Firewall Linux

Posted on July 5, 2017 by macedofernando

Basiclly, there are two types of firewall:

Network Firewall – Known as edge firewall, has the function to manage all the packages that come in and go out from network (INPUT, OUTPUT, NAT).

Host Firewall – Configured locally in each server, your main function is manage the exchange of packages between the host and internal or external network (INPUT/OUTPUT).

A firewall makes afilter of network packages. The data are transmitted on the internet grouped in TCP packages in general.These packages may contain up to 1460 bytes of data. Beyond of data, 40 bytes additional will togheter in the package. In these 40 bytes contains:

Source IP
Destination IP
Source port
Destination port
Verification code
Number package

The are 65.536 ports TCP and UDP. They are numbered from 0 to 65.535.The low ports are in the range between 0 and 1023. They are reserved to services like email, NFS, Samba, etc. Hight ports are in the range above 1023.

> Iptables

Iptables is a firewall at level packet and works based in the adress/port of source/destination of packet and priority. It is just a frontend that manage Netfilter support in the Kernel.

It works by comparing rules whether a packet has permission to pass.

It has 3 policies: ACCEPT, DROP and REJECT;

It has 5 chains: INPUT, OUTPUT, FORWARD, PREROUTING, POSTROUTING.

> What are the CHAINS?

Basically, in the iptables the rules are organized in tables, these tables contains what we call of CHAINS. Is on the CHAINS that we define the rules and a default policy (drop or accept).

Basic commands:

```
iptables -L (table default: Filter)
iptables -L -t filter
iptables -L -t nat
iptables -L -t mangle
iptables -L -t raw
iptables -L -t security
```

-L (List) – It's used to list active rules

-t (table) – It's used to define a table

> Tables

Are the places used to store the chains and rules of our Firewall:

FILTER: Rules responsible for determining all comes in and comes out in the localhost. Very used in Host Firewall;
NAT: Used for data that generates another connection, such as masking the internet, redirect request. Essential for network firewall.
MANGLE: Used for special packages changes, such as markup to QOS and link balancing.
RAW: Mark packages to track later, used to configure exceptions, which makes it stay in the top of all other, begin the first to process the packages.
SECURITY: Used to network rules to mandatory access control (MAC – Mandatory Acces Crontol), specific to integrate with the SELINUX.

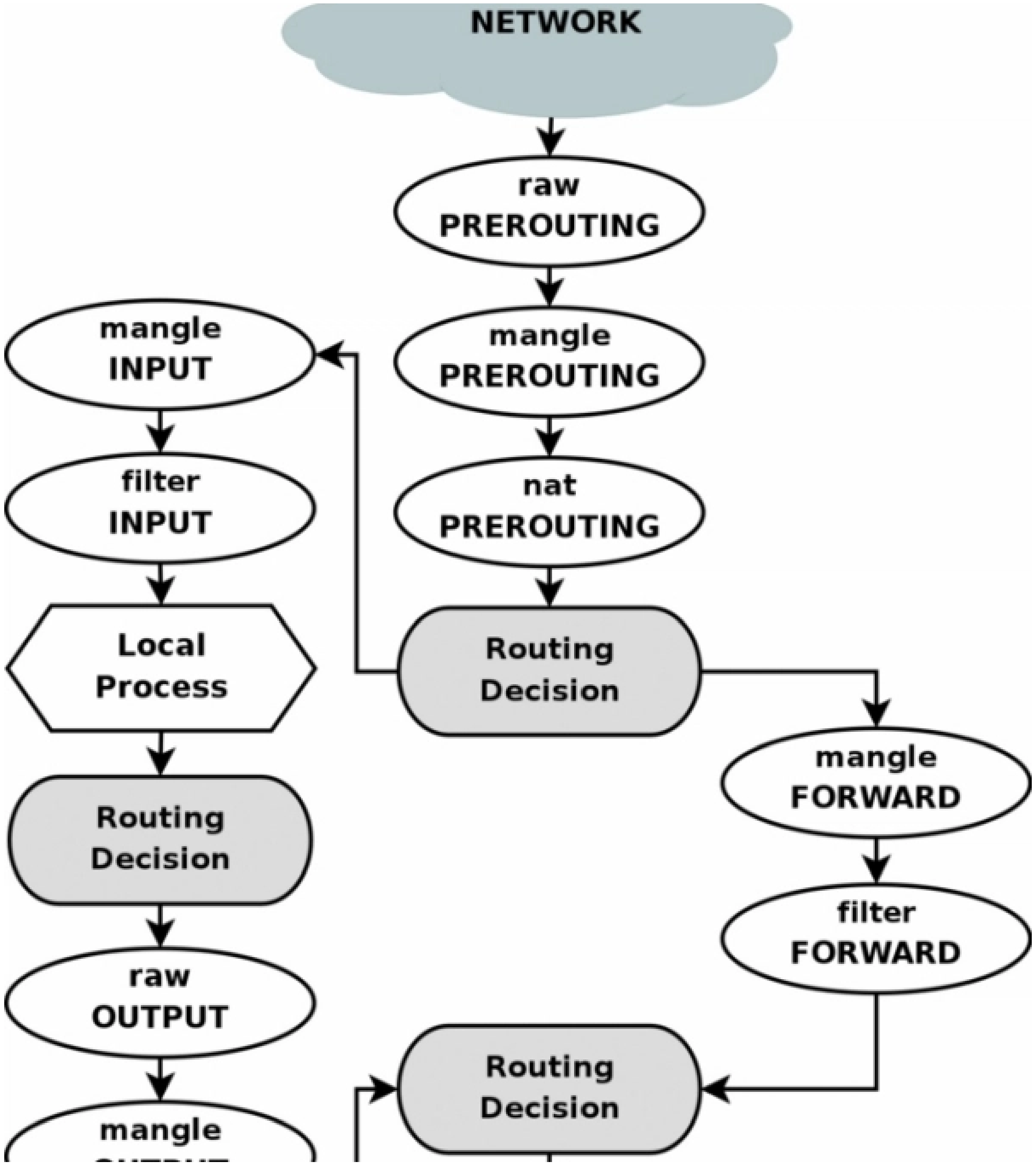
> Chains

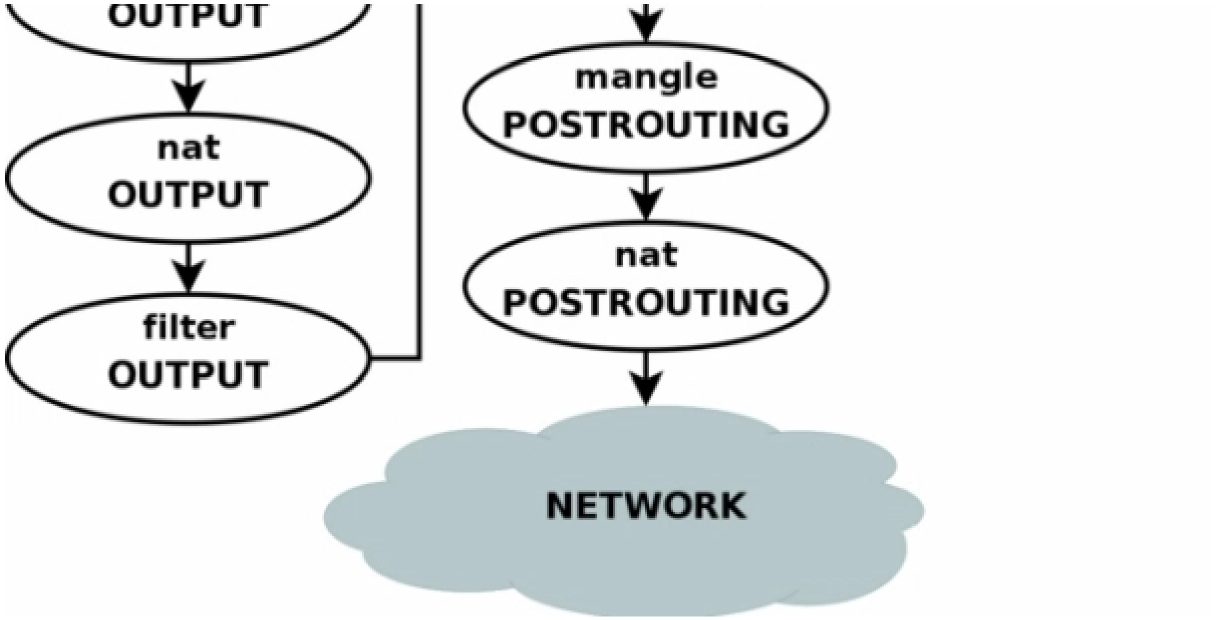
Are the places where the rules are stored according to their tables:

Filter: INPUT, OUTPUT, FORWARD
NAT: PREROUTING, POSTROUTING, INPUT, OUTPUT
Mangle: PREROUTING, POSTROUTIG, INPUT, OUTPUT, FORWARD
RAW: PREROUTING, OUTPUT
Security: INPUT, OUTPUT, FORWARD

The chains can be created ou altered by the users.

Let’s see an image that’s explain this better:


NETWORK
raw PREROUTING
mangle PREROUTING
nat PREROUTING
Routing Decision
mangle INPUT
filter INPUT
Local Process
Routing Decision
raw OUTPUT
mangle
mangle FORWARD
filter FORWARD
Routing Decision



> Policies and Rules

Each CHAIN has a default policy that will to determine what type of rules you will create in the CHAIN.

ACCEPT: Is created rules to release packages
DROP: Block a package silently, no answer is returned.
REJECT: Block a package and return an error, informing that the package was blocked.

It is advisable to define the policy DROP for all CHAINS, and after to configure a rule for each exception.

This prevents DOS/DDOS attacks.

Basic commands:

```
iptables -t filter -S
iptables -t filter -P INPUT DROP
iptables -t filer -nL
```

-S – List the policy of all the chains or one in specific

-P – Define the policy of the chain

Creating rules:

iptables -t **filter** **-A** INPUT **-s 127.0.0.1 -d 127.0.0.1** **–j ACCEPT**

-A – Append the rule in the end of the configuration file

filter – Where the rule will be stored. Always we must indicate the table and chain

-s 127.0.0.1 -d 127.0.0.1 – The rule, the information that we pass in the rule are: source and destination of package, service port, protocol, etc;

-j ACCEPT – The action of the rule. Whether the package will be accept, droped, redirected…

Iptables parameter table:

-P	–policy	Establishe an access policy to a chain
-t	table	Select a table

-A	–append	Adds as last rule of the sequence of a chain
-I	–insert	Insert as frist rule of the sequence of a chain
-N	–new-chain	Create a new chain
-D	–delete	Remove a rule
-X	–delete-chain	Remove all the rules of a user chain
-F	–flush	Remove all the rules present in a default(INPUT, FORWARD etc) or tables (all chains)

```
iptables -t filter -nL --line-numbers
iptables -t filer -D INPUT 1
iptables -t filter -A INPUT -j ACCEPT
iptables -t filter -F
```

Iptables parameter table:

-s	–source	Determine the source package
-d	–destination	Determine the destination package
–dport	–destination-port	Determine the destination port
–sport	–source-port	Determine the source port
-i	–in-interface	Determine the input interface

-o	–out-interface	Determine the output interface
-p	–protocol	Select a protocol (tcp, udp, icmp etc)

All the rules are created in the memory, therefore once the server restart everything is lost. The Iptables doesn't have a configuration file, he offers two commands to you can save the rules and enable in the initialization.

iptables-save – Save the rules in an file
iptables-restore – Enable the rules stored by iptables-save

Examples:

```
iptables-save > /tmp/firewall
cat /tmp/firewall
iptables -F
init 6
iptables-restore < /tmp/firewall
```

> How can I make a specific set of iptables rules permanent?

The simplest method is to use iptables-save and iptables-restore to save the currently-defined iptables rules to a file and (re)load them (e.g., upon reboot).

So, for instance, you would run

```
sudo iptables-save > /etc/iptables.conf
```

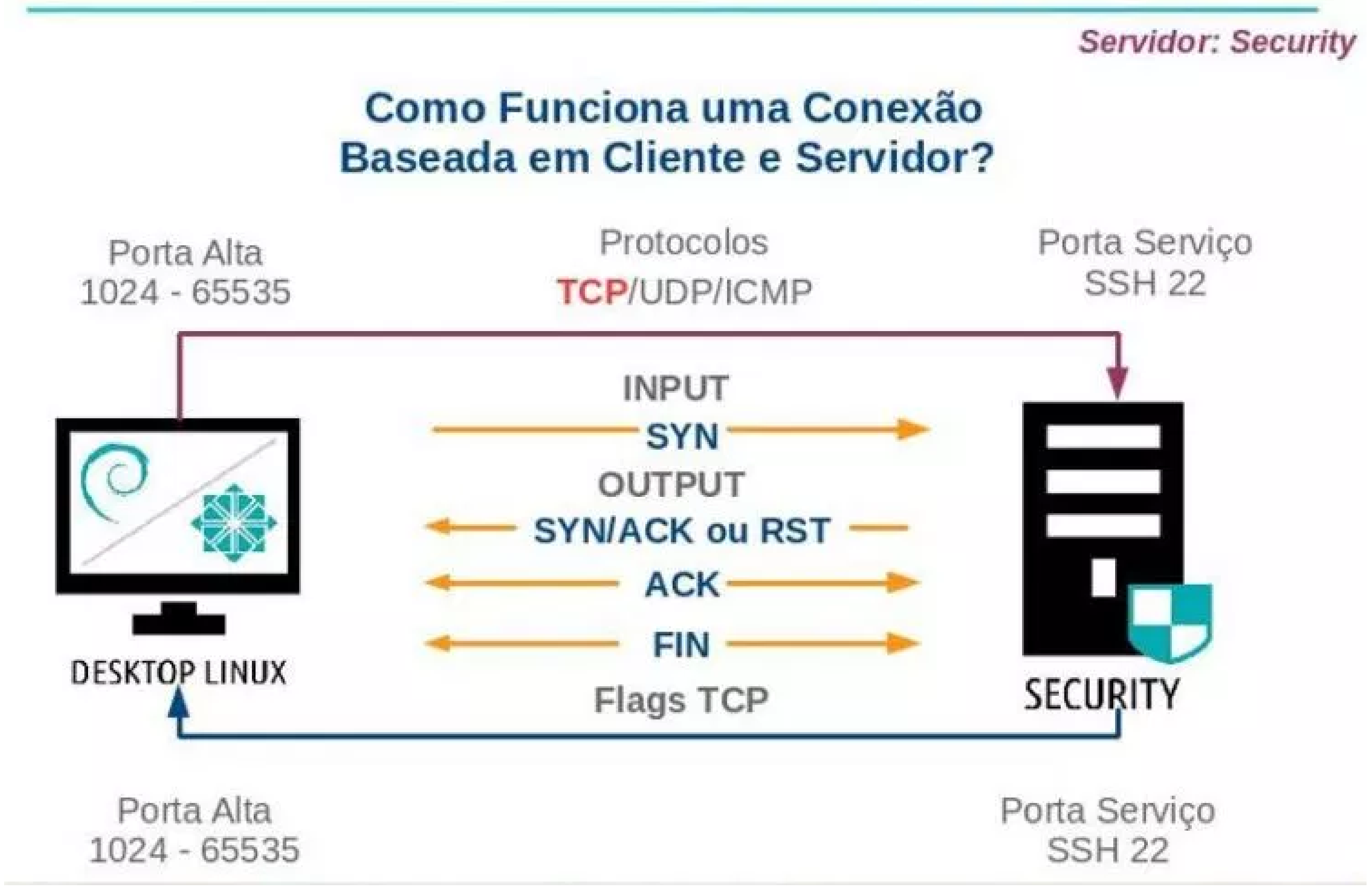
to save your current iptables rules to

```
/etc/iptables.conf
```

and then insert these lines in /etc/rc.local:

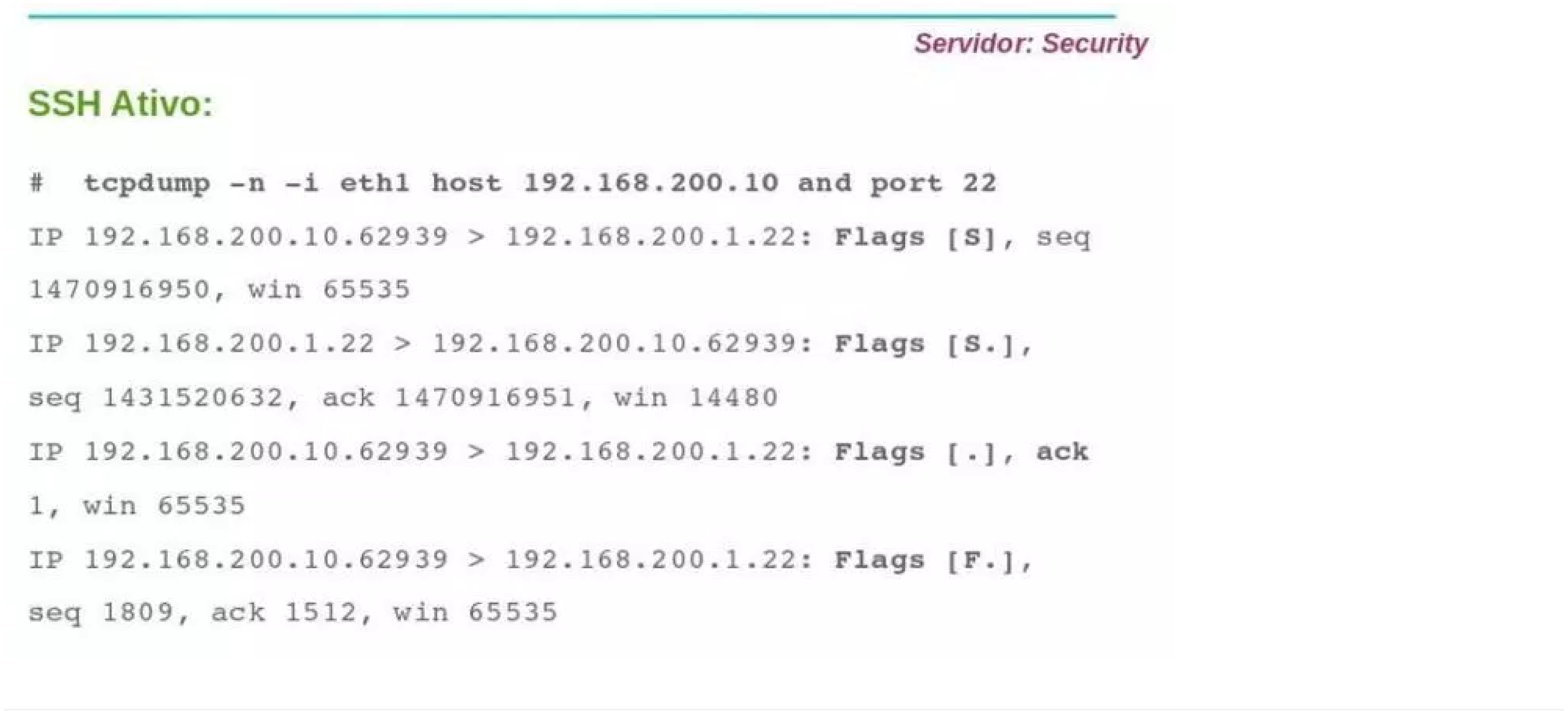
```
# Load iptables rules from this file
iptables-restore < /etc/iptables.conf
```

Let’s see how works a connection client to server using protocol TCP:



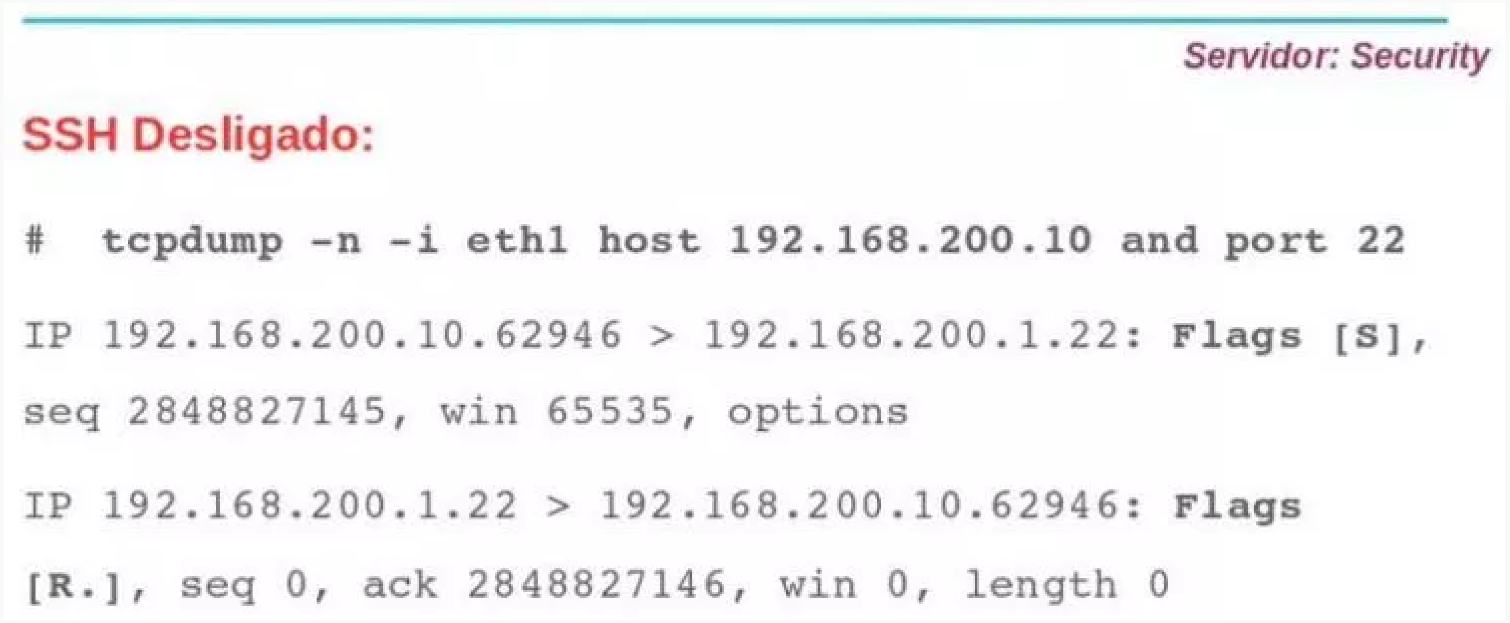
We can use the command tcpdum to monitor and capture the packages with source or destination of the localhost utilized. Where in the commando below we configured the the listen inthe interface enp0s3 with source or destination to the address IP 192.168.0.104 and the source port ou destination port 22:

```
tcpdump -n -i enp0s3 host 192.168.0.104 and port 22
```



SSH Ativo:

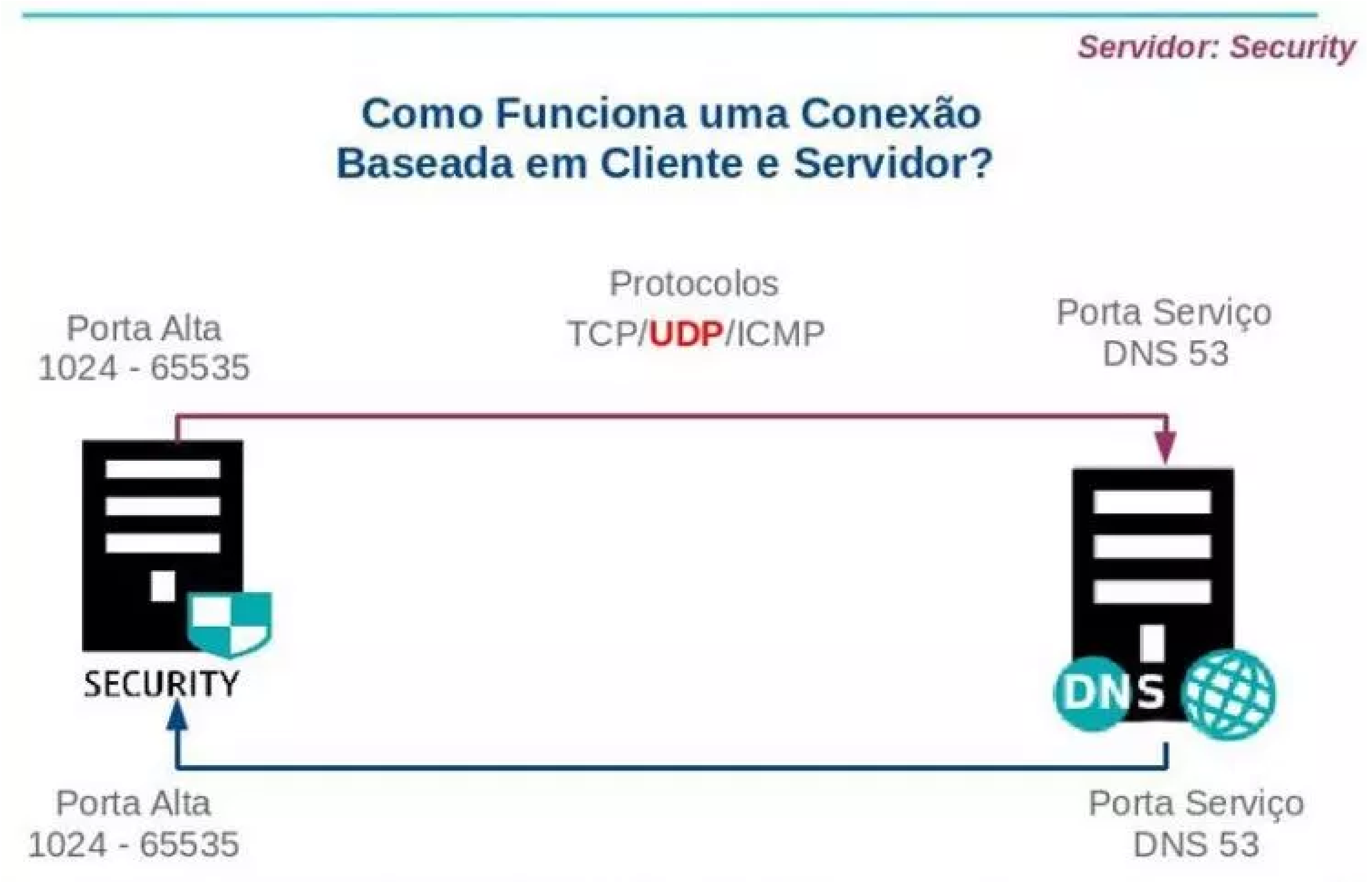
```
#  tcpdump -n -i eth1 host 192.168.200.10 and port 22
IP 192.168.200.10.62939 > 192.168.200.1.22: Flags [S], seq
1470916950, win 65535
IP 192.168.200.1.22 > 192.168.200.10.62939: Flags [S.],
seq 1431520632, ack 1470916951, win 14480
IP 192.168.200.10.62939 > 192.168.200.1.22: Flags [.], ack
1, win 65535
IP 192.168.200.10.62939 > 192.168.200.1.22: Flags [F.],
seq 1809, ack 1512, win 65535
```


Servidor: Security

SSH Desligado:

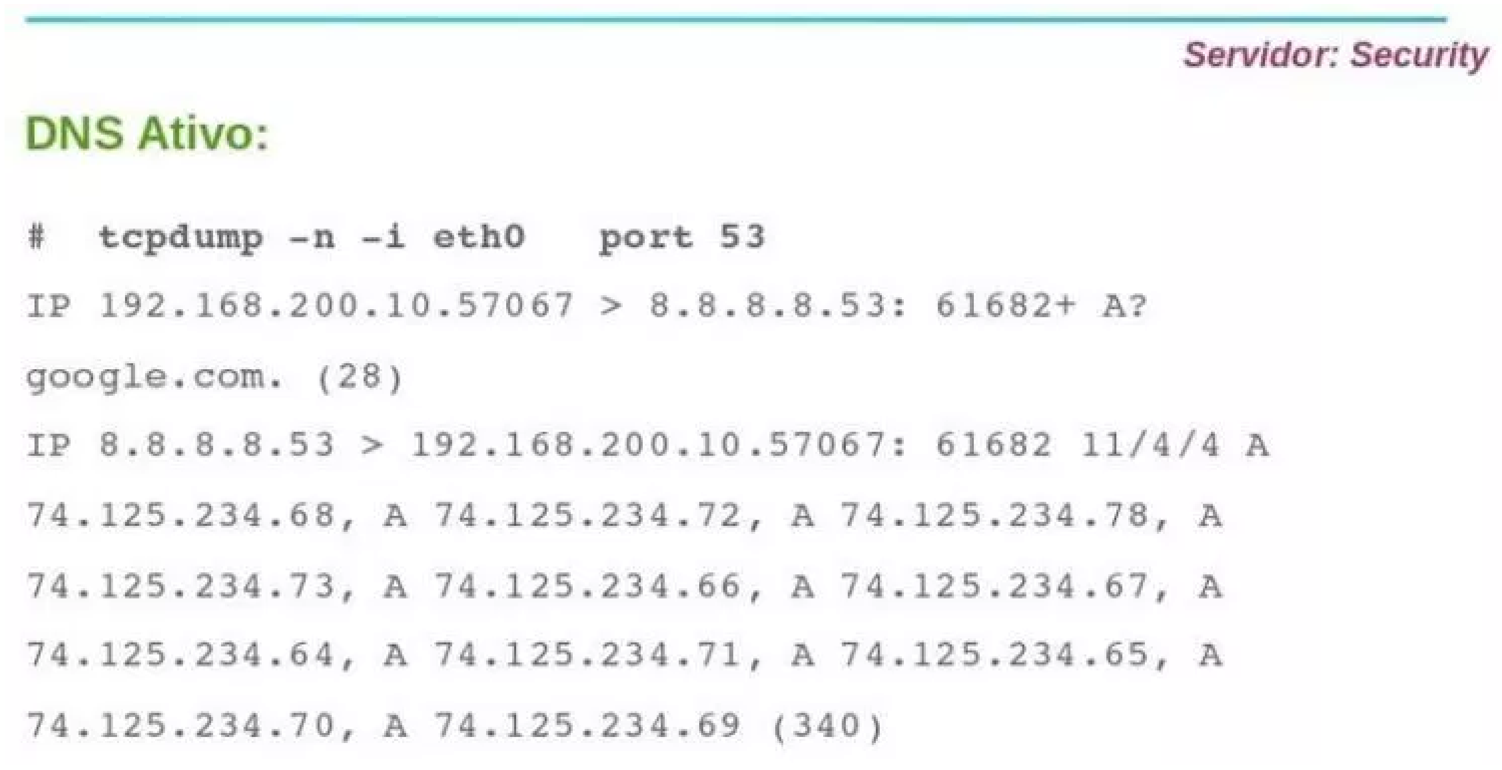
```
#  tcpdump -n -i eth1 host 192.168.200.10 and port 22

IP 192.168.200.10.62946 > 192.168.200.1.22: Flags [S],
seq 2848827145, win 65535, options

IP 192.168.200.1.22 > 192.168.200.10.62946: Flags
[R.], seq 0, ack 2848827146, win 0, length 0
```

Let’s see how works a connection client to server using protocol UDP:



```
tcpdump -i enp0s3 port 53
```

Now, execute a command like dig or nslookup in another terminal, to see the packages...



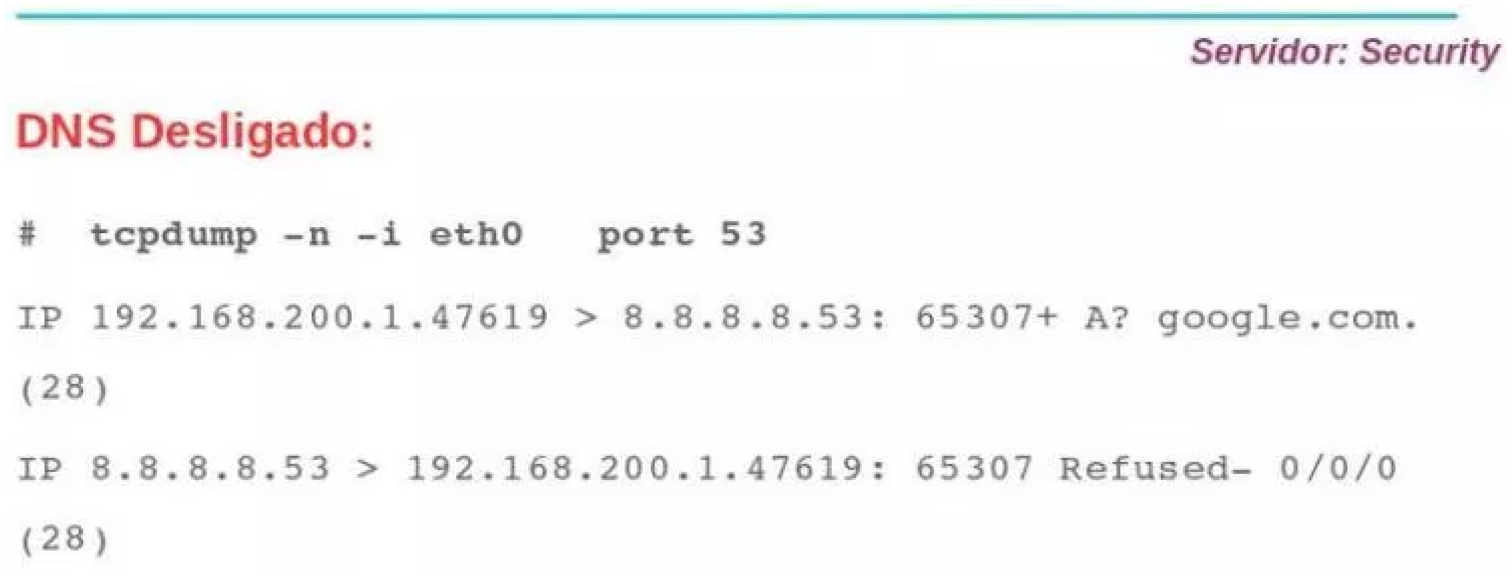
Servidor: Security

DNS Ativo:

```
#  tcpdump -n -i eth0   port 53
IP 192.168.200.10.57067 > 8.8.8.8.53: 61682+ A?
google.com. (28)
IP 8.8.8.8.53 > 192.168.200.10.57067: 61682 11/4/4 A
74.125.234.68, A 74.125.234.72, A 74.125.234.78, A
74.125.234.73, A 74.125.234.66, A 74.125.234.67, A
74.125.234.64, A 74.125.234.71, A 74.125.234.65, A
74.125.234.70, A 74.125.234.69 (340)
```

To block packages from dns:

```
iptables -t filter -A OUTPUT -p udp -s 0/0 -d 0/0 --dport 53 -j DROP
```

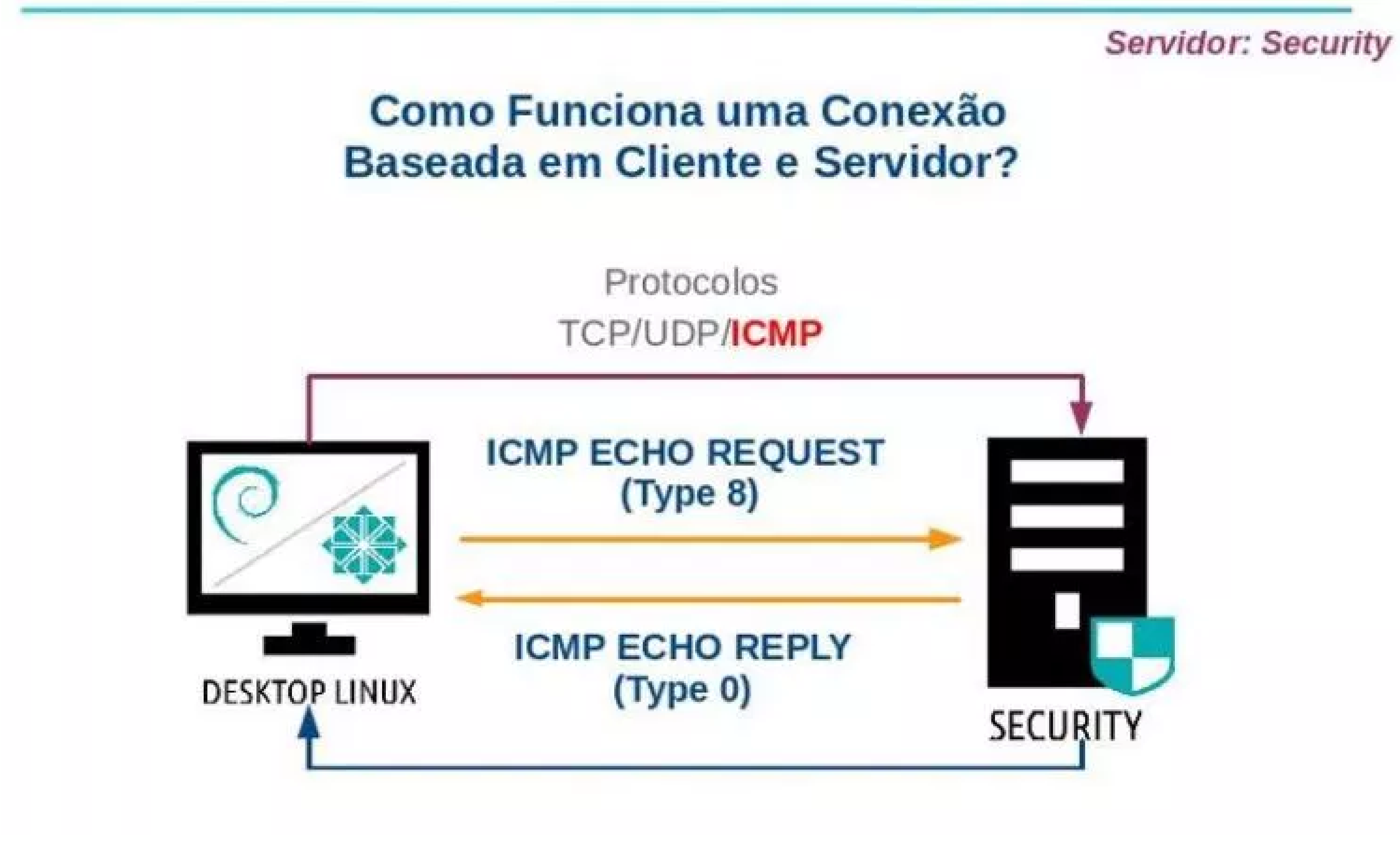


Servidor: Security

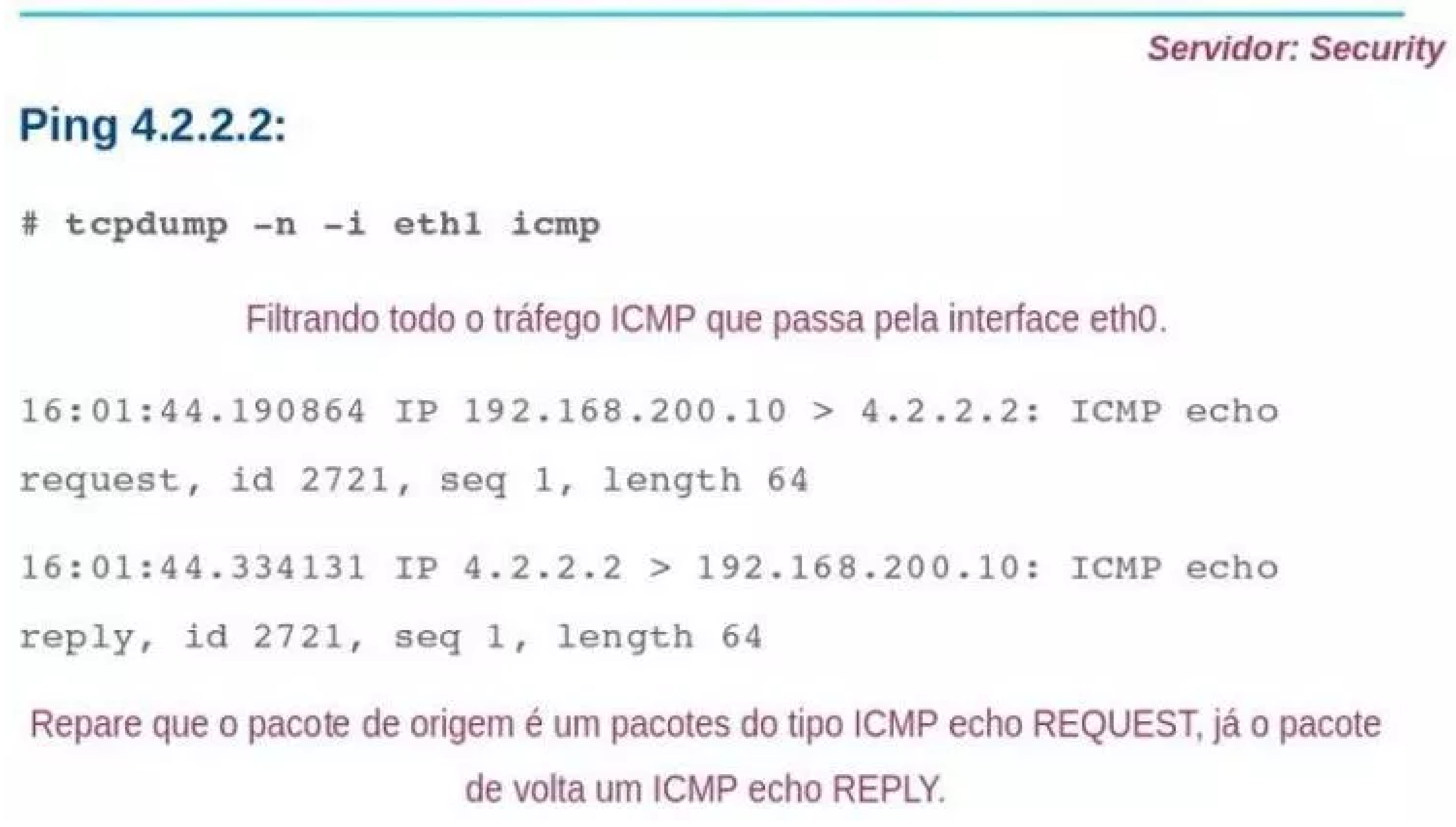
DNS Desligado:

```
#  tcpdump -n -i eth0   port 53
IP 192.168.200.1.47619 > 8.8.8.8.53: 65307+ A? google.com.
(28)
IP 8.8.8.8.53 > 192.168.200.1.47619: 65307 Refused- 0/0/0
(28)
```

Let’s see how works a connection client to server using protocol ICMP:



```
[root@ZabbixServer ~]# tcpdump -i enp0s3 -p icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), capture size 65535 bytes
20:12:17.874220 IP 192.168.0.102 &amp;gt; wsgi_noc: ICMP echo request, id 1, seq 13, length 40
20:12:17.874247 IP wsgi noc &amp;gt; 192.168.0.102: ICMP echo reply, id 1, seq 13, length 40
```



Servidor: Security

Ping 4.2.2.2:

```
# tcpdump -n -i eth1 icmp
```

Filtrando todo o tráfego ICMP que passa pela interface eth0.

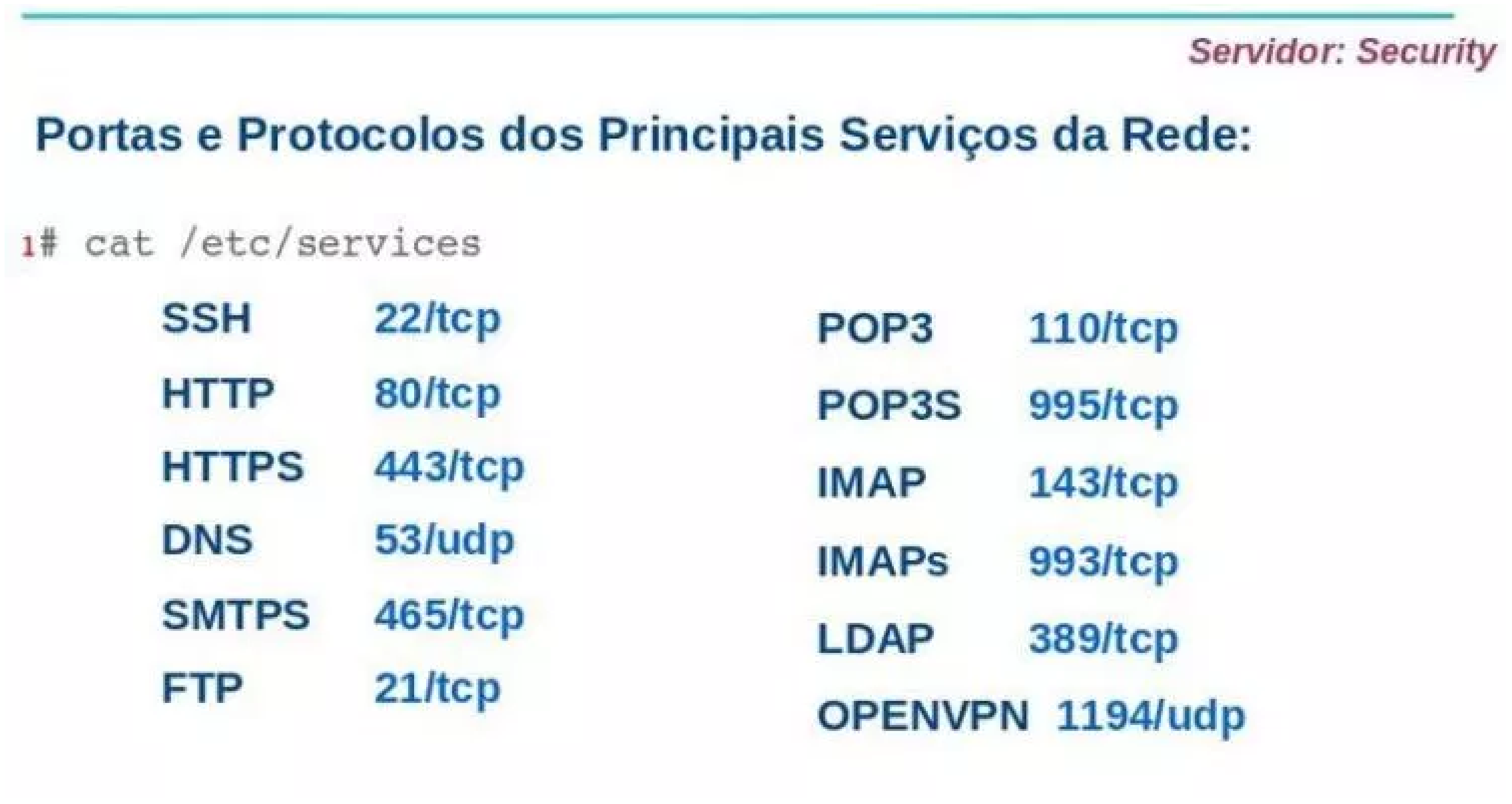
```
16:01:44.190864 IP 192.168.200.10 > 4.2.2.2: ICMP echo
request, id 2721, seq 1, length 64

16:01:44.334131 IP 4.2.2.2 > 192.168.200.10: ICMP echo
reply, id 2721, seq 1, length 64
```

Repare que o pacote de origem é um pacotes do tipo ICMP echo REQUEST, já o pacote de volta um ICMP echo REPLY.

To block icmp packages, do:

```
iptables -t filter -A INPUT -p icmp -s 0/0 -d 0/0 -j DROP
iptables -t filter -A OUTPUT -p icmp -s 0/0 -d 0/0 -j DROP
```

The most import ports and protocols of main network services:


Servidor: Security

Portas e Protocolos dos Principais Serviços da Rede:

```
1# cat /etc/services
```

SSH	22/tcp	POP3	110/tcp
HTTP	80/tcp	POP3S	995/tcp
HTTPS	443/tcp	IMAP	143/tcp
DNS	53/udp	IMAPs	993/tcp
SMTPS	465/tcp	LDAP	389/tcp
FTP	21/tcp	OPENVPN	1194/udp

Let’s configure a firewall server. First let’s define the policies:

1º – It’s most security when we block everything, and just allow what is necessary.

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

2º – Let's allow the access to loopback (127.0.01)

```
iptables -t filter -A INPUT -s 0/0 -d 127.0.0.1 -j ACCEPT
iptables -t filter -A OUTPUT -s 127.0.0.1 -d 0/0 -j ACCEPT
```

3ª Allow ping for the internet and netowrk LAN:

```
iptables -t filter -A OUTPUT -p icmp -s 0/0 -d 0/0 -j ACCEPT
iptables -t filter -A INPUT -p icmp --icmp-type 0 -s 0/0 -d 0/0 -j ACCEPT
iptables -t filter -A OUTPUT -p icmp --icmp-type 8 -s 192.168.0.0/24 -d 0/0 -j ACCEPT
```

A security policy is to block the icmp protocol in the firewall. However it's interesting that the firewall answer icmp to the internal network, to help in troubleshooting of network.

4ª Allow query dns throught the firewall:

```
iptables -t filter -A OUTPUT -p udp -s 192.168.0.104 -d 0/0 --dport 53 -j ACCEPT
iptables -t filter -A INPUT -p udp -s 0/0 --sport 53 -d 192.168.0.104 -j ACCEPT
```

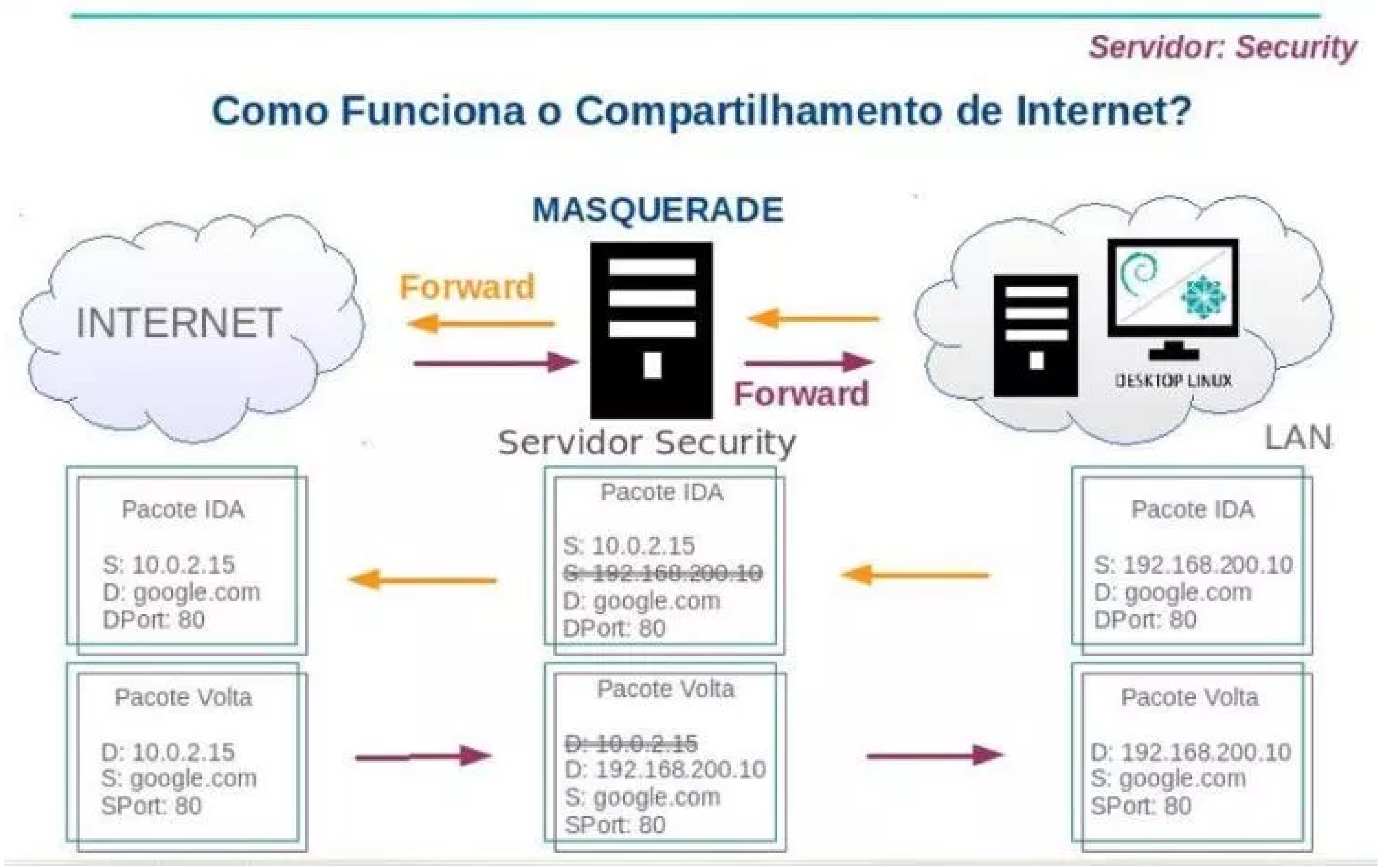
5ª Allow internet accesss throught the firewall:

```
iptables -t filter -A OUTPUT -p tcp -m multiport -s 192.168.0.104 -d 0/0 --dport 80,443 -j ACCEPT
iptables -t filter -A INPUT -p tcp -m multiport -s 0/0 --sport 80,443 -d 192.168.0.104 -j ACCEPT
```

6ª Allow ssh access throught the other server in the same network:

```
iptables -t filter -A INPUT -p tcp -s 192.168.0.102 -d 192.168.0.14 --dport 22 -j ACCEPT
iptables -t filter -A OUTPUT -p tcp -s 192.168.0.104 -d 192.168.0.102 --sport 22 -j ACCEPT
```

Let's see how works the forward function:



7º Allow internet access to the machines of subnetworks:

To work with MASQUERADE, NAT even routing of packages by routing table is necessary to active the transfer of packeges between networks cards in the kernel:

```
vim /etc/sysctl.conf ( This file is a configuration file of kernel)
28 net.ipv4.ip_forward=1
sysctl -p (This command makes the kernel re-read this file)
cat /proc/sys/net/ipv4/ip forward
```

Liberate the access to the internet to the machines of the SubNetworks:

```
iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -d 0/0 -j MASQUERADE
```

All comes from SubNetwork1 with destination to the internet will be MASQUERADE

```
iptables -t filter -A FOWARD -p tcp -m multiport -s 192.168.0.0/24 -d 0/0 --dport 80,443 -j ACCEPT
```

All comes from SubNetwork1 with destination to the internet in the ports 80 and 443 I allow the transfer of packages

```
iptables -t filter -A FORWARD -p tcp -m multiport -s 0/0 --sport 80,443 -d 192.168.0.0/24 -j ACCEP
```

All comes from the internet in the ports 80 and 443 with destination to the SubNetwork1 i allow the transfer of packages

8º Allow query access DNS to the machines of subnetorks:

```
iptables -t filter -A FORWARD -p udp -s 192.168.0.0/24 -d 0/0 --dport 53 -j ACCEPT
```

All comes from the internal network with destination to the internet in the port 53 I allow the transfer os packages

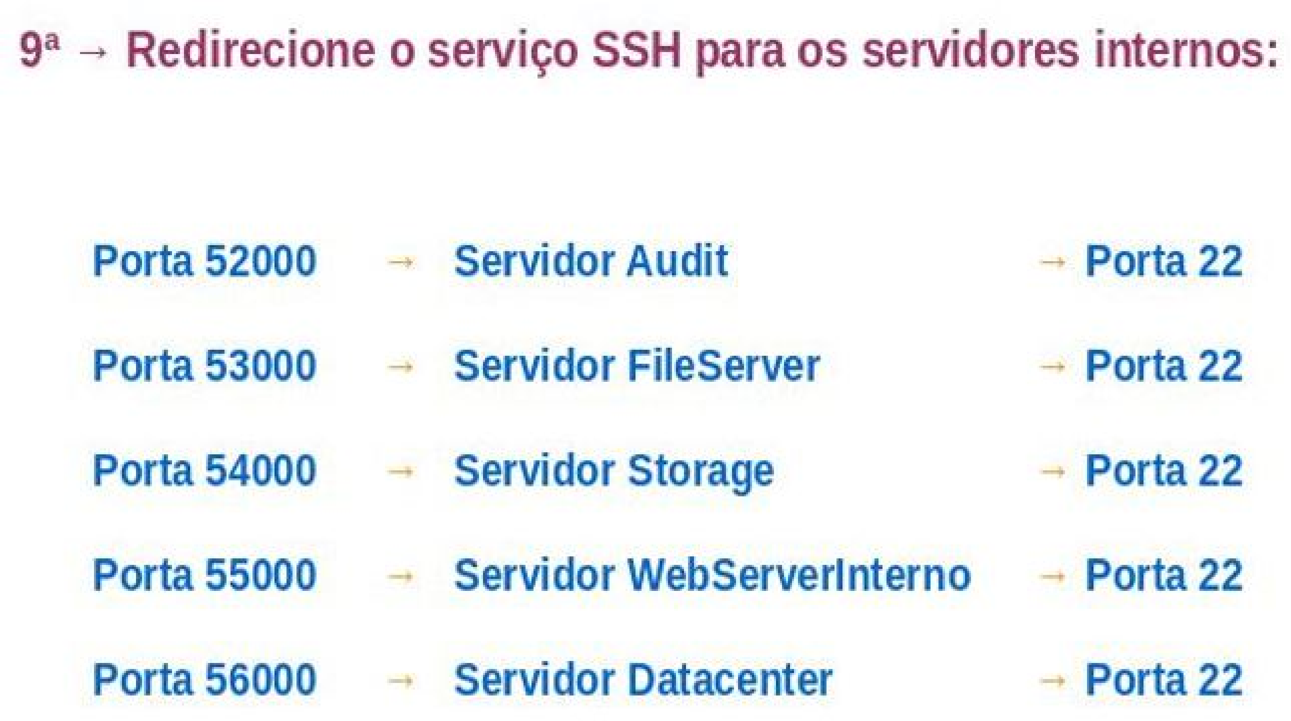
```
iptables -t filter -A FORWARD -p udp -s 0/0 -d 192.168.0.0/24 --sport 53 -j ACCEPT
```

All comes from the internet in the port 53 with destination to the internal network I allow the transfer of packages

Do a test executing the command below:

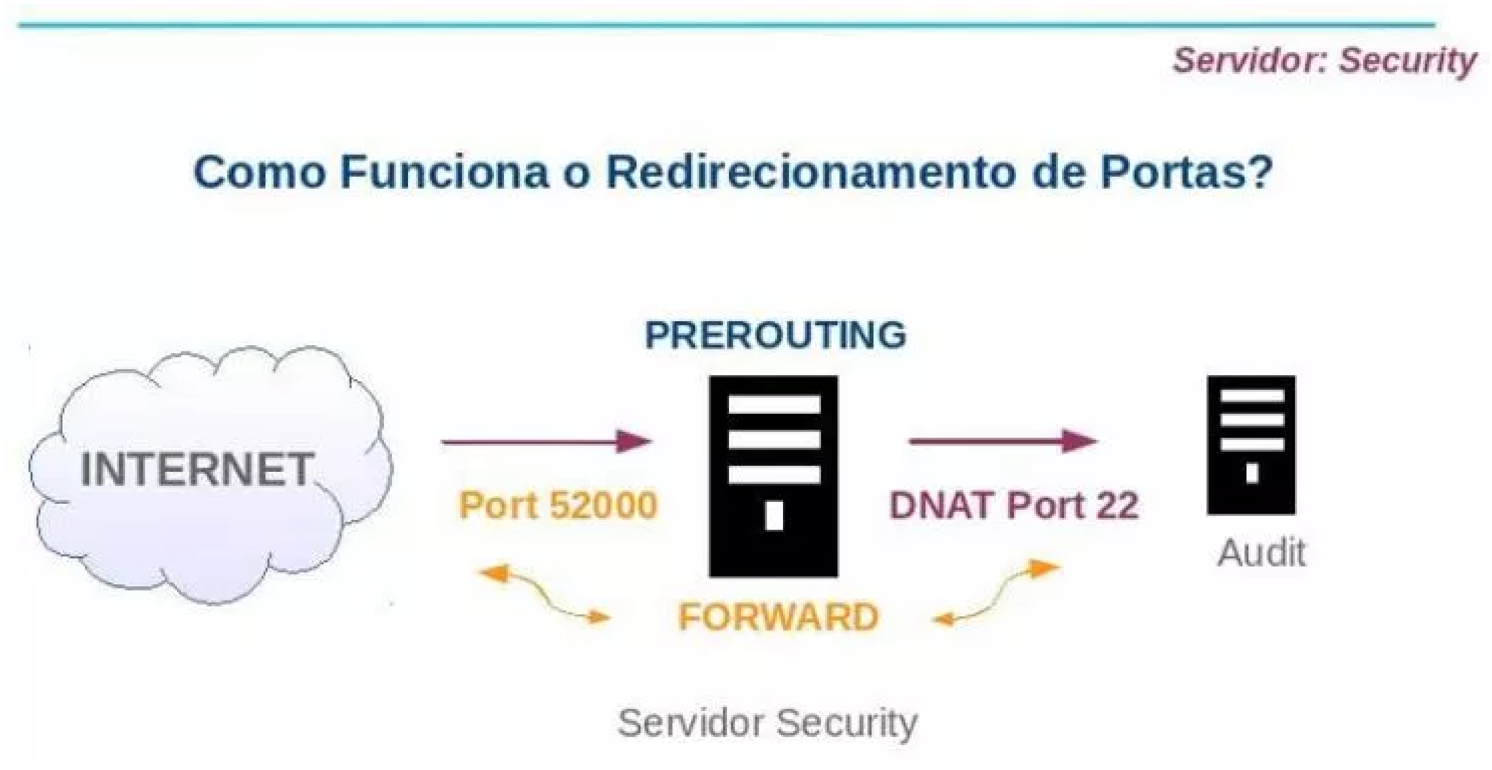
```
apt-get update
```

Let’s see how works port redirects:



9ª → Redirecione o serviço SSH para os servidores internos:

Porta 52000	→	Servidor Audit	→	Porta 22
Porta 53000	→	Servidor FileServer	→	Porta 22
Porta 54000	→	Servidor Storage	→	Porta 22
Porta 55000	→	Servidor WebServerInterno	→	Porta 22
Porta 56000	→	Servidor Datacenter	→	Porta 22



9º Redirect the service ssh to others internal servers:

```
iptables -t nat -A PREROUTING -p tcp -s 0/0 -d 200.100.50.99 --dport 5200 -j DNAT --to 192.168.200
```

All comes from the internet with destination to Security Server in the port 52000 will be redirect to the Audit Server in the port 22.

```
iptables -t filter -A FORWARD -p tcp -s 0/0 -d 192.168.200.30 --dport 22 -j ACCEPT
```

All comes from the internet with destination to the Audit Server in the port 22 I allow the transfer of packages

```
iptables -t filter -A FORWARD -p tcp -s 192.168.200.30 -d 0/0 --sport 22 -j ACCEPT
```

Do a teste using the following command:

```
ssh root@200.100.50.99 -p 5200
```

Customizing the initialization of the firewall:

```
vim /etc/rules
vim /root/firewall/firewall
cp /root/firewall/firewall /etc/init.d
chmod +x /etc/init.d/firewall
insserv /etc/init.d/firewall
/etc/init.d/firewall restart
```

Putting a comment in the rule:

```
iptables -t filter -A INPUT -s 0/0 -d 127.0.0.1 -j ACCEPT -m comment --comment "Accept all loopbac
```

-m (Module)

–comment (function of module)

Creating register fo the logs of iptables:

```
iptables -t filter -A INPUT -s 0/0 -d 100.100.50.99 -p ecp --dport 22 -j LOG --log-prefix '[Access
```

Do an access from another server by SSH e see the log in the Security Server:

```
tail -f /var/log/messages
```

Advertisements

US Seniors On Medicare Are Getting a Big Pay Day

REPORT THIS AD

Share this:

Twitter Facebook

Like

Be the first to like this.

Posted in Sysadmin

← Django Errors

DHCP (Dynamic Host Configuration Protocol) Server →

Leave a Reply

Enter your comment here...

Social

What are you looking for?

Search ...

Testing Applications

- Api Rest Full
- NOC – Network Operations Center
- Simulado LPI-1

- Zabbix Server 3.0

Categories

- AWK (1)
- Certifications (1)
- Containers (2)
- Database (2)
  - MongoDB (1)
  - MySQL (1)
- Development (1)
- Django (3)
  - Errors (1)
  - Tips (1)
- Manipulação Arquivos e Diretórios (3)
- Mental Maps (1)
- Monitoring (5)
  - Zabbix (5)
- Phyton (3)
- PL-SQL (1)
- Presentation (1)
- Projects (1)
- Segurança de Acesso (1)
- Shell Script (47)
  - Models of Scripts \o/ (10)

- SQL (2)
- Sysadmin (71)
- Tips English (7)
- Uncategorized (22)

July 2017

M	T	W	T	F	S	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

« Jun Aug »

Contact Info

Alameda Araguaia, Barueri, São Paulo.
+55 11968653820

Meta

- Register
- Log in
- Entries RSS
- Comments RSS
- WordPress.com