB,
2. WordPress-filstruktur och **wp-config.php**,
3. Lokal brandvägg och **Fail2Ban** för inloggningsskydd,
4. Själva hemsidan (tema och innehåll).

8. Backup-lösning och automatisering

Vi skapade ett **bash-script (.sh)** som automatiserar säkerhetskopiering av WordPress-servern.

Scriptets funktioner

1. Skapa en katalog med **datum/tid** som namn.
2. Fyll katalogen med **MySQL-dump** och **WordPress-filer**.
3. Kopiera katalogen till **backup-servern** via SSH.
4. Radera den lokala kopian efter överföringen.

Automatisering

- Rättigheter sattes med **chmod +x**.

SSH-nyckel genererades och kopierades till backup-servern:

ssh-keygen

ssh-copy-id backup@server

-
- Scriptet körs automatiskt var sjätte timme via **cron (crontab -e)**.

9. Arbetsmetod och förbättringsområden

Arbetsmetod

Gruppen arbetade **agilt** med dagliga korta planeringar och uppföljningar.

Alla medlemmar ansvarade för olika delar (VPN, proxy, backup, webbtjänst) men samarbetade vid felsökning och verifiering.

Förbättringsområden

- Utöka nätverket med **molnuppkoppling** för redundans.
- Lägg till **ytterligare WordPress-/databasserver** för lastbalansering.
- Implementera fler säkerhetslösningar som TLS och flerlagers-autentisering.

10. Lärdomar

Vi lärde oss:

- hur man **planerar och bygger en VPN-baserad nätverksmiljö**,
- hur man **konfigurerar brandväggar, proxytjänster och backups**,
- vikten av att **verifiera varje steg** och samarbeta effektivt,
- att **säkerhet och automatisering** är centrala i moderna IT-miljöer.

11. Sammanfattning

Projektet resulterade i en fungerande, säker och flexibel VPN-miljö byggd på Debian 13.

Med hjälp av ZeroTier, NGINX, WordPress och automatiserade backup-skript skapade gruppen en komplett lösning med fokus på **säkerhet, driftsäkerhet och samarbete**.