SSH tunelování

Datum zpracování: 04. 12. 2023

Zpracovali: Kevin Daněk, Crazy Wolf Raspberry 0.25L

Zadání

1. Pomocí programu SSH vytvořte tunel mezi dvěma virtuálními stroji bez použití lokálních síťových prvků.
2. Pokud to bude nutné, využijte třetí stroj s veřejnou IP adresou.
3. Pomocí programu **iperf3** změřte rychlost spojení.
4. V průběhu práce pořizujte screenshoty a záznamy použitých příkazů; použijte je v elaborátu a okomentujte postup.

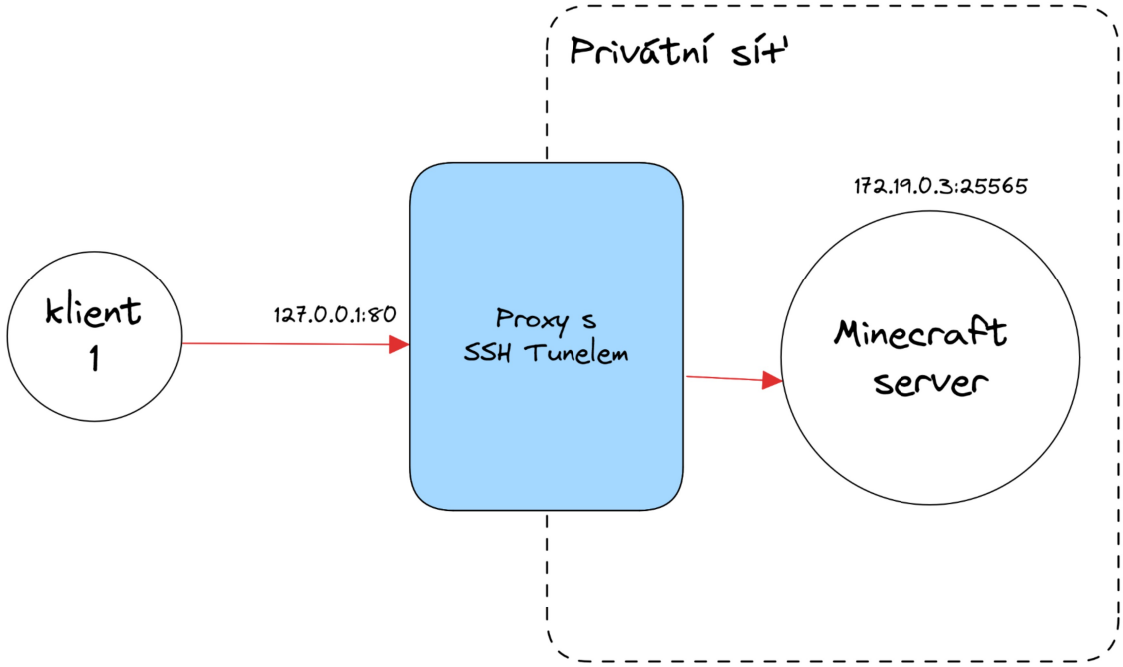
Úlohu zpracovávejte ve 3 - 5 členných týmech.

Elaborát zpracujte do šablony v záhlaví kurzu, odevzdávejte ve formátu PDF.

Do abecedně seřazeného seznamu řešitelů na úvodním listu uveďte reálné složení týmu!

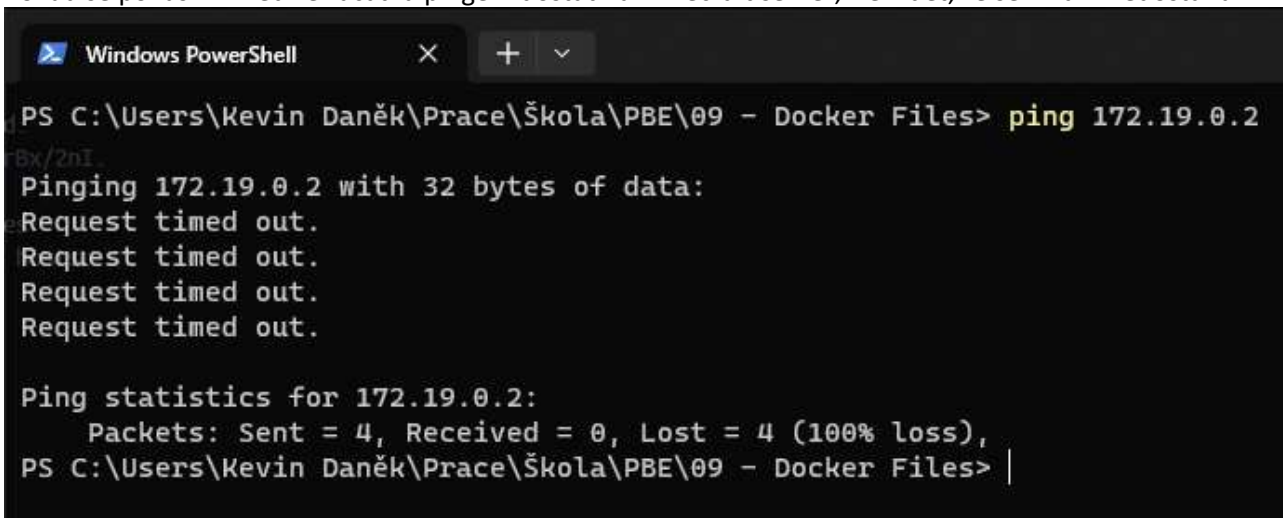
Příprava prostředí

Z důvodu technického stavu učebny A305, kde každou chvíli nějaká část vybavení nefunguje a ve které by cvičení měla správně probíhat, jsem se rozhodl, stejně jako u předchozích dvou cvičení, realizovat ho mimo ní. K simulaci sítě a klientů využiji platformu Docker. Aby bylo cvičení trochu zážívnější a hlavně šlo výsledek tunelování vidět ještě jasněji než z výstupu v konzoli, rozhodl jsem se SSH tunelování uplatnit na následující problém:



Máme situaci, kde `klient_1` (v tomto případě hostující počítač, na kterém běží docker engine) se chce připojit na minecraft server, který běží v privátní síti 172.19.0.0/16. K dispozici máme pouze běžící stroj s Ubuntu, který se v tomto příkladě bude chovat jako gateway do sítě. Cílem je se z hostitelského stroje (klient_1) připojit na minecraft server v jiné síti.

Pokud se pokusím hned ze začátku pingem dostat na minecraft server, lze vidět, že se nikam nedostanu:



Vytvoření SSH tunelu

Do Minecraft server kontejneru jsem doinstaloval openssh-server, který mi dovolí se do něho vzdáleně připojit a naslouchat na portu 22. Protože se nespustil automaticky, ještě jsem ho následně přes příkaz service spustil.

```
root@23dbed0848e7:/data# apt install -y openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
```

```
root@23dbed0848e7:/etc/init.d# service ./ssh start
 * Starting OpenBSD Secure Shell server sshd
```

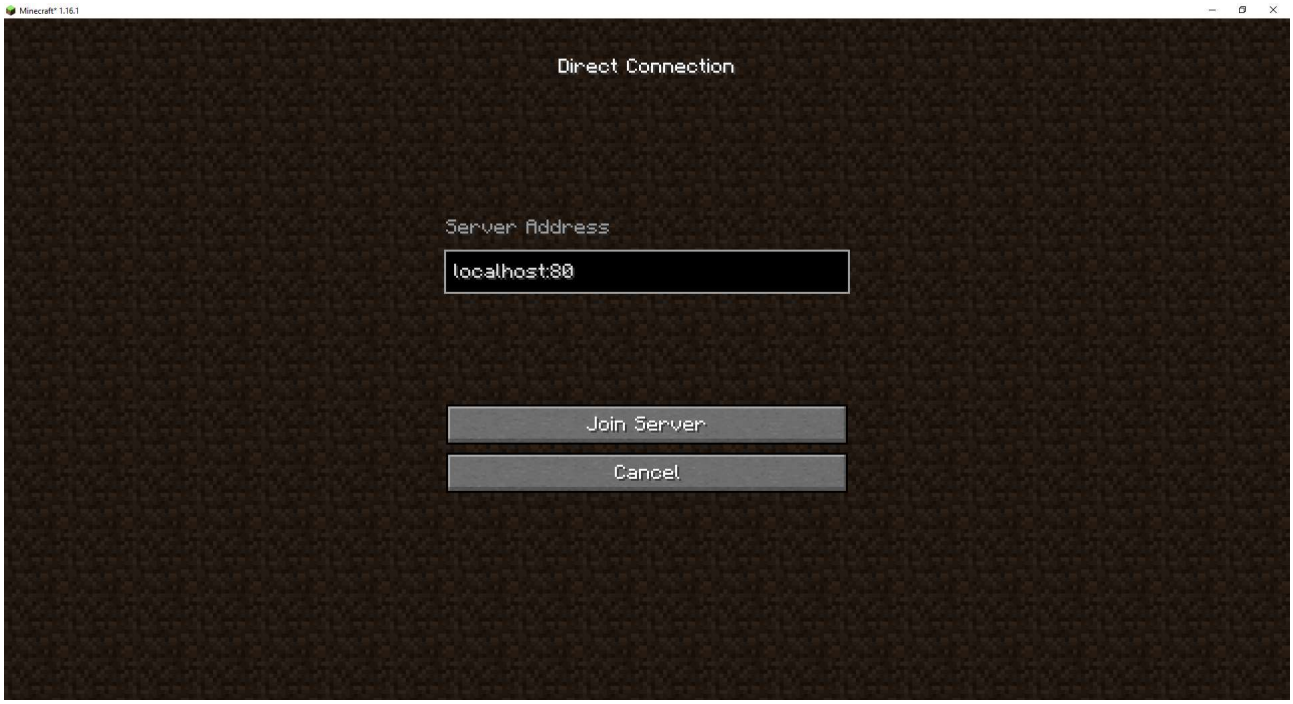
Následně jsem na gateway kontejneru vytvořil RSA klíče, abych se mohl připojit na Minecraft server přes ssh bez nutnosti zadávat heslo. Postup je analogický ke cvičení 6, tudíž ho tady nebudu nějak moc rozepisovat.

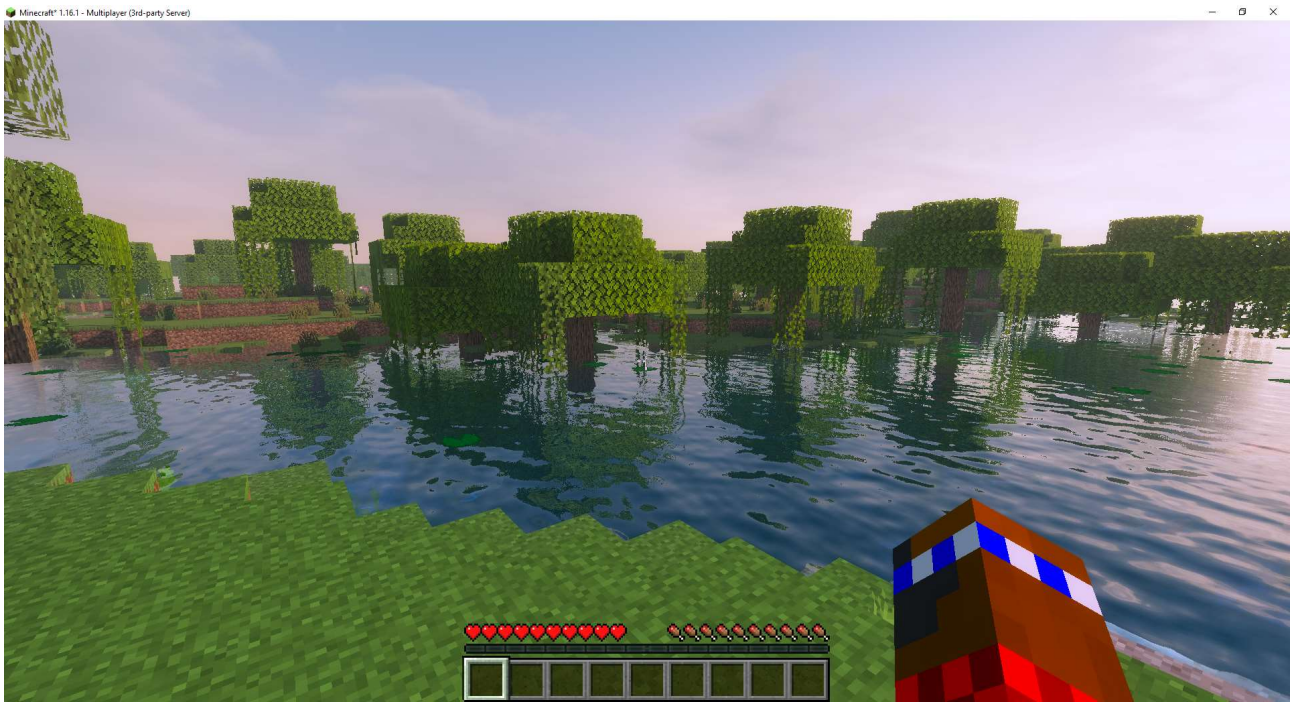
```
root@312ba5c61d5b:/# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:WmSnW3zoPk1CAUYZFpnRVz085aOvz1MDXyT8MRuV2/w root@312ba5c61d5b
The key's randomart image is:
+---[RSA 3072]----+
|       .OX   oo.*|
|       o+ o . oXo|
|        o .o   *@|
|       o +.. ..o*|
|        S.+ ..o o|
|       o +... .oE|
|      . . .+   .o|
|         .. . o. |
|          .. ..o.|
+----[SHA256]-----+
```

No a následně již zbývalo vytvoření samotného tunelu. Na gateway kontejneru jsem na všech dostupných rozhraních přesměroval port 80 na adresu Minecraft serveru ve vnitřní síti (172.19.0.2) a port, na kterém běží, tj. 25565.

```
root@312ba5c61d5b:/# ssh -v -L 0.0.0.0:80:172.19.0.2:25565 root@172.19.0.2
```

Když jsem pak následně na hostitelském počítači, ze kterého byl Minecraft server nepřístupný, zkusil adresu gateway kontejneru (který měl otevřený port 80 na místní smyčce), tak jsem se byl schopný úspěšně připojit do hry. SSH Tunel tak musí fungovat. Po jeho vypnutí se již na server nelze připojit.





Měření rychlosti pomocí iperf3

Pro měření rychlosti jsem do kontejneru s Minecraft serverem doinstaloval iperf3 a na gateway kontejneru vytvořil SSH tunel pro jeho úspěšné použití.

```
root@dee08367d880:/data# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

```
root@312ba5c61d5b:/# ssh -L 0.0.0.0:80:172.19.0.2:5201 root@172.19.0.2
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.133.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Tue Dec  5 00:19:17 2023 from 172.19.0.3
root@dee08367d880:~#
```

Následně, protože na WSL mi iperf3 nefungoval, vytvořil jsem další Ubuntu kontejner, který operoval na hostitelské síti, tudíž používal stejné rozhraní jako hostitelský počítač a prakticky vzato tak i fungoval.

```
PS C:\Users\Kevin Daněk> docker run -dti --network host --entrypoint /bin/bash --name test_client ubuntu
8cdcd13088ddd82546770d5179a630f46bf0721e909eef173b809c4383ffb684
```

Na následující stránce jsou výsledky měření. Jak lze vidět, klienta jsem poslal na 127.0.0.1:80, a na serveru přišlo spojení z 172.19.0.2:5201, tudíž tunel opět funguje.

Měření rychlosti z pohledu klienta

```
root@DESKTOP-N0QO4FQ:/# iperf3 -c 127.0.0.1 -p 80
Connecting to host 127.0.0.1, port 80
[  5] local 127.0.0.1 port 35090 connected to 127.0.0.1 port 80
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec   224 MBytes  1.88 Gbits/sec    0    880 KBytes
[  5]   1.00-2.00   sec   216 MBytes  1.81 Gbits/sec    0    880 KBytes
[  5]   2.00-3.00   sec   216 MBytes  1.81 Gbits/sec    0    880 KBytes
[  5]   3.00-4.00   sec   215 MBytes  1.80 Gbits/sec    0    880 KBytes
[  5]   4.00-5.00   sec   212 MBytes  1.78 Gbits/sec    0    880 KBytes
[  5]   5.00-6.00   sec   218 MBytes  1.82 Gbits/sec    0    880 KBytes
[  5]   6.00-7.00   sec   215 MBytes  1.80 Gbits/sec    0    880 KBytes
[  5]   7.00-8.00   sec   218 MBytes  1.82 Gbits/sec    0    880 KBytes
[  5]   8.00-9.00   sec   216 MBytes  1.81 Gbits/sec    0    880 KBytes
[  5]   9.00-10.00  sec   212 MBytes  1.78 Gbits/sec    0    880 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  2.11 GBytes  1.81 Gbits/sec    0             sender
[  5]   0.00-10.06  sec  2.11 GBytes  1.80 Gbits/sec                  receiver

iperf Done.
```

Měření rychlosti z pohledu serveru

```
Accepted connection from 172.19.0.2, port 34418
[  5] local 172.19.0.2 port 5201 connected to 172.19.0.2 port 34420
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-1.00   sec   207 MBytes  1.73 Gbits/sec
[  5]   1.00-2.00   sec   216 MBytes  1.82 Gbits/sec
[  5]   2.00-3.00   sec   216 MBytes  1.81 Gbits/sec
[  5]   3.00-4.00   sec   216 MBytes  1.81 Gbits/sec
[  5]   4.00-5.00   sec   213 MBytes  1.78 Gbits/sec
[  5]   5.00-6.00   sec   217 MBytes  1.82 Gbits/sec
[  5]   6.00-7.00   sec   215 MBytes  1.81 Gbits/sec
[  5]   7.00-8.00   sec   217 MBytes  1.82 Gbits/sec
[  5]   8.00-9.00   sec   217 MBytes  1.82 Gbits/sec
[  5]   9.00-10.00  sec   213 MBytes  1.78 Gbits/sec
[  5]  10.00-10.06  sec  12.3 MBytes  1.80 Gbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-10.06  sec  2.11 GBytes  1.80 Gbits/sec                  receiver
```

Závěr

Stejně jako u předchozích cvičení, velmi mě bavilo vymýšlet, jak zadání realizovat v Dockeru. Extra s mým dodatkem do zadání, kdy jsem ho uplatnil na (ne)praktické situaci. SSH tunelování je super alternativa např. k proxy serverům typu NGINX a vidím, jak může být využíváno jako jednoduchý systém pro VPN.