

Nftable – Konfiguracja Firewalla od linuxa Cheatsheet

Konfiguracje rozpoczynamy od sprawdzenia czy mamy nftables na maszynie używamy polecenia „*nft --version*”.

Jeśli nie mamy no to pobieramy

“*sudo apt update*

sudo apt install nftables”

oraz sprawdzamy czy istnieje już jakikolwiek np. domyślny ruleset „*sudo nft list ruleset*”

```
(kali@kali)-[~]
$ nft --version
nftables v1.1.0 (Commodore Bullmoose)

(kali@kali)-[~]
$ sudo nft list ruleset
table inet filter {
    chain input {
        type filter hook input priority filter; policy accept;
    }

    chain forward {
        type filter hook forward priority filter; policy accept;
    }

    chain output {
        type filter hook output priority filter; policy accept;
    }
}
```

Usuwaamy istniejący ruleset i konfigurujemy go od nowa(zalecam zrobić to na maszynie wirtualnej żeby uniknąć późniejszych komplikacji, pamiętaj że robimy to w celach edukacyjnych więc VM będzie lepszą opcją) komenda która nam to umożliwi to „*sudo nft flush ruleset*”

```
(kali@kali)-[~]
$ sudo nft flush ruleset

(kali@kali)-[~]
$ sudo nft list ruleset

(kali@kali)-[~]
$ sudo nft add table inet filter

(kali@kali)-[~]
$ sudo nft add chain inet filter input '{ type filter hook input priority 0; policy drop; }'

(kali@kali)-[~]
$ sudo nft add chain inet filter forward '{ type filter hook forward priority 0; policy drop; }'

(kali@kali)-[~]
$ sudo nft add chain inet filter output '{ type filter hook output priority 0; policy accept; }'

(kali@kali)-[~]
$ sudo nft add rule inet filter input iif "lo" accept
```

Tworzenie tabeli

„*sudo nft add table inet filter*”

Następnie tworzę nową tabelę o nazwie filter w rodzinie protokołów inet, co oznacza, że tabela ta będzie obsługiwać zarówno protokoły IPv4, jak i IPv6. Tabele w nftables działają jako kontenery dla łańcuchów i reguł, umożliwiając lepszą organizację reguł związanych z filtrowaniem ruchu sieciowego

Tworzenie łańcuchów

Łańcuch "input"

```
„sudo nft add chain inet filter input '{ type filter hook input priority 0; policy drop; }'”
```

Tworzę łańcuch o nazwie input, który będzie analizował ruch przychodzący do systemu. Definiuję go jako typ filtrujący, ustalam jego priorytet na 0 i domyślną politykę na drop, co oznacza, że wszystkie pakiety, które nie spełniają żadnej reguły, zostaną odrzucone. Celem jest zwiększenie bezpieczeństwa systemu poprzez zabezpieczenie go przed niepożądanym ruchem.

Łańcuch „forward”

```
“sudo nft add chain inet filter forward '{ type filter hook forward priority 0; policy drop; }'”
```

Tworzę łańcuch o nazwie forward, który będzie odpowiedzialny za analizowanie ruchu przekazywanego przez system. Ustawiam również politykę domyślną na drop, aby chronić system przed nieautoryzowanym dostępem do pakietów, które są routowane, co jest szczególnie ważne w przypadku, gdy system działa jako router.

Łańcuch „output”

Dodaję łańcuch o nazwie output, który analizuje ruch wychodzący z systemu. W przeciwieństwie do poprzednich łańcuchów, polityka dla łańcucha output jest ustawiona na accept, co pozwala na swobodny ruch wychodzący. To rozwiązanie jest ważne dla komfortu użytkownika i działania aplikacji, ale wymaga ostrożności w monitorowaniu ruchu.

Dodanie reguł

Akceptacja ruchu z interfejsu loopback

```
“sudo nft add rule inet filter input iif "lo" accept”
```

Dodaję regułę, która akceptuje wszelki ruch przychodzący z interfejsu loopback. To kluczowe dla lokalnej komunikacji między aplikacjami działającymi na tym samym systemie, zapewniając, że mogą one efektywnie wymieniać informacje.

```
(kali@kali)-[~]
$ sudo nft list ruleset
table inet filter {
    chain output {
        type filter hook output priority filter; policy accept;
    }

    chain input {
        type filter hook input priority filter; policy drop;
        iif "lo" accept
    }

    chain forward {
        type filter hook forward priority filter; policy drop;
    }
}
```

Akceptacja ruchu na porcie SSH

`"sudo nft add rule inet filter input tcp dport 22 accept"`

Umożliwiam ruch przychodzący na porcie TCP 22, używanym przez SSH. Ta reguła jest niezbędna dla administratorów, którzy potrzebują zdalnego dostępu do systemu, ale wymaga odpowiedniego zabezpieczenia, aby uniknąć nieautoryzowanego dostępu.

Akceptacja ruchu na portach HTTP i HTTPS

`"sudo nft add rule inet filter input tcp dport 80 accept"`

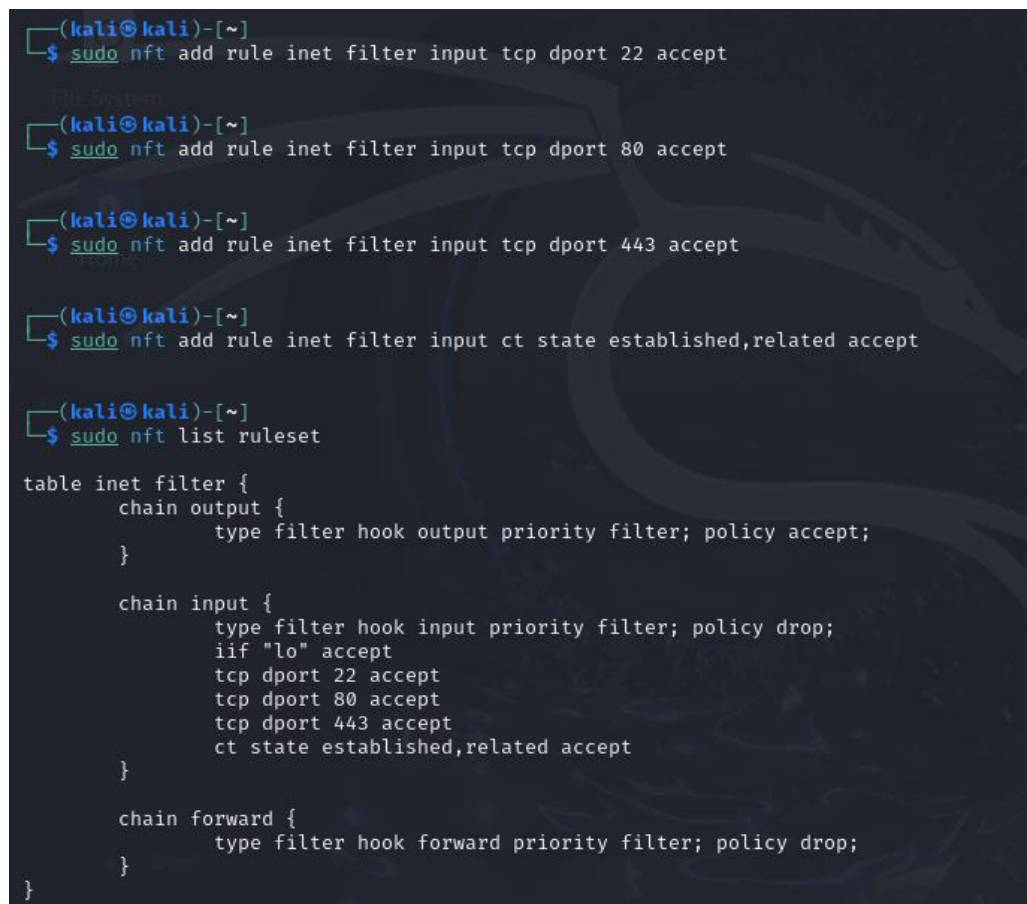
`"sudo nft add rule inet filter input tcp dport 443 accept"`

Te reguły pozwalają na ruch przychodzący na porty TCP 80 (HTTP) i 443 (HTTPS), co jest kluczowe dla działania aplikacji webowych. Dzięki tym regułom, system może przyjmować żądania od użytkowników oraz umożliwiać komunikację z serwisami internetowymi.

Akceptacja odpowiedzi na ruch wychodzący

`sudo nft add rule inet filter input ct state established,related accept`

Dodaję regułę, która akceptuje pakiety będące odpowiedzią na wcześniej nawiązane połączenia oraz te związane z nimi. Umożliwia to utrzymanie sesji użytkownika oraz prawidłowe działanie aplikacji, co jest kluczowe dla funkcjonalności systemu.



```
(kali@kali)-[~]
$ sudo nft add rule inet filter input tcp dport 22 accept

File System
(kali@kali)-[~]
$ sudo nft add rule inet filter input tcp dport 80 accept

(kali@kali)-[~]
$ sudo nft add rule inet filter input tcp dport 443 accept

(kali@kali)-[~]
$ sudo nft add rule inet filter input ct state established,related accept

(kali@kali)-[~]
$ sudo nft list ruleset

table inet filter {
    chain output {
        type filter hook output priority filter; policy accept;
    }

    chain input {
        type filter hook input priority filter; policy drop;
        iif "lo" accept
        tcp dport 22 accept
        tcp dport 80 accept
        tcp dport 443 accept
        ct state established,related accept
    }

    chain forward {
        type filter hook forward priority filter; policy drop;
    }
}
```

Ocena całości rulesetu

Dopasowanie do zasad bezpieczeństwa:

Polityka drop dla łańcuchów input i forward jest zgodna z najlepszymi praktykami w zakresie cyberbezpieczeństwa, ponieważ minimalizuje ryzyko nieautoryzowanego dostępu.

Zezwolenie na ruch z interfejsu loopback oraz akceptacja portów dla SSH, HTTP i HTTPS są dobrze przemyślane, co zapewnia niezbędną funkcjonalność.

Reguła dotycząca akceptacji odpowiedzi na istniejące połączenia jest kluczowa dla utrzymania komunikacji i sprawnego działania aplikacji.

Wynik skanowania skanerem podatności.

```
(kali㉿kali)-[~]
$ nmap -A localhost
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-28 07:48 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000062s latency).
Other addresses for localhost (not scanned): ::1
All 1000 scanned ports on localhost (127.0.0.1) are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.24 seconds
```

Dodajemy regułę chroniącą przed ataki typu "Brute-force" na porcie 22(SSH)

```
(kali㉿kali)-[~]
$ sudo nft add rule inet filter input ip protocol tcp tcp dport 22 ct state new limit rate 5/minute accept

(kali㉿kali)-[~]
$ sudo nft add rule inet filter input ip protocol tcp tcp dport 22 ct state new drop

(kali㉿kali)-[~]
$ sudo nft list ruleset
table inet filter {
    chain output {
        type filter hook output priority filter; policy accept;
    }

    chain input {
        type filter hook input priority filter; policy drop;
        iif "lo" accept
        tcp dport 22 accept
        tcp dport 80 accept
        tcp dport 443 accept
        ct state established,related accept
        ip protocol tcp tcp dport 22 ct state new limit rate 5/minute burst 5 packets accept
        ip protocol tcp tcp dport 22 ct state new drop
    }

    chain forward {
        type filter hook forward priority filter; policy drop;
    }
}
```

Pierwsza reguła

"sudo nft add rule inet filter input ip protocol tcp tcp dport 22 ct state new limit rate 5/minute accept"

Ta reguła zezwala na nowe połączenia TCP (protokół SSH) na porcie 22, ale tylko do 5 prób logowania na minutę z jednego adresu IP. Użycie **ct state new** oznacza, że reguła dotyczy tylko nowych połączeń. Jeśli adres IP przekroczy 5 prób w ciągu minuty, dalsze połączenia będą blokowane przez następną regułę.

Druga reguła

`"sudo nft add rule inet filter input ip protocol tcp tcp dport 22 ct state new drop"`

Ta reguła odrzuca wszystkie nowe połączenia TCP na porcie 22, jeśli zostały one przekroczone przez limit ustalony w pierwszej regule. **Działanie:** Po przekroczeniu limitu 5 prób w ciągu minuty, wszystkie kolejne próby logowania będą odrzucane.

Komenda na sprawdzenie logów

`„sudo journalctl -u nftables”`

```
(kali㉿kali)-[~]
$ sudo journalctl -u nftables
Sep 28 07:06:23 kali systemd[1]: Starting nftables.service - nftables ...
Sep 28 07:06:23 kali systemd[1]: Finished nftables.service - nftables.
```

Scenariusz po zidentyfikowaniu złośliwego adresu IP blokujemy go poprzez dodanie reguł w firewallu

Założmy, że adres IP, który chcemy zablokować, to **192.168.1.100**. Możesz użyć poniższej komendy, aby dodać regułę blokującą ten adres:

`"sudo nft add rule inet filter input ip saddr 192.168.1.100 drop"`

```
(kali㉿kali)-[~]
$ sudo nft add rule inet filter input ip saddr 192.168.1.100 drop

(kali㉿kali)-[~]
$ sudo nft list ruleset
table inet filter {
    chain output {
        type filter hook output priority filter; policy accept;
    }

    chain input {
        type filter hook input priority filter; policy drop;
        iif "lo" accept
        tcp dport 22 accept
        tcp dport 80 accept
        tcp dport 443 accept
        ct state established,related accept
        ip protocol tcp tcp dport 22 ct state new limit rate 5/minute burst 5 packets accept
        ip protocol tcp tcp dport 22 ct state new drop
        ip saddr 192.168.1.100 drop
    }

    chain forward {
        type filter hook forward priority filter; policy drop;
    }
}
```

Udoskonalanie reguł w firewallu

Ochrona przed atakami DDoS

Możesz dodać reguły limitujące ilość połączeń na poziomie IP, aby chronić się przed atakami DDoS. Na przykład, możesz ograniczyć liczbę nowych połączeń na minutę z jednego adresu IP.

```
(kali@kali)-[~]
$ sudo nft add rule inet filter input tcp dport 80 ct state new limit rate 10/minute burst 5 packets accept
(kali@kali)-[~]
$ sudo nft add rule inet filter input tcp dport 80 ct state new drop
(kali@kali)-[~]
$ sudo nft add rule inet filter input tcp dport 23 drop
```

Ochrona przed skanowaniem portów

Możesz dodać reguły, które będą odrzucać nieautoryzowane połączenia na porty, które nie są otwarte. Na przykład, jeśli nie używasz portu 23 (Telnet), możesz go zablokować.

Logowanie nieudanych prób

Możesz dodać reguły logujące nieudane próby połączeń, aby monitorować potencjalne ataki:

```
(kali@kali)-[~]
$ sudo nft add rule inet filter input tcp dport 22 ct state new counter log prefix SSH_Drop drop
(kali@kali)-[~]
$ sudo nft list ruleset
table inet filter {
  chain output {
    type filter hook output priority filter; policy accept;
  }
  chain input {
    type filter hook input priority filter; policy drop;
    iif "lo" accept
    tcp dport 22 accept
    tcp dport 80 accept
    tcp dport 443 accept
    ct state established,related accept
    ip protocol tcp tcp dport 22 ct state new limit rate 5/minute burst 5 packets accept
    ip protocol tcp tcp dport 22 ct state new drop
    ip saddr 192.168.1.100 drop
    tcp dport 80 ct state new limit rate 10/minute burst 5 packets accept
    tcp dport 80 ct state new drop
    tcp dport 23 drop
    tcp dport 22 ct state new counter packets 0 bytes 0 log prefix "SSH_Drop" drop
  }
  chain forward {
    type filter hook forward priority filter; policy drop;
  }
}
```

```
(kali@kali)-[~]
$ sudo nmap -sU localhost
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-28 09:55 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000030s latency).
Other addresses for localhost (not scanned): ::1
All 1000 scanned ports on localhost (127.0.0.1) are in ignored states.
Not shown: 1000 closed udp ports (port-unreach)

Nmap done: 1 IP address (1 host up) scanned in 0.15 seconds
(kali@kali)-[~]
$ sudo nmap -sX localhost
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-28 09:55 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000030s latency).
Other addresses for localhost (not scanned): ::1
All 1000 scanned ports on localhost (127.0.0.1) are in ignored states.
Not shown: 1000 closed tcp ports (reset)

Nmap done: 1 IP address (1 host up) scanned in 0.09 seconds
(kali@kali)-[~]
$ sudo nmap -Pn localhost
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-28 09:56 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000030s latency).
Other addresses for localhost (not scanned): ::1
All 1000 scanned ports on localhost (127.0.0.1) are in ignored states.
Not shown: 1000 closed tcp ports (reset)

Nmap done: 1 IP address (1 host up) scanned in 0.08 seconds
```