FIREWALLS 2 IPTABLES - PFSENSE

LES 5 2019.10.09

NETFILTER - IPTABLES

This firewall is also known as a stateful packet filter

There are five predefined chains:

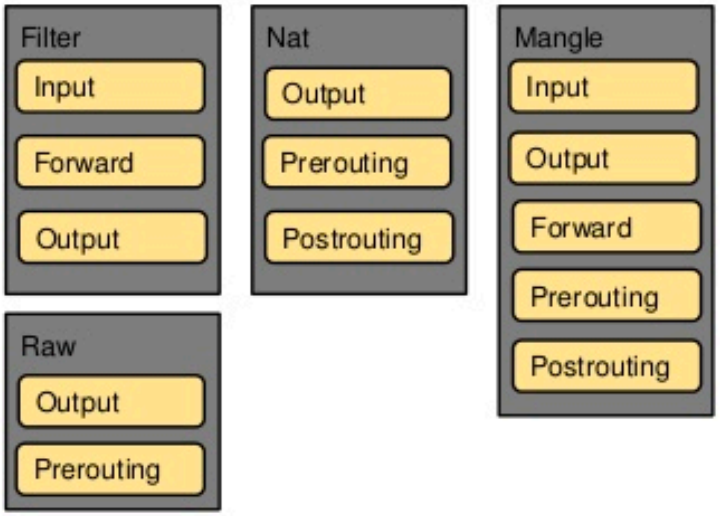
- INPUT: used for incoming packets before routing.
- OUTPUT: used for packets coming into the box itself.
- FORWARD: used for packets being routed through the box.
- PREROUTING: used for locally-generated packets before routing.
- POSTROUTING: used for packets as they about to leave iptables.

Tables overview

Filter is a default table. So, if you don't define you own table, you'll be using filter table.

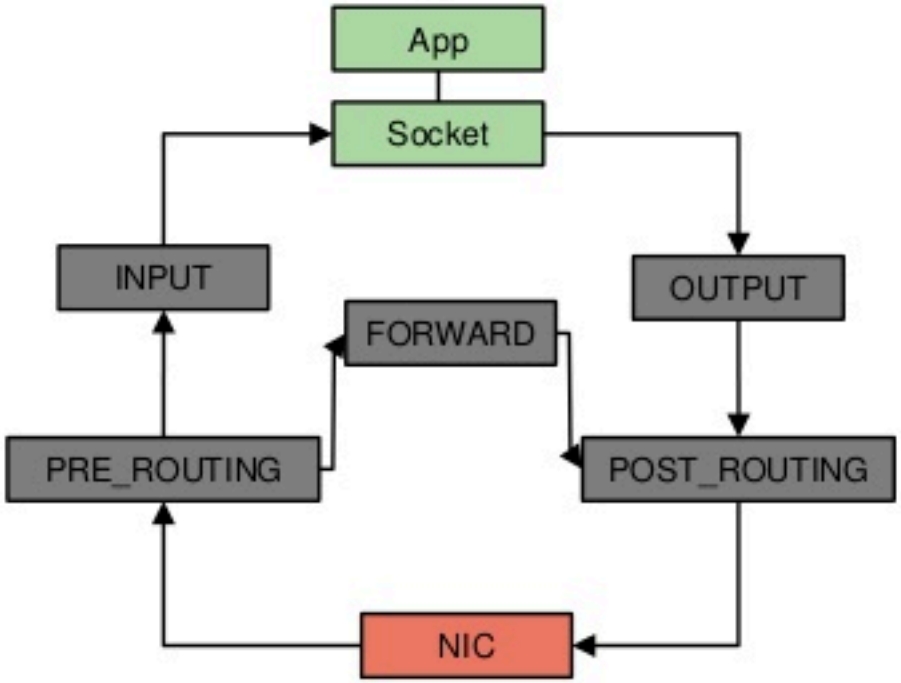
Each table has a number of predefined chains inside.

You can create your own chain.



Tables↓/Chains→	PREROUTING	INPUT	FORWARD	OUTPUT	POSTROUTING
(routing decision)				✓	
raw	✓			✓	
(connection tracking enabled)	✓			✓	
mangle	✓	✓	✓	✓	✓
nat (DNAT)	✓			✓	
(routing decision)	✓			✓	
filter		✓	✓	✓	
security		✓	✓	✓	
nat (SNAT)		✓			✓

Netfilter hooks stages



The Filter Table

The filter table is one of the most widely used tables in `iptables`. The `filter` table is used to make decisions about whether to let a packet continue to its intended destination or to deny its request. In firewall parlance, this is known as "filtering" packets. This table provides the bulk of functionality that people think of when discussing firewalls.

The NAT Table

The `nat` table is used to implement network address translation rules. As packets enter the network stack, rules in this table will determine whether and how to modify the packet's source or destination addresses in order to impact the way that the packet and any response traffic are routed. This is often used to route packets to networks when direct access is not possible.

The Mangle Table

The `mangle` table is used to alter the IP headers of the packet in various ways. For instance, you can adjust the TTL (Time to Live) value of a packet, either lengthening or shortening the number of valid network hops the packet can sustain. Other IP headers can be altered in similar ways.

This table can also place an internal kernel "mark" on the packet for further processing in other tables and by other networking tools. This mark does not touch the actual packet, but adds the mark to the kernel's representation of the packet.

The Raw Table

The `iptables` firewall is stateful, meaning that packets are evaluated in regards to their relation to previous packets. The connection tracking features built on top of the `netfilter` framework allow `iptables` to view packets as part of an ongoing connection or session instead of as a stream of discrete, unrelated packets. The connection tracking logic is usually applied very soon after the packet hits the network interface.

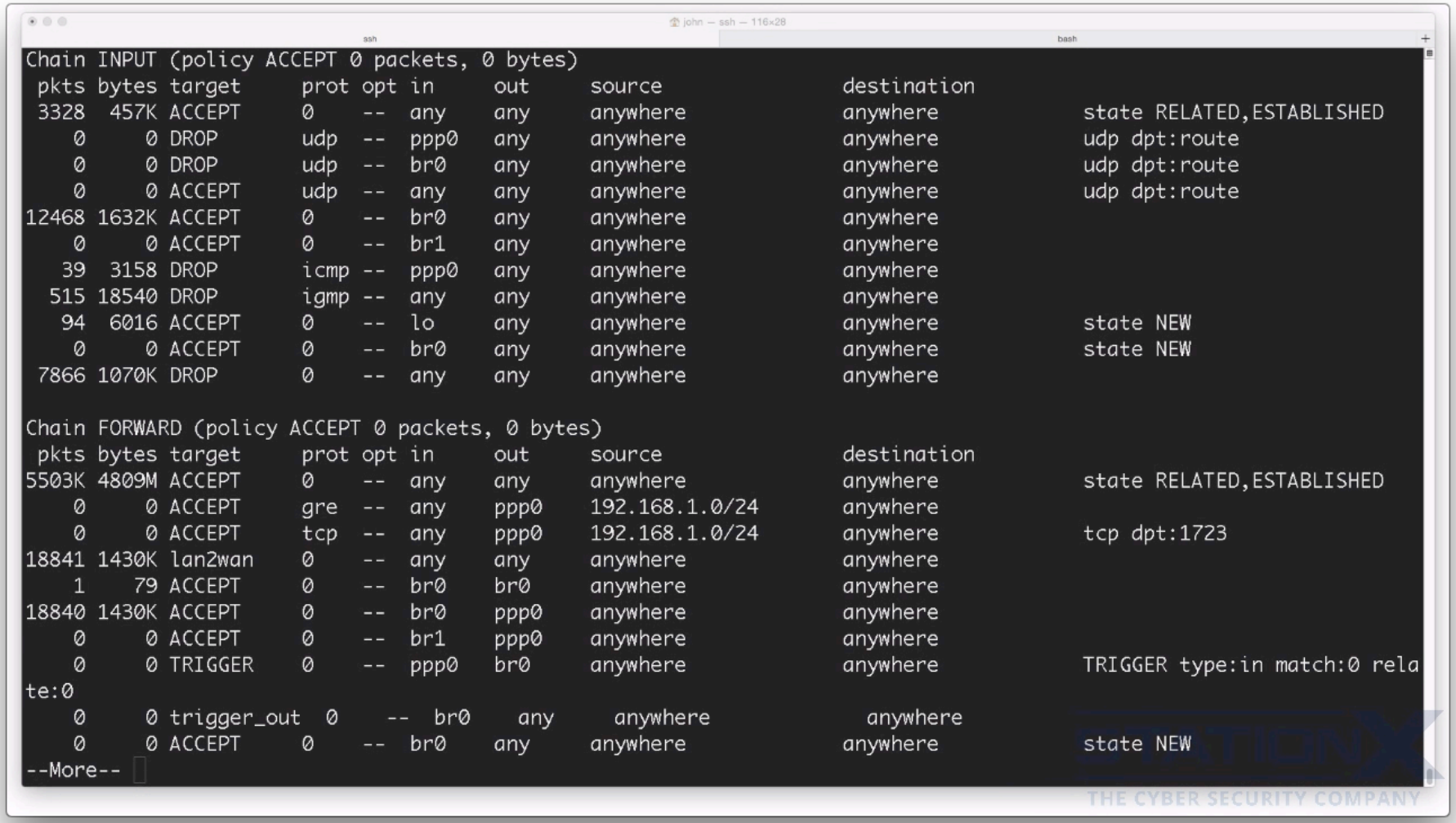
The `raw` table has a very narrowly defined function. Its only purpose is to provide a mechanism for marking packets in order to opt-out of connection tracking.

Available States

Connections tracked by the connection tracking system will be in one of the following states:

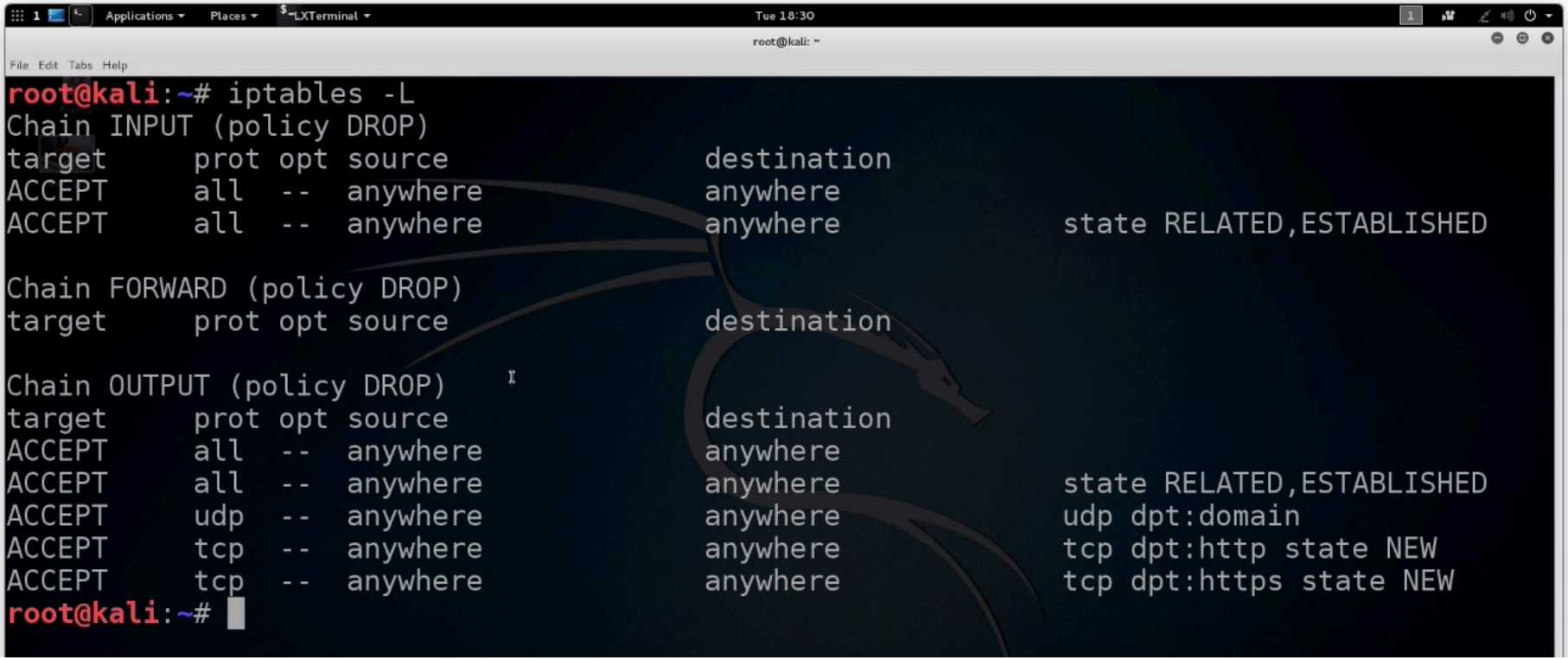
- `NEW`: When a packet arrives that is not associated with an existing connection, but is not invalid as a first packet, a new connection will be added to the system with this label. This happens for both connection-aware protocols like TCP and for connectionless protocols like UDP.
- `ESTABLISHED`: A connection is changed from `NEW` to `ESTABLISHED` when it receives a valid response in the opposite direction. For TCP connections, this means a `SYN/ACK` and for UDP and ICMP traffic, this means a response where source and destination of the original packet are switched.
- `RELATED`: Packets that are not part of an existing connection, but are associated with a connection already in the system are labeled `RELATED`. This could mean a helper connection, as is the case with FTP data transmission connections, or it could be ICMP responses to connection attempts by other protocols.
- `INVALID`: Packets can be marked `INVALID` if they are not associated with an existing connection and aren't appropriate for opening a new connection, if they cannot be identified, or if they aren't routable among other reasons.
- `UNTRACKED`: Packets can be marked as `UNTRACKED` if they've been targeted in a `raw` table chain to bypass tracking.
- `SNAT`: A virtual state set when the source address has been altered by NAT operations. This is used by the connection tracking system so that it knows to change the source addresses back in reply packets.
- `DNAT`: A virtual state set when the destination address has been altered by NAT operations. This is used by the connection tracking system so that it knows to change the destination address back when routing reply packets.

IPTABLES (IPv4 + IPv6)



Netfilter = filewal = Packet Filtering System
=> IPTABLES = CLI UI
- Bijna op alle linux distro's
- DB met alle Firewall rules

(sudo) apt-get install iptables

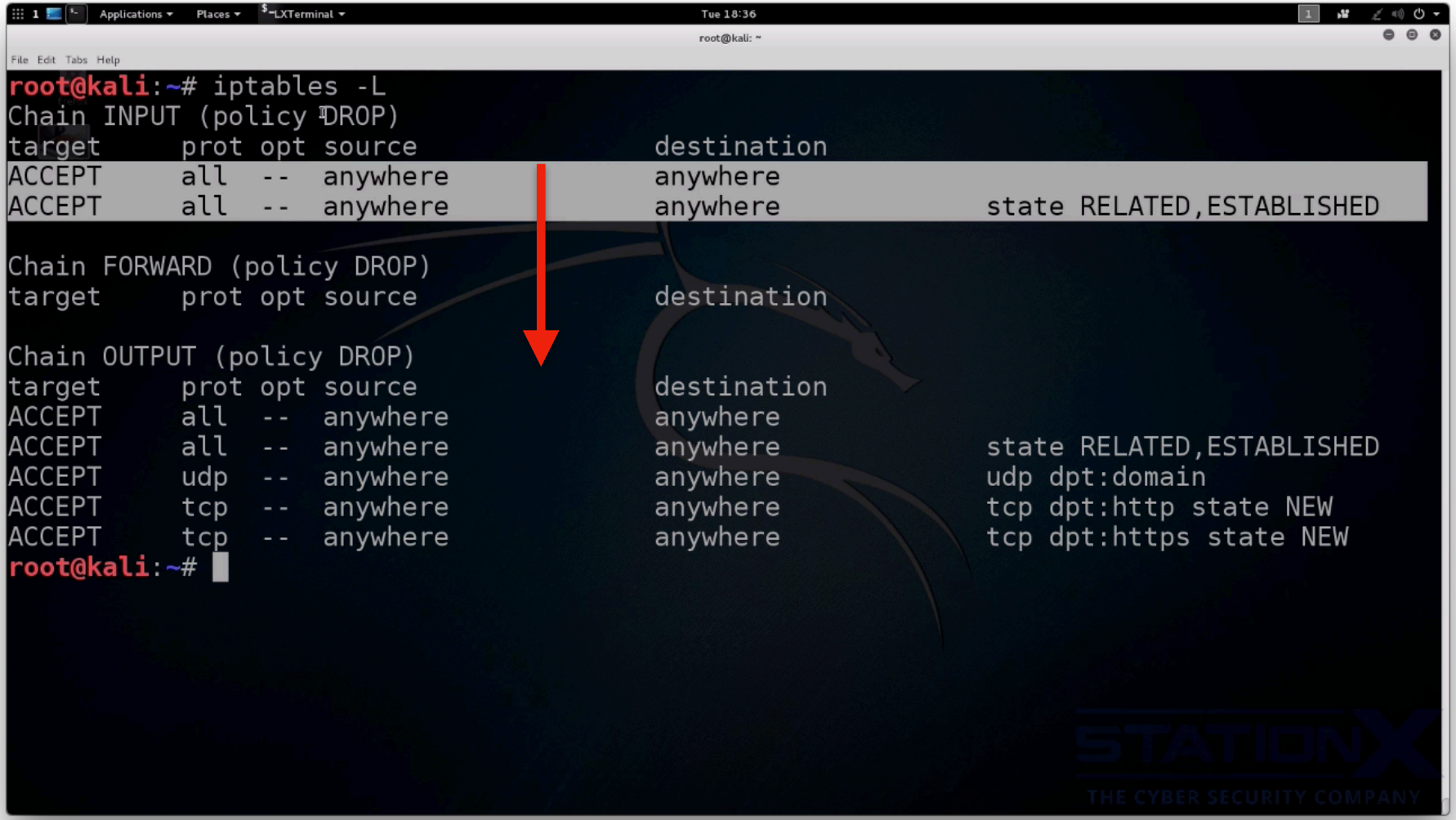


iptables -L => Lijst met alle rules en chains

3 Default chains

1. **INPUT** = Alle inkomende connections
   bv buitenstaander stuurt naar onze laptop
2. **FORWARD** = Ook inkomende connections maar die geforward moeten worden naar andere machines (bv router) Zeer beperkt op gewone computers.
3. **OUTPUT** = Uitgaande connecties. bv. Surfen op het internet.

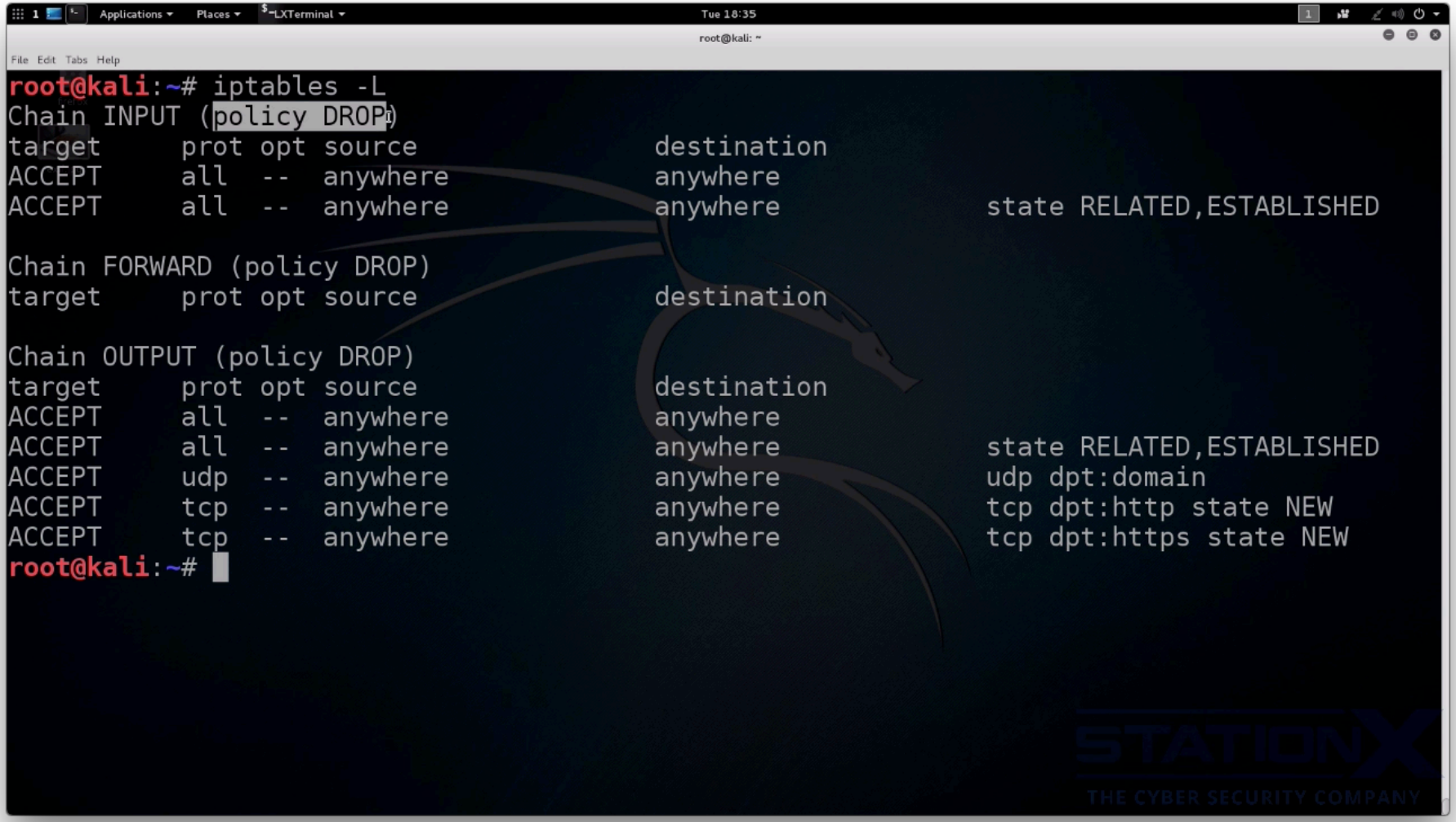
Chains hebben default gedrag bv **DROP** = blokkeer alle verbindingen tenzij er een matching rule gevonden wordt.



katholieke hogeschool
associatie KU Leuven
vives

3 Policies

1. **ACCEPT** = Connectie wordt aanvaard, tenzij er matching DROP rules gevonden worden.
2. **DROP** = Als er geen ACCEPT matching rule gevonden wordt wordt de connectie gedropt. Tegenpartij krijgt timeout!
3. **REJECT** = Zelfde als Drop maar er wordt een error message gestuurd naar de tegenpartij.



katholieke hogeschool
associatie KU Leuven
vives

DELETE & LIST

iptables -F = delete rules
iptables -L = list rules

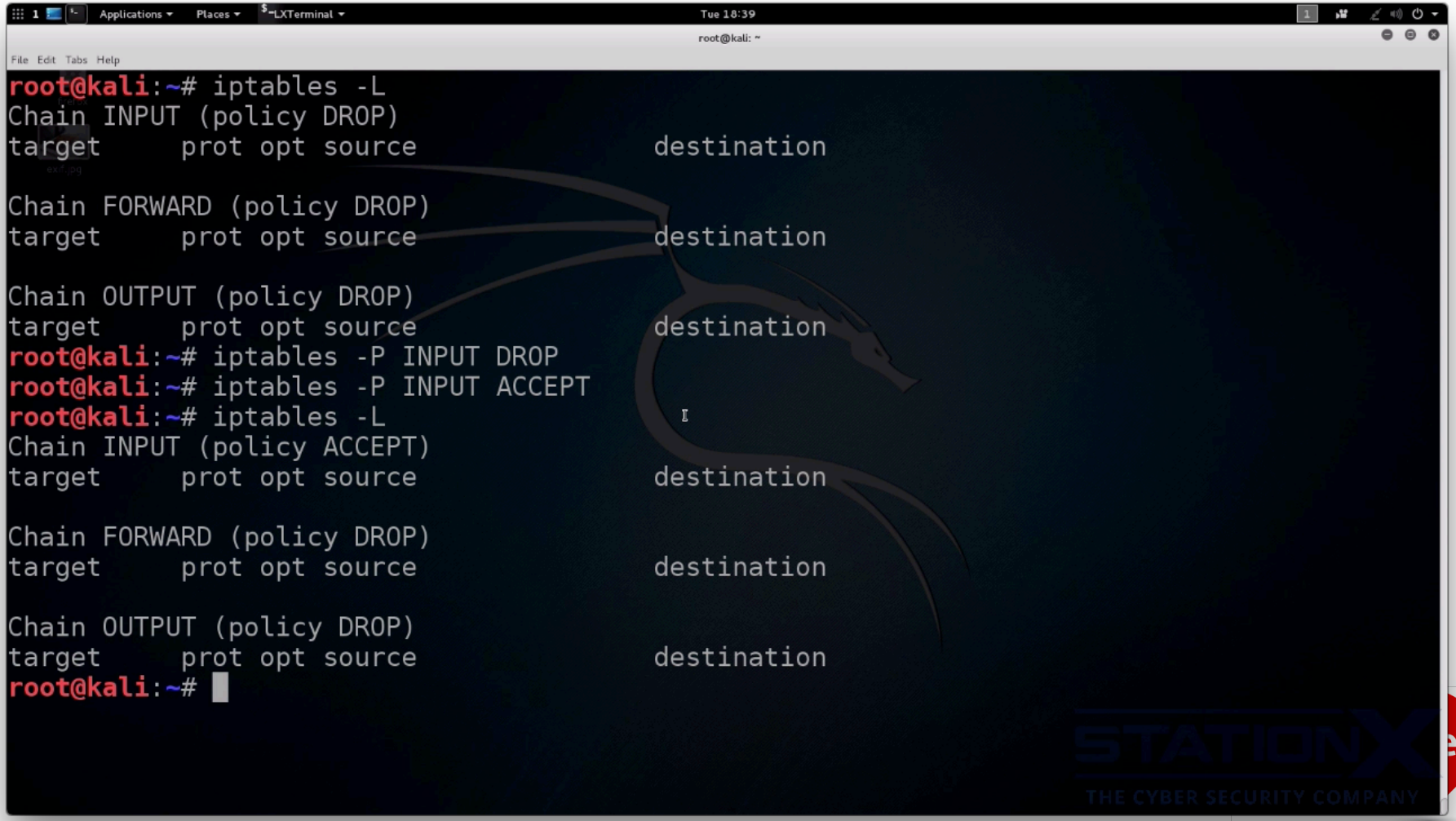
```
root@kali:~# iptables -F
root@kali:~# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy DROP)
target     prot opt source               destination
root@kali:~#
```

SET POLICIES

iptables -P INPUT DROP = default policy voor de INPUT chain = DROP
iptables -P INPUT ACCEPT = default policy voor de INPUT chain = ACCEPT



SET RULES

CASE:
Laptop die enkel mag surfen.

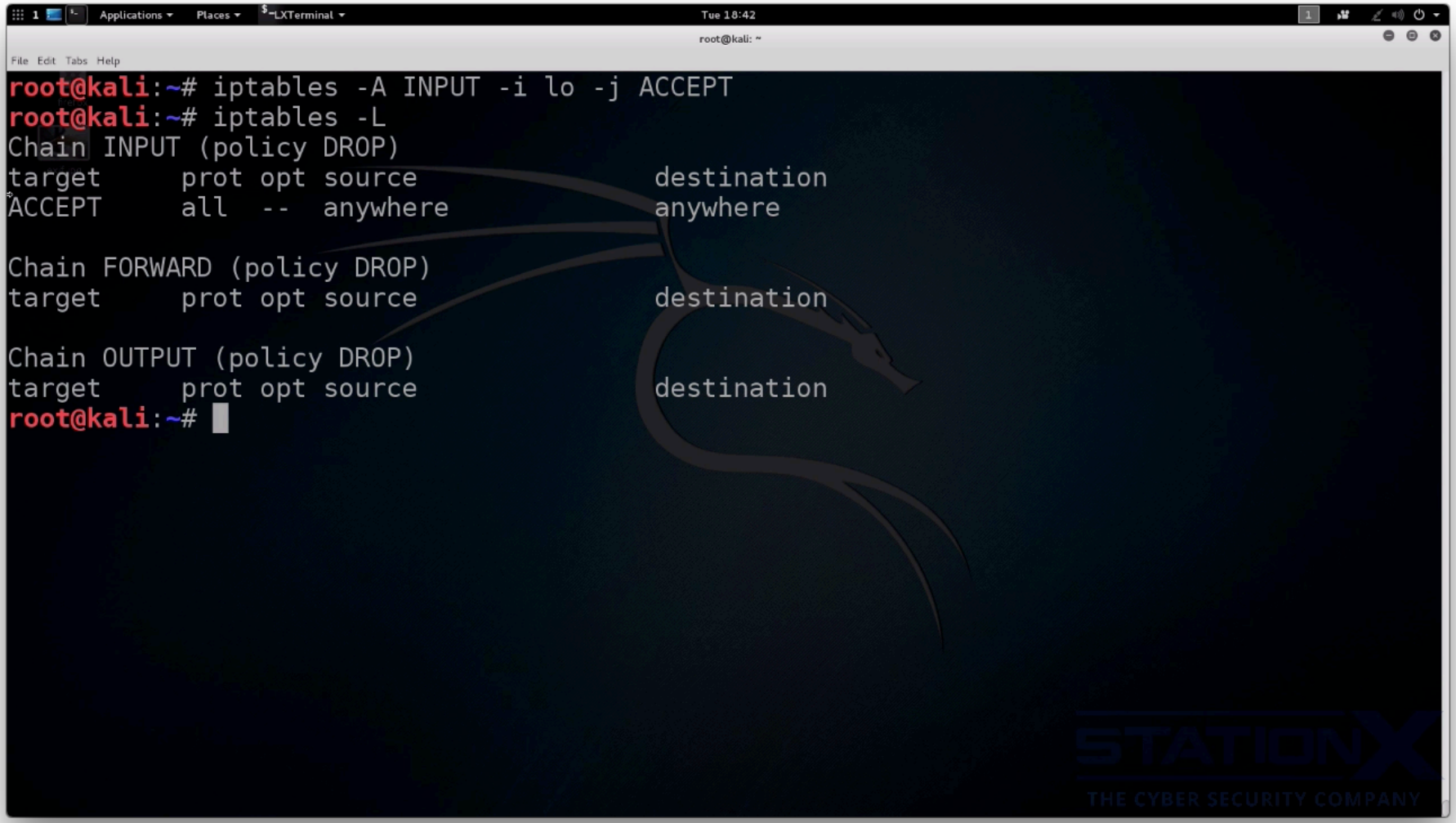
1. INPUT, OUTPUT en FORWARD op **DROP**

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP
```

2. Laat connecties met local interface toe

```
iptables -A INPUT -i lo -j ACCEPT =  alle inkomende packets bestemd voor
                                     de local interface worden aanvaard
```

Veel applicaties moeten kunnen communiceren met de local interface
Altijd instellen!



katholieke hogeschool
associatie KU Leuven
vives

iptables -L -v = verbose listing
iptables -L -v —line-number = verbose listing met genummerde regels
iptables -L -v —line-number -n = verbose listing met genummerde regels en nummers ipv namen.

```
root@kali:~# iptables -L -v
Chain INPUT (policy DROP 2 packets, 386 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     all  --  lo     any     anywhere             anywhere

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

```
root@kali:~# iptables -L -v --line-number
Chain INPUT (policy DROP 9 packets, 1392 bytes)
num   pkts bytes target     prot opt in     out     source               destin

1        0     0 ACCEPT     all  --  lo     any     anywhere             anywhe

Chain FORWARD (policy DROP 0 packets, 0 bytes)
num   pkts
```

```
root@kali:~# iptables -L --line-number -n
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy DROP)
num  target     prot opt source               destination

Chain OUTPUT (policy DROP)
num  target     prot opt source               destination
```

3. Laat gelinkte inkomende packets toe

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

Append

load module state
examine state of the packet
new, established or related?

= enables dynamic packet filtering.
bv regel die uitgaande connecties toelaat naar poort 80 (surfen). Door deze regel worden alle gelinkte inkomende pakketten aan onze http get doorgelaten

```
root@kali:~# iptables -L --line-number -n
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
2    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISH
ED

Chain FORWARD (policy DROP)
num  target     prot opt source               destination

Chain OUTPUT (policy DROP)
num  target     prot opt source               destination
```

4. Laat uitgaande connecties toe verbinding te make

iptables -A OUPUT -o lo -j ACCEPT = alle inkomende packets bestemd voor de local interface worden aanvaard

Veel applicaties moeten kunnen communiceren met de local interface
Altijd instellen!

```
root@kali:~# iptables -A OUTPUT -o lo -j ACCEPT
root@kali:~# iptables -L --line-number -n
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
2    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISH
ED

Chain FORWARD (policy DROP)
num  target     prot opt source               destination

Chain OUTPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
root@kali:~#
```

5. Laat gelinkte uitgaande connecties toe

```
iptables -A OUTPUT -m state —state RELATED,ESTABLISHED -j ACCEPT
```

Append

load module state
examine state of the packet
new, established or related?

= enables dynamic packet filtering.
Deze keer voor uitgaande traffic zodat het terug kan komen

```
root@kali:~# iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
root@kali:~# iptables -L --line-number -n
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
2    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISH
ED

Chain FORWARD (policy DROP)
num  target     prot opt source               destination

Chain OUTPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
2    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISH
ED
```

TUSSENSTAND

Intussen heeft met 4 regels die ervoor zorgen dat:

Applicaties kunnen verbinden met de local host (IN and OUT)

Dynamic packet filtering is enabled (IN and OUT)

6. Laat uitgaande connecties toe bestemd voor poort 53 UDP voor DNS queries

```
iptables -A OUPUT -o eth0 -p udp -m udp —dport 53 -j ACCEPT
```

interface

protocol

load module udp

destination port

= laat alle uitgaande packets via eth0 voor udp, bestemmingspoort 53 door

```
root@kali:~# iptables -L --line-number -n
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
2    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISH
ED

Chain FORWARD (policy DROP)
num  target     prot opt source               destination

Chain OUTPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
2    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISH
ED
3    ACCEPT     udp  --  0.0.0.0/0            0.0.0.0/0           udp dpt:53
```

7. Laat nieuwe uitgaande connecties toe bestemd voor poort 80 (HTTP)

```
iptables -A OUPUT -o eth0 -p tcp -m tcp —dport 80 -m state — —state NEW -j ACCEPT
```

interface

protocol

load module tcp

destination port

load module state

= laat alle uitgaande packets via eth0 voor tcp, bestemmingspoort 80 door met als status NEW door

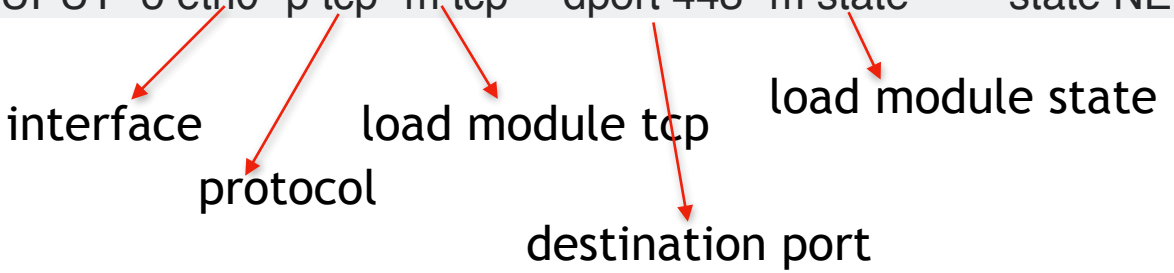
```
root@kali:~# iptables -L --line-number -n
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
2    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISH
ED

Chain FORWARD (policy DROP)
num  target     prot opt source               destination

Chain OUTPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
2    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISH
ED
3    ACCEPT     udp  --  0.0.0.0/0            0.0.0.0/0           udp dpt:53
4    ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:80 state NEW
```

8. Laat nieuwe uitgaande connecties toe bestemd voor poort 443 (HTTPS)

```
iptables -A OUPUT -o eth0 -p tcp -m tcp —dport 443 -m state — —state NEW -j ACCEPT
```



= laat alle uitgaande packets via eth0 voor tcp, bestemmingspoort 443 door met als status NEW door

```
root@kali:~# iptables -L --line-number -n
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
2    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISH
ED

Chain FORWARD (policy DROP)
num  target     prot opt source               destination

Chain OUTPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
2    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISH
ED
3    ACCEPT     udp  --  0.0.0.0/0            0.0.0.0/0           udp dpt:53
4    ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:80 state NEW
5    ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:443 state NEW
```

TESTEN

wget http://www.bbc.co.uk <- HTTP <- DNS -> HTTPS -> wget https://www.bbc.co.uk

```
root@kali:~# wget www.bbc.co.uk
--2016-04-26 18:54:54--  http://www.bbc.co.uk/
Resolving www.bbc.co.uk (www.bbc.co.uk)... 212.58.244.67, 212.58.246.91
Connecting to www.bbc.co.uk (www.bbc.co.uk)|212.58.244.67|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 155326 (152K) [text/html]
Saving to: 'index.html'

index.html              100%[=============================>] 151.69K  --.-KB/s   in 0.09s

2016-04-26 18:54:54 (1.58 MB/s) - 'index.html' saved [155326/155326]
```

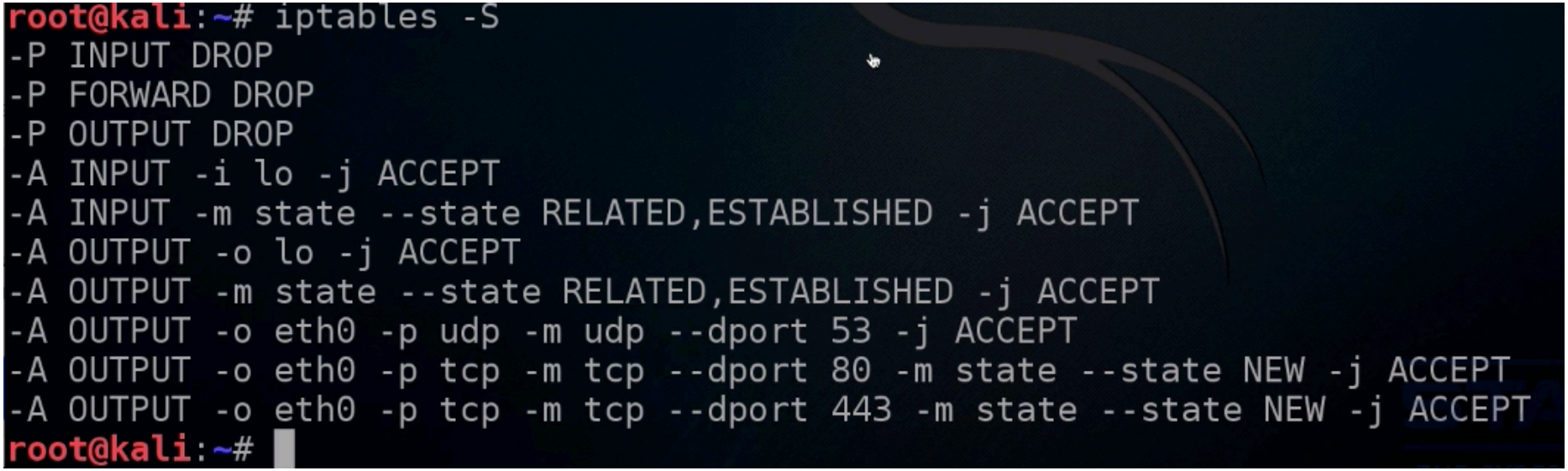
```
root@kali:~# wget https://www.bbc.co.uk
--2016-04-26 18:55:13--  https://www.bbc.co.uk/
Resolving www.bbc.co.uk (www.bbc.co.uk)... 212.58.246.91, 212.58.244.67
Connecting to www.bbc.co.uk (www.bbc.co.uk)|212.58.246.91|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 155311 (152K) [text/html]
Saving to: 'index.html.1'

index.html.1            100%[=============================>] 151.67K  --.-KB/s   in 0.08s

2016-04-26 18:55:13 (1.79 MB/s) - 'index.html.1' saved [155311/155311]
```

Ingevoerde commando's zien

iptables -S = toon eerder ingevoerde commando's



Save rules (Kali + debian)

/sbin/iptables-save = save rules

```
root@kali:~# /sbin/iptables-save
# Generated by iptables-save v1.4.21 on Tue Apr 26 18:56:23 2016
*filter
:INPUT DROP [29:6564]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -o eth0 -p udp -m udp --dport 53 -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp --dport 80 -m state --state NEW -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp --dport 443 -m state --state NEW -j ACCEPT
COMMIT
# Completed on Tue Apr 26 18:56:23 2016
```

Delete rules

iptables -F = Flush (all rules)
iptables -X = Delete alle niet standaard regels
iptables -L --line-number -n = list all rules met line number
iptables -D OUTPUT 5 = delete regel nr. 5 van OUTPUT

```
2    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISH
ED
3    ACCEPT     udp  --  0.0.0.0/0            0.0.0.0/0           udp dpt:53
4    ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:80 state NEW
5    ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:443 state NEW
root@kali:~# iptables -D OUTPUT 5
root@kali:~# iptables -L --line-number -n
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
2    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISH
ED

Chain FORWARD (policy DROP)
num  target     prot opt source               destination

Chain OUTPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
2    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISH
ED
3    ACCEPT     udp  --  0.0.0.0/0            0.0.0.0/0           udp dpt:53
4    ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:80 state NEW
root@kali:~#
```

IPv6

ip6tables -L = List rules
Disable IPv6
ip6tables -P INPUT DROP
ip6tables -P FORWARD DROP
ip6tables -P OUTPUT DROP

Stateless firewall

iptables -A INPUT -p tcp --dport 80 -j ACCEPT

Stateful firewall

iptables -A INPUT -p tcp -m conntrack --ctstate ESTABLISHED -j ACCEPT

iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstate NEW -j ACCEPT

Logging

iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstate NEW -j LOG --log-prefix “In Http:”

Allow Loopback Connections

The **loopback** interface, also referred to as `lo`, is what a computer uses to forward network connections to itself. For example, if you run `ping localhost` or `ping 127.0.0.1`, your server will ping itself using the loopback. The loopback interface is also used if you configure your application server to connect to a database server with a "localhost" address. As such, you will want to be sure that your firewall is allowing these connections.

To accept all traffic on your loopback interface, run these commands:

- `sudo iptables -A INPUT -i lo -j ACCEPT`
- `sudo iptables -A OUTPUT -o lo -j ACCEPT`

Allow Established and Related Incoming Connections

As network traffic generally needs to be two-way—incoming and outgoing—to work properly, it is typical to create a firewall rule that allows **established** and **related** incoming traffic, so that the server will allow return traffic to outgoing connections initiated by the server itself. This command will allow that:

- `sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT`

Allow Established Outgoing Connections

You may want to allow outgoing traffic of all **established** connections, which are typically the response to legitimate incoming connections. This command will allow that:

- `sudo iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT`

Internal to External

Assuming `eth0` is your external network, and `eth1` is your internal network, this will allow your internal to access the external:

- `sudo iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT`

Drop Invalid Packets

Some network traffic packets get marked as **invalid**. Sometimes it can be useful to log this type of packet but often it is fine to drop them. Do so with this command:

- `sudo iptables -A INPUT -m conntrack --ctstate INVALID -j DROP`

Block an IP Address

To block network connections that originate from a specific IP address, `15.15.15.51` for example, run this command:

- `sudo iptables -A INPUT -s 15.15.15.51 -j DROP`

In this example, `-s 15.15.15.51` specifies a **source** IP address of "15.15.15.51". The source IP address can be specified in any firewall rule, including an **allow** rule.

If you want to **reject** the connection instead, which will respond to the connection request with a "connection refused" error, replace "DROP" with "REJECT" like this:

- `sudo iptables -A INPUT -s 15.15.15.51 -j REJECT`

Block Connections to a Network Interface

To block connections from a specific IP address, e.g. `15.15.15.51`, to a specific network interface, e.g. `eth0`, use this command:

- `iptables -A INPUT -i eth0 -s 15.15.15.51 -j DROP`

This is the same as the previous example, with the addition of `-i eth0`. The network interface can be specified in any firewall rule, and is a great way to limit the rule to a particular network.

Service: SSH

If you're using a cloud server, you will probably want to allow incoming SSH connections (port 22) so you can connect to and manage your server. This section covers how to configure your firewall with various SSH-related rules.

Allow All Incoming SSH

To allow all incoming SSH connections run these commands:

- `sudo iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT`
- `sudo iptables -A OUTPUT -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT`

The second command, which allows the outgoing traffic of **established** SSH connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`.

Allow Incoming SSH from Specific IP address or subnet

To allow incoming SSH connections from a specific IP address or subnet, specify the source. For example, if you want to allow the entire `15.15.15.0/24` subnet, run these commands:

- `sudo iptables -A INPUT -p tcp -s 15.15.15.0/24 --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT`
- `sudo iptables -A OUTPUT -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT`

The second command, which allows the outgoing traffic of **established** SSH connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`.

Allow Outgoing SSH

If your firewall `OUTPUT` policy is not set to `ACCEPT`, and you want to allow outgoing SSH connections—your server initiating an SSH connection to another server—you can run these commands:

- `sudo iptables -A OUTPUT -p tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT`
- `sudo iptables -A INPUT -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT`

Allow Incoming Rsync from Specific IP Address or Subnet

Rsync, which runs on port 873, can be used to transfer files from one computer to another.

To allow incoming rsync connections from a specific IP address or subnet, specify the source IP address and the destination port. For example, if you want to allow the entire `15.15.15.0/24` subnet to be able to rsync to your server, run these commands:

- `sudo iptables -A INPUT -p tcp -s 15.15.15.0/24 --dport 873 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT`
- `sudo iptables -A OUTPUT -p tcp --sport 873 -m conntrack --ctstate ESTABLISHED -j ACCEPT`

The second command, which allows the outgoing traffic of **established** rsync connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`.

Service: Web Server

Web servers, such as Apache and Nginx, typically listen for requests on port 80 and 443 for HTTP and HTTPS connections, respectively. If your default policy for incoming traffic is set to drop or deny, you will want to create rules that will allow your server to respond to those requests.

Allow All Incoming HTTP

To allow all incoming HTTP (port 80) connections run these commands:

- `sudo iptables -A INPUT -p tcp --dport 80 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT`
- `sudo iptables -A OUTPUT -p tcp --sport 80 -m conntrack --ctstate ESTABLISHED -j ACCEPT`

The second command, which allows the outgoing traffic of **established** HTTP connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`.

Allow All Incoming HTTPS

To allow all incoming HTTPS (port 443) connections run these commands:

- `sudo iptables -A INPUT -p tcp --dport 443 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT`
- `sudo iptables -A OUTPUT -p tcp --sport 443 -m conntrack --ctstate ESTABLISHED -j ACCEPT`

The second command, which allows the outgoing traffic of **established** HTTP connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`.

Allow All Incoming HTTP and HTTPS

If you want to allow both HTTP and HTTPS traffic, you can use the **multiport** module to create a rule that allows both ports. To allow all incoming HTTP and HTTPS (port 443) connections run these commands:

- `sudo iptables -A INPUT -p tcp -m multiport --dports 80,443 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT`
- `sudo iptables -A OUTPUT -p tcp -m multiport --dports 80,443 -m conntrack --ctstate ESTABLISHED -j ACCEPT`

The second command, which allows the outgoing traffic of **established** HTTP and HTTPS connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`.

Service: MySQL

MySQL listens for client connections on port 3306. If your MySQL database server is being used by a client on a remote server, you need to be sure to allow that traffic.

Allow MySQL from Specific IP Address or Subnet

To allow incoming MySQL connections from a specific IP address or subnet, specify the source. For example, if you want to allow the entire `15.15.15.0/24` subnet, run these commands:

- `sudo iptables -A INPUT -p tcp -s 15.15.15.0/24 --dport 3306 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT`
- `sudo iptables -A OUTPUT -p tcp --sport 3306 -m conntrack --ctstate ESTABLISHED -j ACCEPT`

The second command, which allows the outgoing traffic of **established** MySQL connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`.

Allow MySQL to Specific Network Interface

To allow MySQL connections to a specific network interface—say you have a private network interface `eth1`, for example—use these commands:

- `sudo iptables -A INPUT -i eth1 -p tcp --dport 3306 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT`
- `sudo iptables -A OUTPUT -o eth1 -p tcp --sport 3306 -m conntrack --ctstate ESTABLISHED -j ACCEPT`

The second command, which allows the outgoing traffic of **established** MySQL connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`.

Service: Mail

Mail servers, such as Sendmail and Postfix, listen on a variety of ports depending on the protocols being used for mail delivery. If you are running a mail server, determine which protocols you are using and allow the appropriate types of traffic. We will also show you how to create a rule to block outgoing SMTP mail.

Block Outgoing SMTP Mail

If your server shouldn't be sending outgoing mail, you may want to block that kind of traffic. To block outgoing SMTP mail, which uses port 25, run this command:

- `sudo iptables -A OUTPUT -p tcp --dport 25 -j REJECT`

This configures iptables to **reject** all outgoing traffic on port 25. If you need to reject a different service by its port number, instead of port 25, simply replace it.

Allow All Incoming SMTP

To allow your server to respond to SMTP connections, port 25, run these commands:

- `sudo iptables -A INPUT -p tcp --dport 25 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT`
- `sudo iptables -A OUTPUT -p tcp --sport 25 -m conntrack --ctstate ESTABLISHED -j ACCEPT`

The second command, which allows the outgoing traffic of **established** SMTP connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`.

Note: It is common for SMTP servers to use port 587 for outbound mail.

Allow All Incoming IMAP

To allow your server to respond to IMAP connections, port 143, run these commands:

- `sudo iptables -A INPUT -p tcp --dport 143 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT`
- `sudo iptables -A OUTPUT -p tcp --sport 143 -m conntrack --ctstate ESTABLISHED -j ACCEPT`

The second command, which allows the outgoing traffic of **established** IMAP connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`.

Allow All Incoming IMAPS

To allow your server to respond to IMAPS connections, port 993, run these commands:

- `sudo iptables -A INPUT -p tcp --dport 993 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT`
- `sudo iptables -A OUTPUT -p tcp --sport 993 -m conntrack --ctstate ESTABLISHED -j ACCEPT`

The second command, which allows the outgoing traffic of **established** IMAPS connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`.

Allow All Incoming POP3

To allow your server to respond to POP3 connections, port 110, run these commands:

- `sudo iptables -A INPUT -p tcp --dport 110 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT`
- `sudo iptables -A OUTPUT -p tcp --sport 110 -m conntrack --ctstate ESTABLISHED -j ACCEPT`

The second command, which allows the outgoing traffic of **established** POP3 connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`.

Allow All Incoming POP3S

To allow your server to respond to POP3S connections, port 995, run these commands:

- `sudo iptables -A INPUT -p tcp --dport 995 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT`
- `sudo iptables -A OUTPUT -p tcp --sport 995 -m conntrack --ctstate ESTABLISHED -j ACCEPT`

The second command, which allows the outgoing traffic of **established** POP3S connections, is only necessary if the `OUTPUT` policy is not set to `ACCEPT`.

Oefening:

zie:

iptables lab.docx

indienen in de voorziene uploadzone op Toledo
Permanente evaluatie!

iptables

Julia Evans
@b0rk

jvns.ca/zines

iptables lets you create rules to match network packets and accept / drop / modify them.

It's often used for firewalls or to do NAT

tables have chains

chains have rules

tables: filter, nat, mangle, raw, security

chains: INPUT, FORWARD, PREROUTING, etc

rules: like -s 10.0.0.0/8 -j DROP

iptables-save

This prints out all iptables rules. You can restore them with iptables-restore, but it's also the easiest way to view all rules!

-j TARGET

Every iptables rule has a target (what to do with matching packets). Options:

- ACCEPT / DROP / RETURN
- the name of an iptables chain
- an extension (man iptables-extensions) Popular: DNAT, LOG, MASQUERADE

tables have different chains

filter: INPUT / OUTPUT / FORWARD

mangle: INPUT / OUTPUT / FORWARD / PREROUTING / POSTROUTING

nat: OUTPUT / PREROUTING / POSTROUTING

It helps to learn when packets get processed by a given table / chain (eg filter + OUTPUT = all locally generated packets)

you can match lots of packet attributes

-s: src ip

-p: tcp/udp

-d: dst ip

-i: network interface

-m: lots of things! (bpf rules! cgroups! conntrack state! cpu! ICMP type! more!)

'man iptables-extensions' for more.