

NSWI184 – Řízení počítačových sítí

Cviko deváté

Kateřina Kubecová, Ondřej Zajíček

4. prosince 2025

Cviko deváté

NF tables

NF tables

- ▶ Lze vytvářet tabulky typů ip, ip6, inet, arp, bridge, netdev
- ▶ Tabulky obsahují řetízky (chain) typů filter, route nebo nat
 - hook určuje, ve které fázi zpracování packetu se chain použije (prerouting, input, forward, output...) různé typy řetízků mají různé možnosti
 - priority určuje pořadí řetízků (není-li jasné)
 - policy udávají defaultní verdikt
- ▶ Řetízky obsahují jednotlivá pravidla, která se vyhodnocují dokud nedospějeme k nějakému výsledku (accept/drop)
 - Existují i pravidla jump, goto, continue, return a queue pro navigaci mezi pravidly a řetízky
- ▶ Proměnné definujeme jako define A = 8, referencujeme \$A
- ▶ https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/8/html/configuring_and_managing_networking/getting-started-with-nftables-configuring-and-managing-networking#

NF tables

- ▶ Nf tables lze skládat přes příkazovou řádku, nebo spustit skript
 - Skript se spouští `nft -f /path/to/script.conf`
 - Interaktivní mód `nft -i` (není pro práci s nft nutný)
- ▶ Současný stav pravidel zobrazíte jako `nft list ruleset` (v interaktivním módu pouze `list ruleset`)
- ▶ Do pravidel lze vkládat counter pro měření provozu.
 - např. `meta iifname "zly_interface"counter drop;`
 - `list ruleset` zobrazí `iifname "zly_interface"counter packets 8794 bytes 13190184 drop`
- ▶ Souhrn pravidel najdete zde: https://wiki.nftables.org/wiki-nftables/index.php/Quick_reference-nftables_in_10_minutes

NFT skript

```
#!/usr/sbin/nft -f
flush ruleset # Nejprve uvolníme stará pravidla
define HODNY_INTERFACE = "B"
table inet my_filter { # tabulka typu inet - bereme IPv4 a IPv6 packety
    chain my_input {
        type filter hook input priority filter; # budeme filtrovat a to na vstup
        # Bereme všechno z již přijatých nebo inicializovaných spojení.
        # Pozor, co akceptujeme tady, to už neprojde do dalších pravidel.
        ct state { established, related } accept;
        ct state invalid drop;
        meta iifname $HODNY_INTERFACE counter accept;
        meta iifname "edge6" accept; # iifname = input interface name
        tcp dport { 22 } accept; # Accept SSH
        # reject na všechno, co se nevešlo do předchzích pravidel
        reject with icmpx type port-unreachable;
    }
}
```

NF tables

- ▶ Spusťte soubor `two_simple_devices.sh` a ověřte, že pingnete z A na B
- ▶ Na stroji B vytvořte NFT skript, který přijme pakety od A
- ▶ Povolte ping a ostatní icmp pakety
 - `meta l4proto { icmp, ipv6-icmp } counter accept;`
- ▶ Pomocí příkazu `limit rate 10/minute` omezte množství přijímaných icmp paketů
 - `meta l4proto { icmp, ipv6-icmp } limit rate 10/minute accept`
- ▶ Nezapomeňte zbytek dropnout.
- ▶ Přijímejte všechny pakety z establishovaných spojení vyjma icmp paketů, kterým nechte rate limit
 - Ujistěte se, že stále prochází jen některé pingy

NAT

- ▶ Postavte za sebe tři stroje - klienta, router a server a přiďte jim čtyřkové adresy
- ▶ Klientovi a routeru dejte všechny routy, ale server bude mít pouze routu k routeru
 - `ip route add <čtyřková adresa> via inet6 fe80::... dev C`
- ▶ Ověřte, že pingnete z klienta i serveru na router, ale z klienta na server už ne
- ▶ Vytvořte nft konfigurák pro router s postrouting chainem
- ▶ masquerade dynamicky mění source adresu za adresu routeru, na kterém běží

```
chain postrouting {  
    type nat hook postrouting priority srcnat; policy accept;  
    ip saddr <adresa klienta> counter masquerade; }
```
- ▶ Teď by měl fungovat ping z klienta na server
- ▶ Ověřte v tcpdumpu, že ping od klienta chodí pod adresou routeru

NAT vs Firewall

- ▶ Stále by nemělo být možné pingnout ze serveru na klienta.
- ▶ Přidejte serveru routu na klienta
- ▶ Teď už se server na klienta dopingá. Samotný NAT sice schoval klientovu adresu, ale nechal ji přístupnou.
- ▶ Přidejte tabulce routeru chain s forward hookem (`type filter hook forward priority filter;`)
- ▶ Povolte pouze packety z interface klienta a z established a related spojení
- ▶ Ověřte, že opět pingnete pouze z klienta na server a nikoliv naopak (přestože server routu ke klientovi zná)

NAT64

- ▶ Linux: kernelový modul Jool (stále nezačleněn)
- ▶ Jool potřebuje sudo i uvnitř eduhawka → fix in progress, na špachtli hotfixed
- ▶ PLAT: `jool instance add --netfilter --pool6 64:ff9b::/96`
- ▶ Základní jool debug uvnitř netns: `jool stats ...`
- ▶ <https://github.com/marenamat/Jool/tree/mq-allow-usersns>
- ▶ Nechcete-li dávat eduhawku rootovská práva, potřebujete jool z branchy `mq-allow-usersns`

NAT64

- ▶ Postavte do řady tři stroje - klienta, plat a internet
 - ▶ internetu dejte pouze čtyřkovou adresu `ip a add 252.0.0.1/24 dev plat`
 - ▶ platu bude mít
 - čtyřkovou adresu směrem na internet a šestkovou na klienta
 - default routu `ip r add default via 252.0.0.1 dev A`
 - jool instance `add --netfilter --pool6 64:ff9b::/96`
 - ▶ klient bude mít pouze šestkovou adresu, routu k platu a routu `64:ff9b::/96` jdoucí přes plat
 - ▶ Ověřte, že se z klienta dopingáte na internet
 - adresu internetu sestavíte jako `64:ff9b::` plus adresa internetu `252.0.0.1`
- `ping 64:ff9b::252.0.0.1`
- ▶ Zobrazte si `tcpdump` pingů na stroji plat

NAT64

- ▶ CLAT (dst): `jool_siit instance add --netfilter --pool6 64:ff9b::/96`
- ▶ CLAT (src): `jool_siit eamt add 3fff:0:cc:64::/120 10.0.0.0/24`
bezestavově mapujeme privátní IPv4 rozsah (zde 10.0.0.0/24)
do IPv6 rozsahu (zde 3fff:0:cc:64::/120)
- ▶ Ověřte, že z IPv4-only sítě jsou dosažitelné IPv4 adresy za IPv6-only sítí.
- ▶ paket před CLAT: `src 10.0.0.42 → dst 192.0.2.15`
- ▶ packet za CLAT: `src 3fff:0:cc:64::2b → dst 64:ff9b::c000:20f`
- ▶ primárně se překládá IPv4 podle EAMT, sekundárně do `--pool6`
- ▶ Funguje vám IPv4 přímo na CLAT zařízení? *hint: nemělo by*

NAT64

- ▶ Problém: Jool překládá pouze příchozí pakety v prefilter fázi, takže nezachytí pakety, které vzniknou u nás.
- ▶ Špinavý trik: pořídíme si veth, do kterého nasměrujeme default routu
→ náš provoz nám přijde zpátky a Jool ho může přeložit.

```
ip link add locxlatin type veth peer name locxlatout
ip link set locxlatin up
ip link set locxlatout up
```

```
ip a add 192.0.0.8/32 dev locxlatin
ip a add fe80::64/64 dev locxlatout
ip -4 r add default via inet6 fe80::64 dev locxlatin
```

```
jool_siit eamt add 3fff:0:cc:64::eff 192.0.0.8
```