Linux: Tips & Tricks

niet ideale semi permanente oplossingen

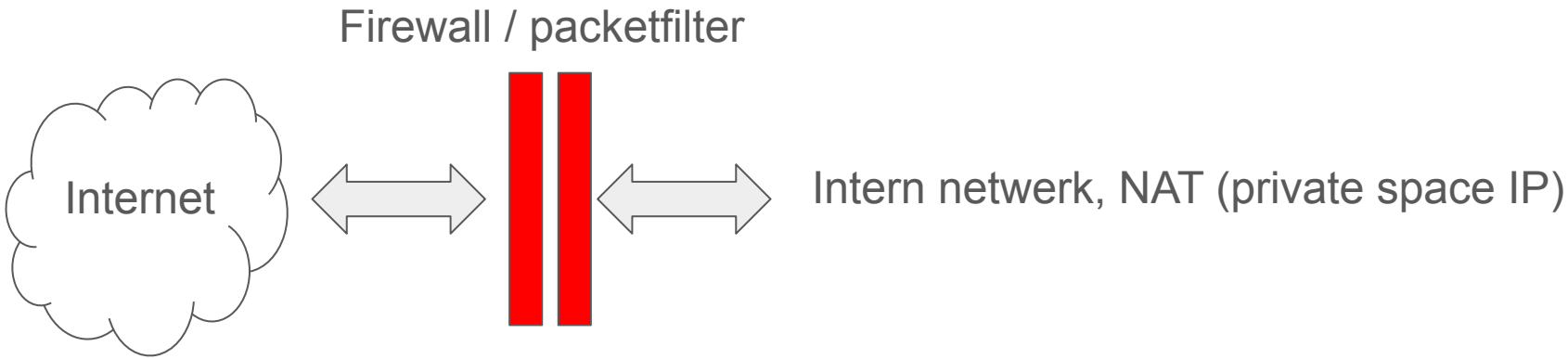
Enkele cases

- 1) Toegang naar private netwerk + 2) “meekijken” met een lokale desktop
- Verkeer doorsturen naar container
- ssh via een jumphost
- “port share” met openvpn
- Verkeer valt “dood” door een tunnel (VPN)
- Geen VARS in bash scripting
- Uitkomst van commando’s vergelijken
- Geen CapsLock
- Tot slot: 10 handige commando’s

Toegang naar private netwerk

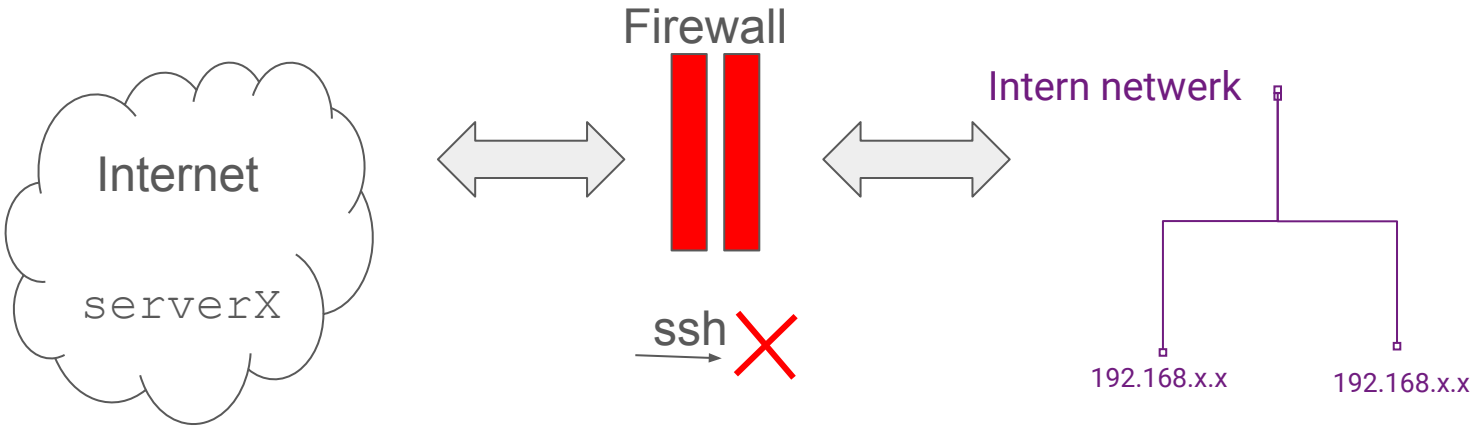
Situatie:

Soms heb je systemen achter een firewall waarbij je geen invloed hebt op de firewall zelf (niet gewenst!)



Vaak wel verkeer van Intern naar buiten toegestaan. En als er uitgaand verkeer wordt geblocked dan meestal niet port 80 (en 443). Om je systeem in het interne netwerk te beheren doe:

Maak een ssh tunnel van binnenuit met -R (“Remote forwarding) :




```
ssh -R 2000:127.0.0.1:22 -p 80 user1@serverX
```

ssh automatisch inloggen

Nodig voor ssh tunneling: asymmetrische versleuteling (public/private key-pair):

op de interne host: `piadmin@raspberrypi:~ $ ssh-keygen`

(geen passphrase opgeven)

Zet public key (default `id_rsa.pub`) in:

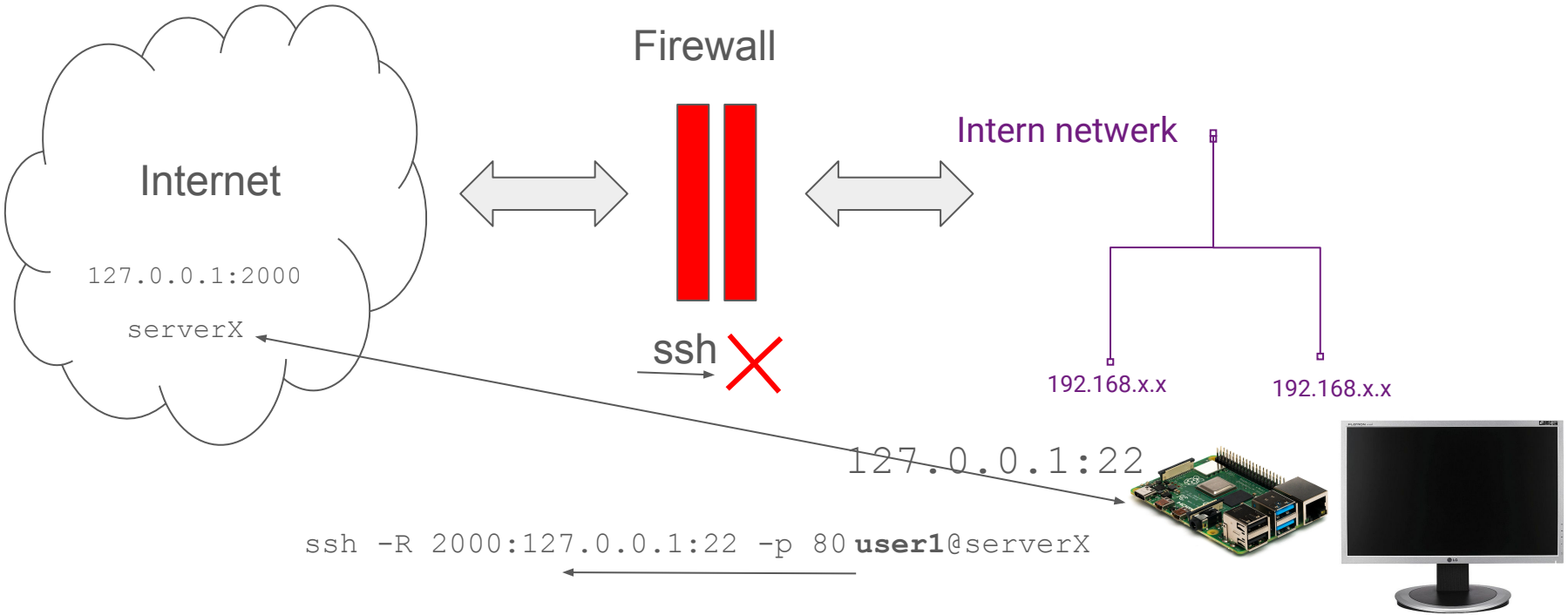
server1:/home/user1/.ssh/authorized_keys

(zorg dat de key 1 regel is!)

```
# chown -R user1:user1 /home/user1/.ssh/
# chmod 600 /home/user1/.ssh/authorized_keys
```

Dan val je als user1 zonder wachtwoord binnen (pubkey authentication moet enabled zijn op de server (/etc/ssh/sshd_config)

Praktijk:


Firewall
Internet
127.0.0.1:2000
serverX
ssh
Intern netwerk
192.168.x.x
192.168.x.x
127.0.0.1:22
ssh -R 2000:127.0.0.1:22 -p 80 user1@serverX

root@raspberrypi:~# cat /etc/systemd/system/**sshtunnel.service**

```
[Unit]

Description=Setup an ssh tunnel between serverX and raspberry pi

After=network.target

[Service]

ExecStart=/usr/bin/ssh -NT -o ServerAliveInterval=60 -o \
ExitOnForwardFailure=yes -R 2000:127.0.0.1:22 -p 80user1@x.x.x.x &

# Restart every >2 seconds to avoid StartLimitInterval failure

RestartSec=15

Restart=always

User=pi

Group=pi

[Install]

WantedBy=multi-user.target
```

2) Extra uitdaging: “meekijken” met de lokale desktop

Aan de raspberry pi hangt een beeldscherm waarop een en ander op wordt getoond (middels een url).

Stappen:

- zorg dat de user (“pi”) een autologin heeft (bv. met `raspi-config`)
- voeg toe aan `~pi/.profile`:
  - `/home/pi/kiosk.sh &`

```
root@raspberrypi:~# cat /home/pi/kiosk.sh
```

#!/bin/bash

xset s noblank

xset s off

unclutter -idle 0.5 -root &

sed -i 's/"exited_cleanly":false/"exited_cleanly":true/' /home/pi/.config/chromium/Default/Preferences

sed -i 's/"exit_type":"Crashed"/"exit_type":"Normal"/' /home/pi/.config/chromium/Default/Preferences

/usr/bin/chromium-browser --noerrdialogs --disable-infobars --disable-crashpad --kiosk https://url-here.. --start-fullscreen &

Maar hoe kijk je nu van afstand mee op het scherm?

M.a.w.: hoe weet je of het scherm nog wel de output van de url toont?

Oplossing: gebruik **maim** om per interval een “screenshot” te maken en die screenshot beschikbaar maken (bv. op server1). Interval is bv. om de minuut..:

In root cron op de raspberry pi:

```
*/2 * * * *  /usr/bin/sudo -u pi /usr/bin/maim -x :0.0 /tmp/desktop.jpg
```

Zet dit image telkens op de webserver (met scp)

Verkeer doorsturen naar container

Met nagios kun je systemen goed monitoren, zeker als je de nrpe plugin gebruikt.

nrpe vereist een actieve connectie van monitoring server -> nrpe daemon op de te monitoren hosts (default port 5666).

Maar wat als je (meerdere) containers (IP: c.c.c.c) met nrpe wilt monitoren?

Oplossingen:

- Beste: bv. in docker: expose en publish port naar public IP van de host
- Of meer algemeen:
  - iptables -t nat -A PREROUTING -i x.x.x.x -p tcp --dport 5667 -j DNAT --to c1.c1.c1.c1:5666
  - iptables -t nat -A PREROUTING -i x.x.x.x -p tcp --dport 5668 -j DNAT --to c2.c2.c2.c2:5666
- Maar kan ook met **redir**:
  - redir --lport=5667 --cport=5666 --caddr=c1.c1.c1.c1
  - redir --lport=5668 --cport=5666 --caddr=c2.c2.c2.c2

ssh proxy via een jumphost (..)

```
server:~$ rsync -av -e ssh -o "ProxyCommand ssh -W %h:%p jumphost"\
          /var/www/html/ 192.168.1.10:/backup/webdata
```

port share met openvpn

Situatie:

Je wilt graag op je eigen server je VPN op port 80 runnen omdat deze poort veelal open staat in firewalls. Maar je hebt op die server ook een webserver runnen en die wil ook graag op port 80 luisteren…

En twee processen op dezelfde port kan niet.

Oplossing als je OpenVPN gebruikt: laat apache op port 81 luisteren en stuur http-verkeer op port 80 door naar port 81.

Dit config je in OpenVPN (wat http-verkeer kan herkennen):

```
port-share x.x.x.x 81
```

(wel lastig met letsencrypt!)

Verkeer valt “dood” door een tunnel (VPN)

Tunnels pakken tcp/ip pakketten in (*encapsulation*) en voegen extra header data toe. Vaak zoveel dat je pakket groter wordt dan de MTU waardoor fragmentatie nodig is.

Dit kan ervoor zorgen dat je TCP/IP verkeer helemaal niet meer doorkomt. Geen debug tool kan dit laten zien: er gebeurd gewoon niets meer. Is dat het geval dan kan het helpen je MTU te verlagen:

```
ip link set mtu 1200 dev ppp0
```

Zie ook: https://networkengineering.stackexchange.com/questions/74571/help-me-understand-why-to-decrease-the-mtu-size :

Gebruik geen variabelen met hoofdletters in bash scripts

Wat gaat hier mis (als je niks invoert)?:

```
#!/bin/bash

SUCCESS=1

while [ $SUCCESS -eq 1 ]

do

  read PATH

  if [ -z $PATH ]

  then

      PATH=$( pwd )

  fi

  cd $PATH

  SUCCESS=$?

done

ls
```

Zie ook: https://stackoverflow.com/questions/66841020/bash-script-fails-with-commands

Uitkomst van commando’s vergelijken

We kennen diff: vergelijk de inhoud van files.

Maar kun je ook de output van command’s vergelijken?

Gebruik “process substitution” om de output van commando’s als een FILE (“named pipe”) te zien:

diff <(cmd1) <(cmd2)

Geen CapsLock

`/usr/bin/setxkbmap -option ctrl:nocaps` # in ~/.bashrc of /etc/bash/bashrc

Kan ook zo (alleen) system-wide:

`XKBOPTIONS="ctrl:nocaps"` in /etc/default/keyboard (console-setup package nodig)

Tot slot: 10 handige commando’s

```
 1.  sudo chattr +i file                  # immutable maken
 2.  cat -vet “file”                      # “onderwater” kijken
 3.  grep -Ril “pattern” “dir” (of .)     # toon files die “pattern” bevatten
 4.  ip r g “IP”                          # via welk interface ga je?
 5.  lsof/strace -p “pid”                 # wat heeft een proces “vast” / wat doet het
 6.  nmap -sP network                     # scan je eigen netwerk
 7.  Gebruik van <Ctrl>-r in bash         # handige history in bash
 8.  Gebruik van !$ en !! in bash         # laatste argument, laatste commando
 9.  netstat -ntulp                       # kijk of iets luistert
10.  vi(ew)                               # snel editen, viewen

Extra’s:

11.  Kijk ook eens naar krachtige history met atuin (dank Bèr!)
12.  En naar nnn, een terminal file manager (Dank Ben!)
```