Firewall and network security basics(iptables/nftables && firewalld). VPN basics

Основи мережевої безпеки та роль firewall

Що таке мережева безпека

Мережа - це головний канал атак. Мета мережевої безпеки - **контроль і фільтрація трафіку**. Основні завдання:

- Захист від несанкціонованого доступу.
- Обмеження вихідного трафіку.
- Виявлення аномальної активності.

Безпека мережі реалізується на кількох рівнях:

- **Фізичний рівень:** доступ до обладнання.
- **Мережевий рівень:** маршрути, IP-адреси, пакети.
- **Транспортний рівень:** TCP/UDP порти, сесії.
- **Прикладний рівень:** протоколи (HTTP, SSH, DNS).

Firewall як ключовий елемент безпеки

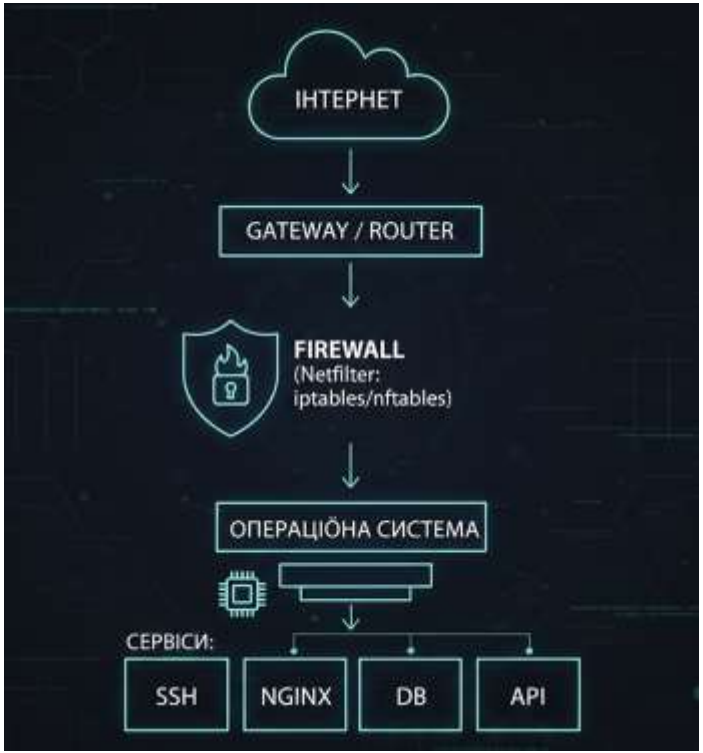
Firewall - це система контролю трафіку на основі набору правил.

- Працює на рівні **ядра** (kernel space), а не лише користувацьких процесів.
- Може виконувати:
  - **Filtering** - блокування або дозвіл пакетів.
  - **NAT (Network Address Translation)** - трансляцію адрес.
  - **Logging** - журналювання підозрілих дій.
  - **Rate limiting** - обмеження частоти запитів.
- У Linux роль firewall виконує **Netfilter** - компонент ядра, з яким працюють:
  - **iptables** (старий інтерфейс)
  - **nftables** (новий, сучасний інтерфейс)

Місце Firewall у моделі безпеки

Рівні захисту:

1. Cloud-level (AWS Security Groups, Network ACLs)
2. Host-level (iptables/nftables)
3. Application-level (WAF, Reverse Proxy, Auth Layer)



Основні типи firewall

Тип	Опис	Приклад
Packet Filtering Firewall	Аналізує окремі пакети, рішення на основі IP/портів	iptables / nftables
Stateful Firewall	Відстежує стан з'єднань (NEW, ESTABLISHED, RELATED)	Netfilter з `conntrack`
Application Firewall (WAF)	Працює на рівні протоколів HTTP, SMTP, DNS	AWS WAF, Nginx ModSecurity
Next-Gen Firewall (NGFW)	Поєднання всіх підходів + DPI (Deep Packet Inspection)	Palo Alto, Fortinet, Cisco ASA

Netfilter — серце Linux Firewall

Netfilter - це ядрова підсистема Linux, що перехоплює пакети.

- Використовує **hooks** - спеціальні точки у стеку TCP/IP:
  - `PREROUTING` - перед маршрутизацією
  - `INPUT` - для локальних пакетів
  - `FORWARD` - для транзитних
  - `OUTPUT` - для вихідних
  - `POSTROUTING` - після маршрутизації
- Із цими hook’ами взаємодіють:
  - `iptables`
  - `nftables`
  - `firewalld`, `ufw`, тощо

Концепція “Stateful Firewall”

Conntrack (connection tracking):

- Firewall пам’ятає стан з’єднання.
- Стани:
  - **NEW** - новий запит
  - **ESTABLISHED** - продовження дозволеного з’єднання
  - **RELATED** - пов’язаний трафік (FTP data, ICMP error)
  - **INVALID** - пошкоджені або неочікувані пакети

Сучасний firewall не аналізує пакет ізольовано - він пам’ятає, які з’єднання уже встановлені.

Як тільки клієнт відкрив SSH-сесію - **усі наступні пакети з цією сесією автоматично пропускаються**.

Це називається **stateful-фільтрація** і вона значно зменшує навантаження та підвищує безпеку.

Приклад iptables

iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

Частина	Значення
`-m state`	увімкнення відстеження стану
`--state ESTABLISHED,RELATED`	дозволяємо тільки відомі з'єднання
`-j ACCEPT`	пропускаємо пакет

Netfilter більше, ніж просто DROP/ACCEPT

Функція	Навіщо використовується
Filtering	Дозволити / Заборонити доступ
NAT	Інтернет-доступ, публікація сервісів
Mangle	Маніпуляції з полями TCP/IP
Logging	Журнали інцидентів
Rate-Limit	Захист від DoS/Bruteforce

Взаємодія з інструментами в Linux

Рівень	Інструменти	Коментар
CLI firewall	iptables, nftables	Прямий контроль Netfilter
High-level firewall	firewalld, UFW	Спрощена абстракція
Containers / Kubernetes	Docker iptables rules, Kube-proxy	Автоматична зміна NAT/filters

Що таке iptables

iptables - інструмент керування правилами Netfilter, який працює з **IPv4** (для IPv6 існує `ip6tables`) та базується на структурі:

- **Tables** (таблиці) - функціональні групи правил.
- **Chains** (ланцюги) - точки перехоплення пакетів.
- **Rules** (правила) - інструкції для прийняття рішення.
- **Targets** - що робити з пакетом (ACCEPT, DROP).

Таблиці iptables та їх призначення

Таблиця	Призначення	Хук-ланцюги
filter	Фільтрація (вхід, вихід, транзит)	INPUT, FORWARD, OUTPUT
nat	Трансляція адрес (DNAT/SNAT)	PREROUTING, OUTPUT, POSTROUTING
mangle	Маніпуляція заголовками (QoS/TTL)	Усі
raw	Вимкнення conntrack	PREROUTING, OUTPUT

Типи ланцюгів (chains)

Chain	Значення
INPUT	Для трафіку, адресованого цьому хосту
OUTPUT	Вихідні пакети з даного хоста
FORWARD	Трафік між інтерфейсами (хост як маршрутизатор)
PREROUTING	До визначення маршруту пакета
POSTROUTING	Після маршрутизації, перед відправкою

Основні таргети (дії над пакетом)

Target	Опис
ACCEPT	Пропустити пакет
DROP	Відкинути без повідомлення
REJECT	Відкинути з повідомленням (TCP reset/ICMP error)
LOG	Записати подію в журнал
MASQUERADE	NAT для динамічного зовнішнього IP
DNAT / SNAT	Перепризначення адрес/портів
RETURN	Вийти з поточного ланцюга

Перегляд поточного ruleset

iptables -L -n -v

Опція	Що робить
`-L`	Список правил
`-n`	Не робити DNS-lookup (швидкість і точність)
`-v`	Лічильники пакетів і байт

Додавання правил

Приклад: Дозволити SSH:

iptables -A INPUT -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT

Компонент	Пояснення
`-A INPUT`	Додаємо правило в кінець INPUT
`-p tcp`	Протокол TCP
`--dport 22`	Порт призначення (SSH)
`-m state`	Модуль станів
`--state NEW,ESTABLISHED`	Нові + існуючі з'єднання
`-j ACCEPT`	Дозволити

Типова структура правил у продакшені

Порядок правил критично важливий:

1. Дозволити loopback (localhost)
   iptables -A INPUT -i lo -j ACCEPT
1. Дозволити ESTABLISHED,RELATED
   iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
1. Дозволити необхідні сервіси (наприклад SSH)
   iptables -A INPUT -p tcp --dport 22 -j ACCEPT
1. Логування підозрілих пакетів
   iptables -A INPUT -j LOG --log-prefix "DROP INPUT: "
1. DROP all (останнє правило)
   iptables -A INPUT -j DROP

Типова структура правил у продакшені

Порядок правил критично важливий:

1. Дозволити loopback (localhost)
   iptables -A INPUT -i lo -j ACCEPT
1. Дозволити ESTABLISHED,RELATED
   iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
1. Дозволити необхідні сервіси (наприклад SSH)
   iptables -A INPUT -p tcp --dport 22 -j ACCEPT
1. Логування підозрілих пакетів
   iptables -A INPUT -j LOG --log-prefix "DROP INPUT: "
1. DROP all (останнє правило)
   iptables -A INPUT -j DROP

Файл резервного ruleset

Перед будь-якими змінами обов’язково робимо backup.

iptables-save > /root/fw.backup

У випадку аварії — одна команда повертаємо доступ.

iptables-restore < /root/fw.backup

iptables persistence (після перезавантаження)

Linux **не зберігає** правила iptables автоматично: після `reboot` все повертається до дефолту (Потрібно вручну увімкнути **iptables persistence**).

Встановлення пакету:

sudo apt-get install iptables-persistent

Збереження поточного ruleset:

sudo netfilter-persistent save

Файли конфігів:
/etc/iptables/rules.v4
/etc/iptables/rules.v6

nftables — сучасна заміна iptables

Чому nftables замінює iptables?

Проблеми iptables:

- правило → тільки текстовий список (повільний пошук зверху вниз)
- IPv4 і IPv6 окремо (iptables vs ip6tables)
- складність масштабування (тисячі правил)
- дубльована логіка в різних таблицях
- важко валідувати та автоматизувати

Переваги nftables:

- **Єдина мова** для IPv4/IPv6 (`family inet`)
- **Sets & Maps** — зберігання множин IP/портів
- **Менше правил — більше ефективності**
- **Компактний синтаксис**
- Вищий перформанс на великих системах
- Нативна інтеграція з conntrack/counters/log

Архітектура nftables

На відміну від iptables, nftables створює **logically grouped firewall конфігурацію**.Таблиця - це контейнер, у якому є chains, а всередині - правила з логікою. Така структура дозволяє легко читати, оновлювати та масштабувати політику.

Tables
└── ***Chains***
└── ***Rules***
└── ***Expressions & Statements***

nftables ruleset basics

Команда для перегляду всього ruleset:
nft list ruleset
Створення таблиці та ланцюга в nftables:
nft add table inet firewall
nft add chain inet firewall input { type filter hook input priority 0 ; }

Команда	Значення
`table inet firewall`	таблиця для IPv4 та IPv6 одночасно
`chain input`	ланцюг для вхідного трафіку
`type filter`	фільтрація
`hook input`	точка входу
`priority 0`	порядок виконання

Повний базовий ruleset nftables

```
nft add table inet firewall
nft add chain inet firewall input { type filter hook input priority 0 ; }
nft add rule inet firewall input iif lo accept
nft add rule inet firewall input ct state established,related accept
nft add rule inet firewall input tcp dport 22 accept
nft add rule inet firewall input ct state invalid drop
nft add rule inet firewall input limit rate 3/min log prefix "DROP RATE: "
nft add rule inet firewall input drop
```

VPN basics

Вступ до VPN: Що це і навіщо нам це?

VPN - це абревіатура від **Virtual Private Network**, що перекладається як **Віртуальна Приватна Мережа**. Фактично, VPN створює **безпечний, зашифрований тунель** для передачі даних через менш захищену, або й зовсім незахищену, публічну мережу, якою найчастіше є **Інтернет**.

Для DevOps-інженерів, VPN - це **критично важливий інструмент** безпеки та роботи.

- **Безпечний доступ до інфраструктури:** Ми використовуємо VPN для безпечного підключення до **приватних підмереж** у хмарі (**AWS VPC**, **Azure VNet**, **Google Cloud VPC**) або до **дата-центрів** компанії. Це гарантує, що конфігураційні файли, ключі та чутливі дані не будуть скомпрометовані під час передачі через публічний Інтернет.
- **Управління ресурсами:** VPN дозволяє нам працювати так, ніби ми фізично знаходимося в офісі або в тій же локальній мережі, що й сервери. Це необхідно для доступу до внутрішніх систем, таких як **Jenkins**, **GitLab**, **Prometheus/Grafana** або **приватні реєстри Docker**.

Отже, VPN - це не просто анонімність, це **основа безпечної віддаленої роботи** з критичною інфраструктурою.

Тунелювання та Шифрування

VPN базується на двох фундаментальних концепціях: **Тунелювання** (Tunneling) та **Шифрування** (Encryption).

Тунелювання - це процес інкапсуляції, тобто **упаковки** одного мережевого протоколу всередину іншого.

- **Мета:** Створити логічне, ізольоване з'єднання, яке проходить через публічну мережу.
- **Як це працює:** Коли ваш комп'ютер відправляє дані, VPN-клієнт бере ваш **IP-пакет** (наприклад, IP-пакет, призначений для вашого внутрішнього сервера), **загортає** його в новий IP-пакет, додаючи **VPN-заголовок**(header). Це як лист, загорнутий у більший, зовнішній конверт.
- **Результат:** Зовнішній пакет переміщується через Інтернет, а внутрішній пакет залишається **недоторканим і прихованим** до моменту досягнення VPN-сервера.

Шифрування - це математичний процес перетворення даних (plaintext) на нечитабельний код (ciphertext).

- **Мета:** Забезпечити **конфіденційність** (Confidentiality) даних під час передачі. Навіть якщо зловмисник перехопить тунель, він побачить лише безладний набір символів.
- **Роль у VPN:** Шифрування застосовується до інкапсульованого (упакованого) пакету, перш ніж він буде відправлений у тунель. На VPN-сервері відбувається зворотний процес - **дешифрування**.

Типи VPN: Site-to-Site vs. Remote Access

Remote Access VPN (Віддалений доступ)

- **Сценарій:** З'єднання встановлюється між **окремим користувачем** (наприклад, вашим ноутбуком або смартфоном) та **приватною мережею** компанії.
- **Мета:** Дозволити віддаленим працівникам безпечно підключатися до корпоративних ресурсів, ніби вони знаходяться в офісі.

Site-to-Site VPN (Мережа-до-Мережі)

- **Сценарій:** VPN-з'єднання встановлюється між **VPN-шлюзами** (VPN Gateways) на обох кінцях. Це не кінцевий користувач, а мережеве обладнання.
- **Мета:** Створити єдину, велику приватну мережу, що охоплює географічно віддалені офіси або хмарні середовища.

Популярні VPN-Протоколи: IPsec, OpenVPN, WireGuard

Протокол: IPsec

IPsec (Internet Protocol Security)

- **Опис:** Це **набір протоколів**, що працює на мережевому рівні (**Layer 3**). Він є галузевим стандартом, особливо для **Site-to-Site VPN**.
- **Ключові компоненти:**
  - **AH (Authentication Header):** Забезпечує цілісність даних (Data Integrity) та автентифікацію (Authentication), але не шифрує.
  - **ESP (Encapsulating Security Payload):** Забезпечує і цілісність, і автентифікацію, **і шифрування**. Найчастіше використовується саме ESP.
- **Режими роботи:**
  - **Transport Mode:** Шифрує лише дані (payload). Використовується рідко.
  - **Tunnel Mode:** Шифрує **весь IP-пакет** (і заголовок, і дані). Це стандарт для VPN-з'єднань.

Протокол: OpenVPN

- **Опис:** Це **відкрите, гнучке та дуже популярне** VPN-рішення.
- **Переваги:** Висока гнучкість, може працювати поверх **TCP** (для надійності, хоча повільніше) або **UDP** (для швидкості). Активно використовує бібліотеку **OpenSSL** для шифрування.
- **Конфігурація:** Часто вимагає складнішої конфігурації порівняно з іншими, але є **де-факто стандартом** для кастомних рішень.

Протокол: WireGuard

- **Опис:** Відносно **новий** (порівняно з IPsec та OpenVPN) та **інноваційний** протокол.
- **Переваги:**
  - **Швидкість:** Значно швидший за OpenVPN та IPsec завдяки меншій кількості коду та використанню новітніх криптографічних примітивів (як-от ChaCha20).
  - **Простота:** Має дуже маленьку кодову базу, що спрощує аудит і зменшує ймовірність помилок безпеки.
- **Статус:** Швидко набирає популярність і вже інтегрований у ядро Linux, що свідчить про його надійність.

Firewalld

Firewalld is a firewall management tool for Linux operating systems. It provides firewall features by acting as a front-end for the Linux kernel's netfilter framework via the nftables user space utility (before v0.6.0 iptables backend), acting as an alternative to the nft command line program. The name firewalld adheres to the Unix convention of naming system daemons by appending the letter "d"

firewalld supports both IPv4 and IPv6 networks and can administer separate firewall zones with varying degrees of trust as defined in zone profiles. Administrators can configure Network Manager to automatically switch zone profiles based on known Wi-Fi (wireless) and Ethernet (wired) networks, but firewalld cannot do this on its own.

Services and applications can use the D-Bus interface to query and configure the firewall. firewalld supports timed rules, meaning the number of connections (or "hits") to a service can be limited globally. There is no support for hit-counting and subsequent connection rejection per source IP; a common technique deployed to limit the impact of brute-force hacking and distributed denial-of-service attacks.

firewalld's command syntax is similar to but more verbose than other iptables front-ends like Ubuntu's Uncomplicated Firewall (ufw). The command-line interface allows managing firewall rulesets for protocol, ports, source and destination; or predefined services by name.

Install Firewalld

The default firewall system for Ubuntu is ufw but you can install and use Firewalld if you prefer. Install Firewalld on Ubuntu 18.04 / Ubuntu 16.04 by running the commands:

```
sudo apt-get install firewalld
```

By default, the service should be started, if not running, start and enable it to start on boot:

```
sudo systemctl enable firewalld
sudo systemctl start firewalld
```

Install Firewalld

Confirm that the service is running:

```
$ sudo firewall-cmd --state
running
```

If you have ufw enabled, disable it to make firewalld your default firewall

```
sudo ufw disable
```

Use Firewalld

Now that the package has been installed and firewalld service started, let' look at few usage examples See below examples for the basic usage of firewalld.

1. List all firewall rules configured

`ssh` and `dhcpv6-client` services are enabled by default when you start firewalld service.

```
# firewall-cmd --list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh dhcpv6-client
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Use Firewalld

2. Get a list of all services that can be enabled using a name

```
sudo firewall-cmd --get-services
```

3. Enable `http` service

```
sudo firewall-cmd --add-service=http --permanent
```

The `--permanent` option means persist rules against server reboots.

Use Firewalld

4. Enable both http and https on a single line

```
sudo firewall-cmd --permanent --add-service={http,https} --permanent
```

5. Enable TCP port 7070

```
sudo firewall-cmd --add-port=7070/tcp --permanent
```

Use Firewalld

6. Enable UDP port 514

```
sudo firewall-cmd --add-port=514/udp --permanent
```

7. Create a new zone

```
sudo firewall-cmd --new-zone=myzone --permanent
```

8. Enable service on a specific zone

```
sudo firewall-cmd --zone=myzone --add-port=4567/tcp --permanent
```

9. Set default zone

```
sudo firewall-cmd --set-default-zone=public --permanent
```

Use Firewalld

10. Add an interface to a zone

```
sudo firewall-cmd --get-zone-of-interface=eth0 --permanent
sudo firewall-cmd --zone=<zone> --add-interface=eth0 --permanent
```

11. Allow access to a port from specific subnet/IP

```
$ sudo firewall-cmd --add-rich-rule 'rule family="ipv4" service name="ssh" \
source address="192.168.0.12/32" accept' --permanent
$ sudo firewall-cmd --add-rich-rule 'rule family="ipv4" service name="ssh" \
source address="10.1.1.0/24" accept' --permanent
```

Use Firewalld

12. List rich rules

```
sudo firewall-cmd --list-rich-rules
```

13. Configure Port forwarding

```
# Enable masquerading
$ sudo firewall-cmd --add-masquerade --permanent

# Port forward to a different port within same server ( 22 > 2022)
$ sudo firewall-cmd --add-forward-port=port=22:proto=tcp:toport=2022 --permanent

# Port forward to same port on a different server (local:22 > 192.168.2.10:22)
$ sudo firewall-cmd --add-forward-port=port=22:proto=tcp:toaddr=192.168.2.10 --permanent

# Port forward to different port on a different server (local:7071 > 10.50.142.37:9071)
$ sudo firewall-cmd --add-forward-port=port=7071:proto=tcp:toport=9071:toaddr=10.50.142.37 --permanent
```

Use Firewalld

14. Removing port/service

Replace `--add` with `--remove`

Q&A