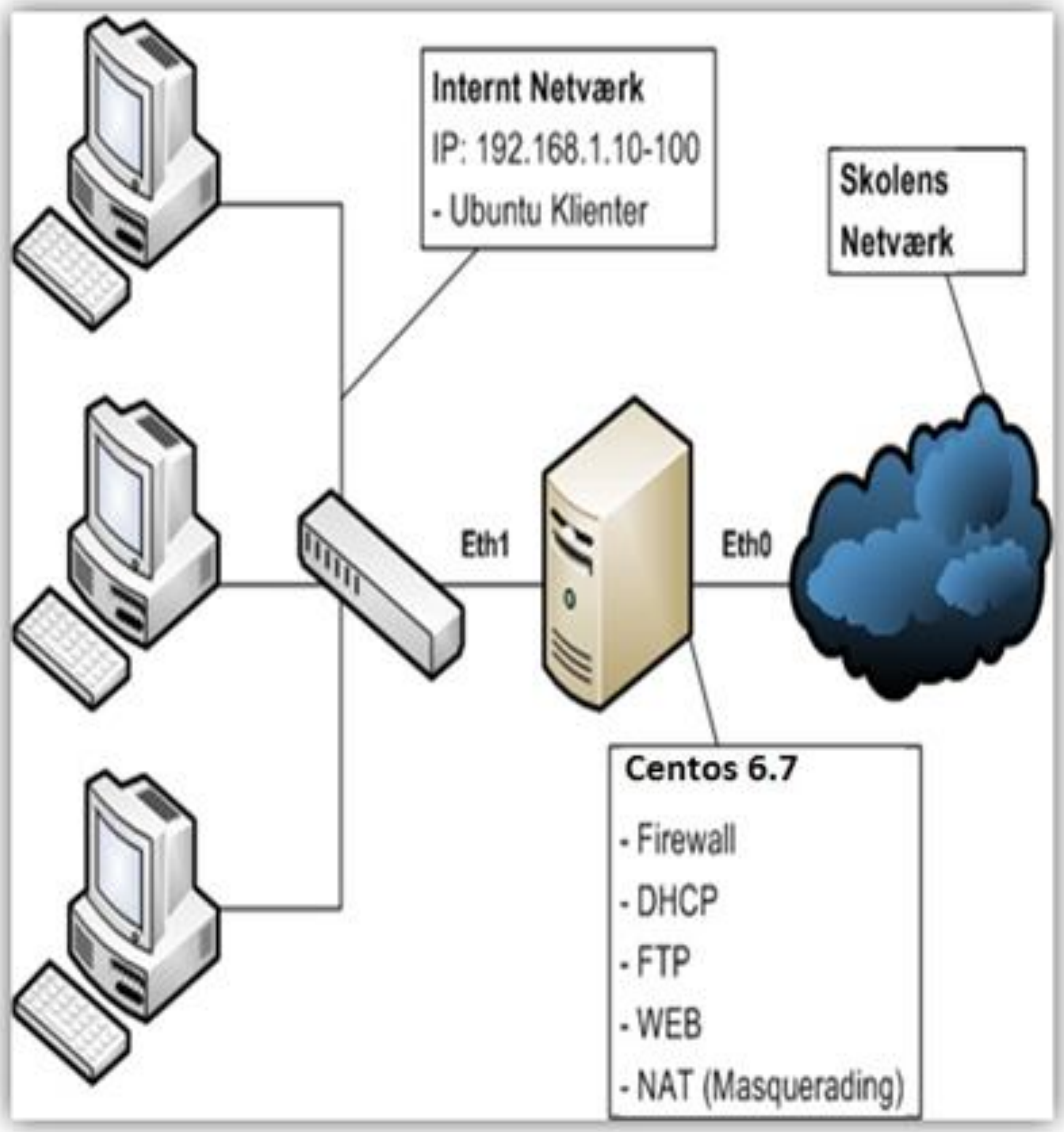

Internt Netværk
IP: 192.168.1.10-100
- Ubuntu Klienter
Skolens Netværk
Eth1
Eth0
Centos 6.7
- Firewall
- DHCP
- FTP
- WEB
- NAT (Masquerading)

Projektoplæg til Linux

Målet med projektet er at beskrive og opbygge et netværk, som er baseret på linux, der tilfredsstiller de krav, der kunne være aktuelle i forhold til et lille nystartet firma, som har 10 medarbejdere og hvor sikkerhed – indtrængen (såvel udefra som indefra), er et væsentlig aspekt. Opgaven bør løses i virtuelt miljø som Vmware.

Netværket skal indeholde følgende enheder:

1. Der skal opstilles en linux Centos 6.7 , som adskiller jeres netværk fra skolens. Den skal fungere som:

- Der skal opsættes RAID 5 på serveren
- Firewall (Linux, den skal have 2 netkort), med pakke iptables
- NAT (masquerading)
- DHCP server for klienter
- At opsætte og anvende ssh og telnet
- Web/FTP server
- Installation af proxy server (Squid)
- Iptables firewalls
- At bruge shell script for at oprette brugere

Følgende IP nummer skal (statisk) tildeles til firewallen:

- Eth0 192.168.1.1
- Eth1 DHCP fra skolens netværk

2. På indersiden af netværket sættes 1. klientmaskiner op med følgende operativsystem:

- Linux ubuntu

Til klientmaskinen skal der tildeles IP adresser dynamisk.

Step-By-Step Configuration of NAT with iptables

This tutorial shows how to set up network-address-translation (NAT) on a Linux system with iptables rules so that the system can act as a gateway and provide internet access to multiple hosts on a local network using a single public IP address. This is achieved by rewriting the source and/or destination addresses of IP packets as they pass through the NAT system.

Requirements:

CPU - PII or more

OS - Any Linux distribution

Software - Iptables

Network Interface Cards: 2

Here is my considerations:

Replace xx.xx.xx.xx with your WAN IP

Replace yy.yy.yy.yy with your LAN IP

(i.e. 192.168.0.0/16, 172.16.0.0/12, 10.0.0.0/8 as suggested by Mr. tzs)

WAN = eth0 with public IP xx.xx.xx.xx

LAN = eth1 with private IP yy.yy.yy.yy/ 255.255.0.0

Step by Step Procedure

Step #1. Add 2 Network cards to the Linux box

Step #2. Verify the Network cards, Wether they installed properly or not

```
ls /etc/sysconfig/network-scripts/ifcfg-eth* | wc -l
```

```
  DEVICE=eth0

NAME=eth0

BOOTPROTO=none

DEFROUTE=yes

ONBOOT=yes

IPADDR=192.168.80.230

PREFIX=24

GATEWAY=192.168.80.2

DNS1=192.168.80.2

echo 1 > /proc/sys/net/ipv4/ip_forward
```

Step #4. Configure eth1 for LAN with a Private IP (Internal private network)

```
cat /etc/sysconfig/network-scripts/ifcfg-eth1
```

```
DEVICE=eth1
BOOTPROTO=none
IPADDR=192.168.1.1
PREFIX=24
DNS1=192.168.80.2
DEFROUTE=yes
NAME=eth1
ONBOOT=yes
```

Step #6. Gateway Configuration

```
cat /etc/sysconfig/network
```

```
    NETWORKING=yes

    HOSTNAME=nat

    GATEWAY=xx.xx.xx.1    # Internet Gateway, provided by the ISP
```

Step #7. DNS Configuration

```
cat /etc/resolv.conf
```

```
    nameserver 203.145.184.13      # Primary DNS Server provided by the ISP

    nameserver 202.56.250.5        # Secondary DNS Server provided by the ISP
```

Step #8. NAT configuration with IP Tables

# Delete and flush. Default table is "filter". Others like "nat" must be explicitly stated.

```
iptables --flush             # Flush all the rules in filter and nat tables

iptables --table nat --flush

iptables --delete-chain
```

# Delete all chains that are not in default filter and nat table

```
iptables --table nat --delete-chain
```

# Set up IP FORWARDing and Masquerading

```
iptables --table nat --append POSTROUTING --out-interface eth0 -j MASQUERADE
```

```
iptables --append FORWARD --in-interface eth1 -j ACCEPT
```

# Enables packet forwarding by kernel

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

#Apply the configuration

```
service iptables restart
```

Step #9. Testing

# Ping the Gateway of the network from client system

```
ping 192.168.2.1
```

Try it on your client systems

```
ping google.com
```

Configuring PCs on the network (Clients)

- All PC's on the private office network should set their "gateway" to be the local private network IP address of the Linux gateway computer.
- The DNS should be set to that of the ISP on the internet.

Windows '95, 2000, XP, Configuration:

- Select "Start" + Settings" + "Control Panel"
- Select the "Network" icon
- Select the tab "Configuration" and double click the component "TCP/IP" for the ethernet card. (NOT the TCP/IP -> Dial-Up Adapter)
- Select the tabs:

o "Gateway": Use the internal network IP address of the Linux box. (192.168.2.1)

o "DNS Configuration": Use the IP addresses of the ISP Domain Name Servers. (Actual internet IP address)

o "IP Address": The IP address (192.168.XXX.XXX - static) and netmask (typically 255.255.0.0 for a small local office network) of the PC can also be set here.

DHCP server for klienter:

For at installerer dhcp server skal du indtaste # yum install dhcp
For at starte dhcp server automatisk skal du indtaste =chkconfig dhcpd on

Opret følgende dhcp config fil i /etc/dhcp/dhcpd.conf

- Dhcpd.conf filen skal indeholder følgende informationer:

```
#
# DHCP Server Configuration file.
#   see /usr/share/doc/dhcp*/dhcpd.conf.sample
#   see 'man 5 dhcpd.conf'
#

# Herunder konfigureres DHCP:
ddns-update-style interim;

# Her sættes subnet og mask, og konfiguration påbegyndes
subnet 192.168.1.0 netmask 255.255.255.0 {

# DHCP range sættes
range 192.168.1.10 192.168.1.100;

# Lease-perioden defineres
default-lease-time 86400;
max-lease-time 86400;

# Default gateway for PC klienterne
option routers 192.168.1.1;

# Broadcast adresse samt mask for PC klienterne
option broadcast-address 192.168.1.255;
option subnet-mask 255.255.255.0;

# DNS for PC klienterne
option domain-name-servers 8.8.8.8;

}
```

Install Telnet Server

Telnet server is available under default yum repositories. Execute following command to install it

```
# yum install telnet-server
```

Enable Telnet Service

Telnet is an xinetd based service, First edit telnet xinetd configuration file **/etc/xinetd.d/telnet** and set disable to **no**.

```
service telnet
{
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/sbin/in.telnetd
        log_on_failure  += USERID
        disable         = no
}
```

Now restart xinetd service

```
# service xinetd restart
```

Connect to Telnet Server

Now connect to Telnet server using telnet client. For this example, we are connecting to Telnet server from windows host

```
c:\> telnet svr1.tecadmin.net

CentOS release 6.5 (Final)
Kernel 2.6.32-431.17.1.el6.i686 on an i686
login: rahul
Password:
Last login: Tue Apr 29 19:19:24 from 192.168.1.115
[rahul@svr1 ~]$
```

Introduction

In this How-To, we will walk you through the installation of a LAMP stack on a CentOS 6.7 based server. Although we are writing this article in the context of CentOS 6.7, a Linux, Apache, Mysql, PHP(LAMP) server is a common installation stack capable of being hosted on many different Operating Systems. Examples of such are Debian and Debian based distributions like Ubuntu, or RHEL and RHEL based distributions such as Fedora or Scientific Linux. You'll see these installations occurring on a variety of hosting platforms such as shared web hosting, VPS hosting, dedicated hosting and cloud hosting.

In the case of this article, we'll be utilizing the YUM package manager associated with the RHEL distribution CentOS.

Prerequisites

A server with CentOS 6.7 installed will take care of the Linux aspect of the LAMP stack install. If you do not have a server, consider a reliable SSD cloud hosting solution from Atlantic.Net.

Installing Apache on CentOS 6.7

Install Apache with the following command to begin the install:

- sudo yum install httpd

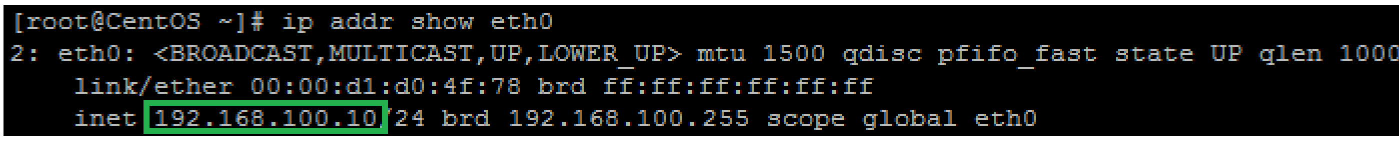
Start Apache with the following command:

- sudo service httpd start

We can now verify Apache is working by opening your browser and entering the URL http://your-server's-address. you should get a blue Apache 2 test page similar to the image below.

Note: If you do not know your IP address, run the following command:

- *ip addr show eth0*



```
[root@CentOS ~]# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:00:d1:d0:4f:78 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.10/24 brd 192.168.100.255 scope global eth0
```

An example of running the command: ip addr show eth0 and getting 192.168.100.10 for the IP address.

In our example we would put http://192.168.100.10 into our browser's address bar.

Opret din egne index.html fil på følgende sti:

```
nano /var/www/html/index.html
```

Her er mine html koder for min hjemmeside med et billede:

```
<HTML>
<HEAD>
<TITLE>Your Title Here</TITLE>
</HEAD>
<BODY BGCOLOR="FFFFFF">
<H1>Min første billede i HTML</H1>

<CENTER><IMG SRC="images.jpg" ALIGN="BOTTOM"> </
CENTER>
<H1> I skal finde et billede for at indsætte jeres hjemmeside</
H1>
</BODY>
</HTML>
```

Brug din fantasi at lave din egne hjemmeside med personlig præg og du kan finde mange eksampler på nettet som, du kan blive insireret af.

vsftpd

Warning: FTP is inherently insecure. If you must use FTP, consider securing your FTP connection with SSL/TLS. Otherwise, it is best to use SFTP, a secure alternative to FTP.

The first two letters of vsftpd stand for "very secure" and the program was built to have strongest protection against possible FTP vulnerabilities.

Step One—Install vsftpd

You can quickly install vsftpd on your virtual private server in the command line:

```
sudo yum install vsftpd
```

We also need to install the FTP client, so that we can connect to an FTP server:

```
sudo yum install ftp
```

Once the files finish downloading, vsftpd will be on your VPS. Generally speaking, the virtual private server is already configured with a reasonable amount of security. However, it does provide access to anonymous users.

Step Two—Configure VSFTP

Once VSFTP is installed, you can adjust the configuration.

Open up the configuration file:

```
sudo vi /etc/vsftpd/vsftpd.conf
```

One primary change you need to make is to change the Anonymous_enable to No:

```
anonymous_enable=NO
```

Prior to this change, vsftpd allowed anonymous, unidentified users to access the VPS's files. This is useful if you are seeking to distribute information widely, but may be considered a serious security issue in most other cases. After that, uncomment the local_enable option, changing it to yes.

```
local_enable=YES
```

Finish up by uncommenting command to chroot_local_user. When this line is set to Yes, all the local users will be jailed within their chroot and will be denied access to any other part of the server.

```
chroot_local_user=YES
```

Finish up by restarting vsftpd:

```
sudo service vsftpd restart
```

In order to ensure that vsftpd runs at boot, run chkconfig:

```
chkconfig vsftpd on
```

Step Three—Access the FTP server

Once you have installed the FTP server and configured it to your liking, you can now access it.
You can reach an FTP server in the browser by typing the domain name into the address bar and logging in with the appropriate ID. Keep in mind, you will only be able to access the user's home directory.
ftp://example.com
Alternatively, you can reach the FTP server through the command line by typing:

```
ftp example.com
```

Then you can use the word, "exit," to get out of the FTP shell.

Step 1: Installing Squid

I'm going to assume that you have a new CentOS server. You can now install Squid with `yum`:

```
yum install squid
```

Step 2: Editing configuration

You can now configure Squid. The configuration file is located at the following path:

```
vi /etc/squid/squid.conf
```

Open this file with your favorite text editor in order to configure Squid settings. You can find an overview of them on the official Squid website.

```
[root@prox ~]#
vi /etc/squid/squid.conf
acl CONNECT method CONNECT
# line 29: add ( define new ACL )
acl lan src 10.0.0.0/24
http_access allow localhost
# line 57: add ( allow defined ACL above )
http_access allow lan
# line 62: change
http_port
8080
# add follows to the end
request_header_access Referer deny all
request_header_access X-Forwarded-For deny all
request_header_access Via deny all
request_header_access Cache-Control deny all
```

```
# define hostname
visible_hostname prox.srv.world
# not display IP address
forwarded_for off
[root@prox ~]#
/etc/rc.d/init.d/squid start
Starting squid:
[ OK ]
[root@prox ~]#
chkconfig squid on
```

Fra <https://www.server-world.info/en/note?os=CentOS_6&p=squid>

Step 3: Open port in firewall

Now open the Squid port in the firewall. The default port is `3128`. If you changed it, naturally, open the port you set Squid to run on:

```
iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 3128 -j ACCEPT
```

Step 3: Restarting service

You can now restart the Squid service:

```
service squid restart
```

To make Squid start every time you start your server, use `chkconfig`:

```
chkconfig squid on
```

In order to see your users' activity, you can use the `tail` command to read the logs:

```
tail -f /var/log/squid/access.log
```

Fra <https://www.vultr.com/docs/how-to-install-squid-proxy-on-centos>

Troubleshooting :
If you not able to browse using proxy settings , Disable the firewall (iptables) and selinux service on your squid proxy server .
Disable firewall (Iptables) »

```
[root@leela ~]# service iptables stop
[root@leela ~]# chkconfig iptables off
```

Disable Selinux » open the file /etc/selinux/config and find the line

```
SELINUX=enforcing
```

and replace with

```
SELINUX=disabled
```

now reboot the server and try step 4.

Configure squid proxy as web filter:

You can restrict user access to particular websites or keywords using access control lists (ACLs) .

» Restricting Access to specific web sites :

For example , we can see how to block facebook.com and gmail.com .

Step 1 » create a file (/etc/squid/blockedsites.squid) and add the site names one per line.

```
[root@leela ~]# cat /etc/squid/blockedsites.squid
#blocked sites
www.facebook.com
www.gmail.com
```

Step 2 » Open the /etc/squid/squid.conf and create a new acl " blocksites" and acl type "dstdomain" in the acl section like the below .

acl Safe_ports port 488 # gss-http acl Safe_ports port 591 # filemaker acl Safe_ports port 777 # multiling http acl CONNECT method CONNECT # ACL blocksites acl blocksites dstdomain "/etc/squid/blockedsites.squid"

```
acl Safe_ports port 488       # gss-http
acl Safe_ports port 591       # filemaker
acl Safe_ports port 777       # multiling http
acl CONNECT method CONNECT
# ACL blocksites
acl blocksites dstdomain "/etc/squid/blockedsites.squid"
```

and add the following line "http_access deny blocksites" to http_section to deny the access to the acl "blocksites" .

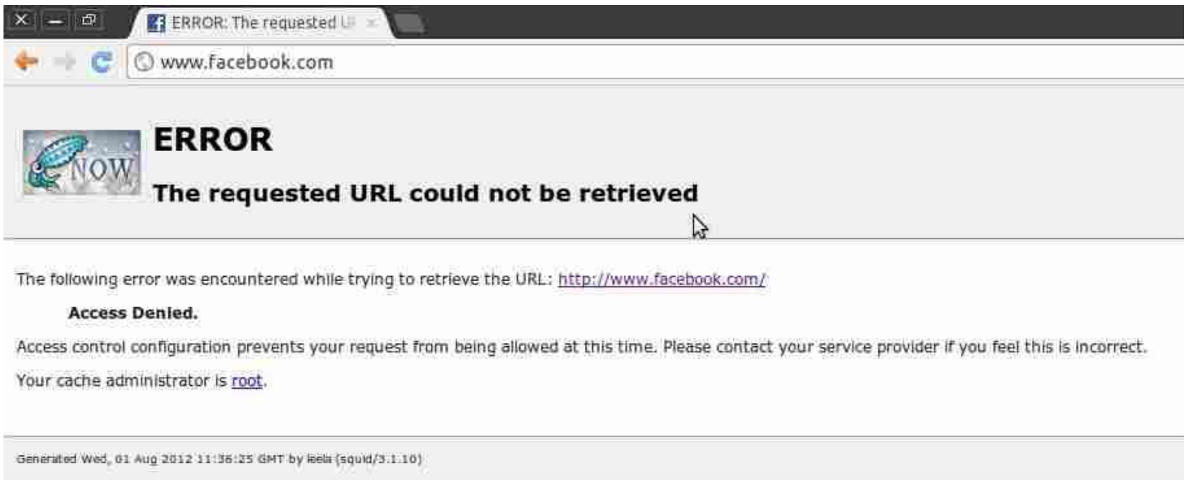
# Recommended minimum Access Permission configuration: # # Only allow cachemgr access from localhost http_access allow manager localhost # Deny access to blocksites ACL http_access deny blocksites

```
# Recommended minimum Access Permission configuration:
#
# Only allow cachemgr access from localhost
http_access allow manager localhost
# Deny access to blocksites ACL
http_access deny blocksites
```

Step 3 » Now restart squid service

```
[root@leela ~]# service squid restart
```

Step 4 » Try to access facebook.com in your browser . you could see the blocked page as below .



ERROR

The requested URL could not be retrieved

The following error was encountered while trying to retrieve the URL: http://www.facebook.com/

Access Denied.

Access control configuration prevents your request from being allowed at this time. Please contact your service provider if you feel this is incorrect.

Your cache administrator is root.

Generated Wed, 01 Aug 2012 11:36:25 GMT by leela (squid/3.1.10)

and check the log file you can see the facebook request is denied .

Fra <http://www.krizna.com/centos/how-to-install-squid-proxy-on-centos-6/>

IP tables:

People always complain about how hard iptables are to understand and configure. I'm not saying it's an easy process but once you get the hang of it, it should seem a little less difficult. Iptables is a user space tool which is used to create rules for packet filtering and NAT modules. Basically, iptables create the firewall for your Internet connection in Linux. If you are new to Linux but will use it on a regular basis, then you must learn how to use iptables as your whole system security is based on it.

Kernel support check

Before you configure iptables, you first have to check if your system has been configured properly. First of all, you must check if the kernel was compiled with iptables support. This can be done with the 'grep' command on the kernel config file. The command:

```
# cat /boot/config-your.kernel.version.here | grep -i "CONFIG_IP_NF"
```

should print some lines ending with '=y' or '=m'. Also, "CONFIG_IP_NF_IPTABLES" line must end with '=m' (that means iptables was compiled as a module).

Iptables check/installing

You must check if you have iptables installed by executing the command:

# rpm -qa | grep iptables.

This should print iptables-your.installed.version and eventually iptables-devel-your.installed.version (this is optional).

If you don't have it on your system, you can easily install it either by downloading the latest rpm package from iptables homepage and executing the command:

# rpm -Uvh iptables-downloaded.version.rpm.

Or by using yum:

# yum install iptables

Where are the main iptables files stored?

/etc/init.d/iptables is the INIT script which is used to start, stop the service and/or to save the rulesets.

/etc/sysconfig/iptables this is the file that holds the saved rulesets.

/sbin/iptables and this is the iptables binary.

Before you actually start configuring the rules, let's take a look at the current configuration:

#iptables -L

By default, there are only three chains which hold exclusive rules: INPUT, OUTPUT and FORWARD. The INPUT chain contains rules for traffic incoming to your server, OUTPUT contains the rules for traffic outgoing from your server to the Internet and FORWARD contains the rules for traffic that will be forwarded to other computers behind yours (if your server is a firewall for a LAN).

Iptables are mostly configured to deal with traffic incoming from the Internet to the Linux server, so the chain INPUT is used. When traffic passes through your Linux kernel, a TARGET is determined based on whether the packet matches a rule in the rulesets or not. The mainly used targets are:

ACCEPT: traffic is allowed to pass through to its destination;

REJECT: traffic is blocked from reaching its destination and a packet is sent back to the sending host with a quick explanation;

DROP: traffic is blocked with no explanation sent back whatsoever.

Configuring iptables rulesets

Before the actual configuration, I insist on giving you a few tips I've learned the hard way so you don't have to. You must keep in mind that the order in which you add rules is everything. For example, if you make the mistake of adding the first rule to deny everything, then, no matter what you set to allow after, it will still be denied.

The second thing you must keep in mind is that the rulesets are in memory and are not saved on disk automatically, so if you reboot your computer without first running the INIT script (or iptables-save) to save them, everything you've worked for will be gone.

Another important thing that applies if you work on your server from a remote PC, through ssh: it's important not to block yourself out so make this your first rule:

# iptables -A INPUT -s 213.10.10.13 -d 192.168.1.1 -p TCP -dport 22 -j ACCEPT

Explanation:

-A: appends the rule to the INPUT chain;

-s: source IP, in this case, the IP you are ssh'ing from;

-d: destination IP, in this case, the server IP;

-p: communication protocol;

--dport: destination port, in our case it's the SSHd port;

-j: stands for 'Jump'. If everything matches, the packet is accepted.

Next, let's set some basic rules for general traffic. One of iptables' features is the ability to determine the state a packet is in. This is the packets' state on a new connection:

NEW: 1st server sends 2nd a SYN packet that it wants to create a new connection.

RELATED: 2nd server receives the SYN packet and sends to 1st server a SYN-ACK packet which tells that everything is alright.

ESTABLISHED: 1st server receives the SYN-ACK packet and sends 2nd server an ACK packet which is the final acknowledgment, the connection finishes establishing and the traffic start between the two servers.

In order for your server to be able to establish TCP connections with other servers, iptables must be configured with the following rules:

# iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

# iptables -A FORWARD -i eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT

# iptables -A OUTPUT -m state --state RELATED,ESTABLISHED

Custom rules

To block IPs:

# iptables -A INPUT -s 213.10.10.13 -j DROP

This rule blocks any incoming traffic from 192.168.1.13.

```
# iptables -A INPUT -d 192.168.1.15 -j REJECT
```

This rule blocks any incoming traffic for LAN computer with IP 192.168.1.15 (behind your server).

To allow IPs:

```
# iptables -A INPUT -s 213.10.10.13 -d 192.168.1.4 -p tcp --dport 21
```

This rule accepts incoming traffic from source IP to destination IP which is a FTP server.

After you've configured all the rules for every port on your local or network service you want to allow (or block) traffic to or from, it's time to block the rest:

```
# iptables -A INPUT -j REJECT

# iptables -A FORWARD -j REJECT
```

These block rules MUST be added last.

To delete a rule, simply write it again but replace the '-A' before the chain with '-D' (Delete).

Saving rulesets

To save your iptables configuration, simply execute the command:

```
# /etc/init.d/iptables save
```

To stop iptables and flush all rules: (careful, use the save INIT script before stopping iptables and flushing rules)

```
# /etc/init.d/iptables stop
```

To start the iptables again and loading the rulesets from /etc/sysconfig/iptables:

# /etc/init.d/iptables start

The End

This is a very, very basic guide. Iptables can be used way beyond these basics, in extremely numerous combinations; therefore, it's not humanly possible to write a complete guide about iptables. However, for a basic security and for one to make an idea about how iptables work, I hope this guide will help someone, someday

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [18:1314]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p udp -m state --state NEW -m udp --dport 137 -j ACCEPT
-A INPUT -p udp -m state --state NEW -m udp --dport 138 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 139 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 21 -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 23 -j ACCEPT
-A FORWARD -i eth1 -j ACCEPT
-A FORWARD -p icmp -j ACCEPT
COMMIT
# Completed on Tue May 24 12:26:47 2016
```