Linux Servers

Paul Cobbaut

Linux Servers

Paul Cobbaut

lt-0.970.129

Published Thu 02 Oct 2008 00:17:40 CEST

Abstract

This book is meant to be used in an instructor-led training. For self-study, the idea is to read this book next to a working Linux computer so you can immediately do every subject, even every command.

This book is aimed towards novice Linux system administrators (and might be interesting and useful for home users that want to know a bit more about their Linux system). However, this book is not meant as an introduction to Linux desktop applications like text editors, browsers, mail clients, multimedia or office applications.

More information and free .pdf available at http://linux-training.be .

Feel free to contact the authors:

- Paul Cobbaut: paul.cobbaut@gmail.com, http://www.linkedin.com/in/cobbaut

Contributors to the Linux Training project are:

- Serge van Ginderachter: serge@ginsys.be, docbook xml and pdf build scripts
- Hendrik De Vloed: hendrik.devloed@ugent.be, buildheader.pl script

We'd also like to thank our reviewers:

- Wouter Verhelst: wouter@grep.be, http://grep.be
- Geert Goossens: goossens.geert@gmail.com, http://www.linkedin.com/in/geertgoossens
- Elie De Brauwer: elie@de-brauwer.be, http://www.de-brauwer.be
- Christophe Vandeplas: christophe@vandeplas.com, http://christophe.vandeplas.com

Table of Contents

List of Tables

Chapter 1. Routers

1.1. terminology

1.1.1. router or firewall

A router is a device that connects two networks. A **firewall** is a device that besides acting as a **router**, also contains (and implements) rules to determine whether packets are allowed to travel from one network to another. A firewall can be configured to block access based on networks, hosts, protocols and ports. Firewalls can also change the contents of packets while forwarding them.

1.1.2. packet forwarding

Packet forwarding means allowing packets to go from one network to another. When a multihomed host is connected to two different networks, and it allows packets to travel from one network to another through its two network interfaces, it is said to have enabled **packet forwarding**.

1.1.3. packet filtering

Packet filtering is very similar to packet forwarding, but every packet is individually tested against rules that decide on allowing or dropping the packet. The rules are stored by iptables.

1.1.4. stateful

A **stateful** firewall is an advancement over stateless firewalls that inspect every individual packet. A stateful firewall will keep a table of active connections, and is knowledgeable enough to recognise when new connections are part of an active session. Linux iptables is a stateful firewall.

1.1.5. NAT (network address translation)

A **NAT** device is a router that is also changing the source and/or target ip-address in packets. It is typically used to connect multiple computers in a private address range (rfc 1918) with the (public) internet. A NAT can hide private addresses from the internet.

It is important to understand that people and vendors do not always use the right term when referring to a certain type of NAT. Be sure you talk about the same thing. We can distuinguish several types of NAT.

1.1.5.1. PAT (port address translation)

NAT often includes PAT. A **PAT** device is a router that is also changing the source and/or target tcp/udp port in packets. PAT is Cisco terminology and is used by **SNAT**, **DNAT**, **masquerading** and **port forwarding** in Linux. RFC 3022 calls it **NAPT** and defines the NAT/PAT combo as "traditional NAT". A device sold to you as a NAT-device will probably do NAT and PAT.

1.1.5.2. SNAT (source network address translation)

A **SNAT** device is changing the source ip-address when a packet passes our NAT. SNAT configuration with iptables includes a fixed target source address.

1.1.5.3. masquerading

Masquerading is a form of SNAT that will hide the (private) source ip-addresses of your private network using a public ip-address. Masquerading is common on dynamic internet interfaces (broadband modem/routers). Masquerade configuration with iptables uses a dynamic target source address.

1.1.5.4. DNAT (destination network address translation)

A **DNAT** device is changing the destination ip-address when a packet passes our NAT.

1.1.5.5. port forwarding

When static DNAT is set up in a way that allows outside connections to enter our private network, then we call it **port forwarding**.

1.2. packet forwarding

1.2.1. about packet forwarding

Packet forwarding means allowing packets to go from one network to another. When a multihomed host is connected to two different networks, and it allows packets to travel from one network to another through its two network interfaces, it is said to have enabled packet forwarding.

1.2.2. /proc/sys/net/ipv4/ip_forward

Whether a host is forwarding packets is defined in **/proc/sys/net/ipv4/ip_forward**. The following screenshot shows how to enable packet forwarding on Linux.

```
[root@RHEL5 ~]# echo 1 > /proc/sys/net/ipv4/ip_forward
```

The next command shows how to disable packet forwarding.

```
[root@RHEL5 ~]# echo 0 > /proc/sys/net/ipv4/ip_forward
```

Use cat to check if packet forwarding is enabled.

```
[root@RHEL5 ~]# cat /proc/sys/net/ipv4/ip_forward
```

1.2.3. /etc/sysctl.conf

By default, most Linux computers are not configured for automatic packet forwarding. To enable packet forwarding whenever the system starts, change the **net.ipv4.ip_forward** variable in **/etc/sysctl.conf** to the value 1.

```
[root@RHEL5 ~]# grep ip_forward /etc/sysctl.conf
net.ipv4.ip_forward = 0
```

1.2.4. Practice: packet forwarding

1. Set up two dsl (Damn Small Linux) machines, one on vmnet1, the other on vmnet8. Make sure they both get an ip-address in the correct subnet. These two machines will be 'left' and 'right' from the 'router'.

2. Set up a RHEL server with two network cards, one on vmnet1, the other on vmnet8. This computer will be the 'router'. Complete the table below with the relevant names, ip-addresses and mac-addresses.

Table 1.1. Packet Forwarding Exercise

	left:	**router:**		**right:**
MAC				
IP				

3. How can you verify whether the RHEL will allow packet forwarding by default or not ? Test that you can ping from the RHEL to the two dsl machines, and from the two dsl machines to the RHEL. Use **arp -a** to make sure you are connected with the correct MAC addresses.

4. Ping from one dsl to the other. Enable and/or disable packet forwarding on the RHEL server and verify what happens to the ping between the two dsl machines. If you do not succeed in pinging between the two dsl machines (on different subnets), then use a sniffer like wireshark or tcpdump to discover the problem.

5. Use wireshark or tcpdump -xx to answer the following questions. Does the source MAC change when a packet passes through the filter ? And the destination MAC ? What about source and destination IP-addresses ?

1.2.5. Solution: packet forwarding

1. Set up two dsl (Damn Small Linux) machines, one on vmnet1, the other on vmnet8. Make sure they both get an ip-address in the correct subnet. These two machines will be 'left' and 'right' from the 'router'.

The configuration of the dsl machines can be similar to the following two screenshots. Both machines must be in a different subnet (here 192.168.187.0/24 and 172.16.122.0/24)

```
root@ttyp1[root]# ifconfig eth0 | grep -A1 eth0
eth0 Link encap:Ethernet  HWaddr 00:0C:29:08:F4:C1
     inet addr:192.168.187.130  Bcast:192.168.187.255  Mask:255.255.255.0
root@ttyp1[root]# route
Kernel IP routing table
Destination    Gateway         Genmask        Flags Metric Ref  Use Iface
192.168.187.0  *               255.255.255.0  U     0      0      0 eth0
default        192.168.187.128 0.0.0.0        UG    0      0      0 eth0
root@ttyp1[root]#
```

```
root@ttyp1[root]# ifconfig eth0 | grep -A1 eth0
eth0 Link encap:Ethernet  HWaddr 00:0C:29:6E:1A:AA
     inet addr:172.16.122.129  Bcast:172.16.122.255  Mask:255.255.255.0
root@ttyp1[root]# route
Kernel IP routing table
Destination    Gateway         Genmask        Flags Metric Ref  Use Iface
172.16.122.0   *               255.255.255.0  U     0      0      0 eth0
default        172.16.122.128  0.0.0.0        UG    0      0      0 eth0
root@ttyp1[root]#
```

2. Set up a RHEL server with two network cards, one on vmnet1, the other on vmnet8. This computer will be the 'router'.

The 'router' can be set up like this screenshot shows.

```
[root@RHEL5 ~]# ifconfig | grep -A1 eth
eth1 Link encap:Ethernet  HWaddr 00:0C:29:8C:90:49
     inet addr:192.168.187.128  Bcast:192.168.187.255  Mask:255.255.255.0
--
eth2 Link encap:Ethernet  HWaddr 00:0C:29:8C:90:53
     inet addr:172.16.122.128  Bcast:172.16.122.255  Mask:255.255.255.0
[root@RHEL5 ~]#
```

Your setup may use different ip and mac addresses than the ones in the table below. This table serves as a reference for the screenshots from this solution to the practice.

Table 1.2. Packet Forwarding Solution

left: dsl	router: RHEL5		right: dsl
00:0c:29:08:f4:c1	00:0c:29:8c:90:49	00:0c:29:8c:90:53	00:0c:29:6e:1a:aa
192.168.187.130	192.168.187.128	172.16.122.128	172.16.122.129

3. How can you verify whether the RHEL will allow packet forwarding by default or not ? Test that you can ping from the RHEL to the two dsl machines, and from the two dsl machines to the RHEL. Use **arp -a** to make sure you are connected with the correct MAC addresses.

This can be done with "**grep ip_forward /etc/sysctl.conf**" (1 is enabled, 0 is disabled).

```
[root@RHEL5 ~]# grep ip_for /etc/sysctl.conf
net.ipv4.ip_forward = 0
```

4. Ping from one dsl to the other. Enable and/or disable packet forwarding on the RHEL server and verify what happens to the ping between the two dsl machines. If you do not succeed in pinging between the two dsl machines (on different subnets), then use a sniffer like ethereal or tcpdump to discover the problem.

Did you forget to add a default gateway to the dsl machines ? Use **route add default gw 'ip-address'**.

You should be able to ping when packet forwarding is enabled (and both default gateways are properly configured). The ping will not work when packet forwarding is disabled or when gateways are not configured correctly.

5. Use wireshark or tcpdump -xx to answer the following questions. Does the source MAC change when a packet passes through the filter ? And the destination MAC ? What about source and destination IP-addresses ?

Both MAC addresses are changed when passing the router. The screenshots below show tcpdump -xx output on the router. The first one is taken on the eth1(vmnet1) interface in the 192.168.187.0/24 network, the second one is from the other interface (eth2 on vmnet8 in 172.16.122.0/24). The first six bytes are the destination MAC, the next six are the source.

```
[root@RHEL5 ~]# tcpdump -xx -i eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
04:18:23.817854 IP 192.168.187.130 > 172.16.122.129: ICMP echo request...
 0x0000:  000c 298c 9049 000c 2908 f4c1 0800 4500
 0x0010:  0054 0000 4000 4001 97ec c0a8 bb82 ac10
 0x0020:  7a81 0800 3b28 a717 0006 8059 d148 d614
 0x0030:  0300 0809 0a0b 0c0d 0e0f 1011 1213 1415
 0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
 0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
04:18:23.817962 IP 172.16.122.129 > 192.168.187.130: ICMP echo reply...
```

```
 0x0000:  000c 2908 f4c1 000c 298c 9049 0800 4500
 0x0010:  0054 d364 0000 3f01 0588 ac10 7a81 c0a8
 0x0020:  bb82 0000 4328 a717 0006 8059 d148 d614
 0x0030:  0300 0809 0a0b 0c0d 0e0f 1011 1213 1415
 0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
 0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
```

```
[root@RHEL5 ~]# tcpdump -xx -i eth2
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth2, link-type EN10MB (Ethernet), capture size 96 bytes
04:18:33.904697 IP 192.168.187.130 > 172.16.122.129: ICMP echo request...
 0x0000:  000c 296e 1aaa 000c 298c 9053 0800 4500
 0x0010:  0054 0000 4000 3f01 98ec c0a8 bb82 ac10
 0x0020:  7a81 0800 2320 a717 0008 8a59 d148 e41a
 0x0030:  0300 0809 0a0b 0c0d 0e0f 1011 1213 1415
 0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
 0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
04:18:33.944514 IP 172.16.122.129 > 192.168.187.130: ICMP echo reply...
 0x0000:  000c 298c 9053 000c 296e 1aaa 0800 4500
 0x0010:  0054 d366 0000 4001 0486 ac10 7a81 c0a8
 0x0020:  bb82 0000 2b20 a717 0008 8a59 d148 e41a
 0x0030:  0300 0809 0a0b 0c0d 0e0f 1011 1213 1415
 0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425
 0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435
```

Chapter 2. Iptables

2.1. about iptables

Iptables is a user-space application that allows a user to configure the Linux kernel's Netfilter. By default there are three tables in the kernel that contain sets of rules. The filter table is used for packet filtering, the NAT table for address translation and the mangle table for special-purpose processing of packets. Series of rules in each table are called a **chain**.

The following screenshot shows how to stop and start iptables.

```
[root@RHEL5 ~]# /etc/init.d/iptables stop
[root@RHEL5 ~]# /etc/init.d/iptables start
[root@RHEL5 ~]#
```

2.2. packet filtering

2.2.1. about packet filtering

Packet filtering is a bit more than packet forwarding. Packet forwarding only uses a routing table to make decisions, the kernel now also uses a list of rules. So with packet filtering, the kernel will inspect each packet and decide based on iptables rules to allow or drop a packet.

2.2.2. filter table

The filter table in iptables has three chains (sets of rules). The INPUT chain is used for any packet coming into the system. The OUTPUT chain is for any packet leaving the system. And the FORWARD chain is for packets that are forwarded (routed) through the system.

The screenshot below shows how to list the filter table and all its rules.

```
[root@RHEL5 ~]# iptables -t filter -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[root@RHEL5 ~]#
```

As you can see, all three chains in the filter table are set to ACCEPT everything. ACCEPT is the default behaviour.

2.2.3. Changing default policy rules

To start, let's set the default policy for all three chains to drop everything. Note that you might lose your connection when typing this over ssh ;-).

```
[root@RHEL5 ~]# iptables -P INPUT DROP
[root@RHEL5 ~]# iptables -P FORWARD DROP
[root@RHEL5 ~]# iptables -P OUTPUT DROP
```

Next, we allow the server to use its own loopback device (this allows the server to access its services running on localhost). We first append a rule to the INPUT chain to allow (ACCEPT) traffic from the lo (loopback) interface, then we do the same to allow packets to leave the system through the loopback interface.

```
[root@RHEL5 ~]# iptables -A INPUT -i lo -j ACCEPT
[root@RHEL5 ~]# iptables -A OUTPUT -o lo -j ACCEPT
```

Looking at the filter table again (omitting -t filter because it is the default table).

```
[root@RHEL5 ~]# iptables -nL
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
```

2.2.4. Allowing ssh over eth0

This example show how to add two rules to allow ssh access to your system from outside.

```
[root@RHEL5 ~]# iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
[root@RHEL5 ~]# iptables -A OUTPUT -o eth0 -p tcp --sport 22 -j ACCEPT
```

The filter table will look something like this screenshot (note that -v is added for more verbose output).

```
[root@RHEL5 ~]# iptables -nvL
Chain INPUT (policy ACCEPT 7 packets, 609 bytes)
 pkts bytes target prot opt in    out   source     destination
```

```
    0     0 ACCEPT all  --  lo    *     0.0.0.0/0  0.0.0.0/0
    0     0 ACCEPT tcp  --  eth0  *     0.0.0.0/0  0.0.0.0/0  tcp dpt:22

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

Chain OUTPUT (policy ACCEPT 3 packets, 228 bytes)
 pkts bytes target prot opt in    out   source     destination
    0     0 ACCEPT all  --  *     lo    0.0.0.0/0  0.0.0.0/0
    0     0 ACCEPT tcp  --  *     eth0  0.0.0.0/0  0.0.0.0/0  tcp spt:22
[root@RHEL5 ~]#
```

2.2.5. Allowing access from a subnet

This example shows how to allow access from any computer in the 10.1.1.0/24 network, but only through eth1. There is no port (application) limitation here.

```
[root@RHEL5 ~]# iptables -A INPUT -i eth1 -s 10.1.1.0/24 -p tcp -j ACCEPT
[root@RHEL5 ~]# iptables -A OUTPUT -o eth1 -d 10.1.1.0/24 -p tcp -j ACCEPT
```

Together with the previous examples, the policy is expanding.

```
[root@RHEL5 ~]# iptables -nvL
Chain INPUT (policy ACCEPT 7 packets, 609 bytes)
 pkts bytes target prot opt in    out   source      destination
    0     0 ACCEPT all  --  lo    *     0.0.0.0/0   0.0.0.0/0
    0     0 ACCEPT tcp  --  eth0  *     0.0.0.0/0   0.0.0.0/0  tcp dpt:22
    0     0 ACCEPT tcp  --  eth1  *     10.1.1.0/24 0.0.0.0/0

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

Chain OUTPUT (policy ACCEPT 3 packets, 228 bytes)
 pkts bytes target prot opt in    out   source      destination
    0     0 ACCEPT all  --  *     lo    0.0.0.0/0   0.0.0.0/0
    0     0 ACCEPT tcp  --  *     eth0  0.0.0.0/0   0.0.0.0/0  tcp spt:22
    0     0 ACCEPT tcp  --  *     eth1  0.0.0.0/0   10.1.1.0/24
```

2.2.6. iptables save

Use **iptables save** to automatically implement these rules when the firewall is (re)started.

```
[root@RHEL5 ~]# /etc/init.d/iptables save
Saving firewall rules to /etc/sysconfig/iptables:          [  OK  ]
[root@RHEL5 ~]#
```

2.2.7. scripting example

You can write a simple script for these rules. Below is an example script that implements the firewall rules that you saw before in this chapter.

```
#!/bin/bash
# first cleanup everything
iptables -t filter -F
iptables -t filter -X
iptables -t nat -F
iptables -t nat -X

# default drop
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP

# allow loopback device
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# allow ssh over eth0 from outside to system
iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 22 -j ACCEPT

# allow any traffic from 10.1.1.0/24 to system
iptables -A INPUT -i eth1 -s 10.1.1.0/24 -p tcp -j ACCEPT
iptables -A OUTPUT -o eth1 -d 10.1.1.0/24 -p tcp -j ACCEPT
```

2.2.8. Allowing ICMP(ping)

When you enable iptables, you will get an **'Operation not permitted'** message when trying to ping other hosts.

```
[root@RHEL5 ~# ping 192.168.187.130
PING 192.168.187.130 (192.168.187.130) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
```

The screenshot below shows you how to setup iptables to allow a ping from or to your machine.

```
[root@RHEL5 ~]# iptables -A INPUT -p icmp --icmp-type any -j ACCEPT
[root@RHEL5 ~]# iptables -A OUTPUT -p icmp --icmp-type any -j ACCEPT
```

The previous two lines do not allow other computers to route ping messages through your router, because it only handles INPUT and OUTPUT. For routing of ping, you will need to enable it on the FORWARD chain. The following command enables routing of icmp messages between networks.

```
[root@RHEL5 ~]# iptables -A FORWARD -p icmp --icmp-type any -j ACCEPT
```

2.2.9. Practice: packet filtering

1. Make sure you can ssh to your router-system when iptables is active.

2. Make sure you can ping to your router-system when iptables is active.

3. Define one of your networks as 'internal' and the other as 'external'. Configure the router to allow visits to a website (http) to go from the internal network to the external network (but not in the other direction).

4. Make sure the internal network can ssh to the external, but not the other way around.

2.3. network address translation

2.3.1. about NAT

A NAT device is a router that is also changing the source and/or target ip-address in packets. It is typically used to connect multiple computers in a private address range with the (public) internet. A NAT can hide private addresses from the internet.

NAT was developed to mitigate the use of real ip addresses, to allow private address ranges to reach the internet and back, and to not disclose details about internal networks to the outside.

The nat table in iptables adds two new chains. PREROUTING allows altering of packets before they reach the INPUT chain. POSTROUTING allows altering packets after they exit the OUTPUT chain.

Use **iptables -t nat -nvL** to look at the NAT table. The screenshot below shows an empty NAT table.

```
[root@RHEL5 ~]# iptables -t nat -nL
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[root@RHEL5 ~]#
```

2.3.2. SNAT (Source NAT)

The goal of source nat is to change the source address inside a packet before it leaves the system (e.g. to the internet). The destination will return the packet to the NAT-device. This means our NAT-device will need to keep a table in memory of all the packets it changed, so it can deliver the packet to the original source (e.g. in the private network).

Because SNAT is about packets leaving the system, it uses the POSTROUTING chain.

Here is an example SNAT rule. The rule says that packets coming from 10.1.1.0/24 network and exiting via eth1 will get the source ip-address set to 11.12.13.14. (Note that this is a one line command!)

```
iptables -t nat -A POSTROUTING -o eth1 -s 10.1.1.0/24 -j SNAT \
--to-source 11.12.13.14
```

Of course there must exist a proper iptables filter setup to allow the packet to traverse from one network to the other.

2.3.3. SNAT example setup

This example script uses a typical nat setup. The internal (eth0) network has access via SNAT to external (eth1) webservers (port 80).

```
#!/bin/bash
#
# iptables script for simple classic nat websurfing
# eth0 is internal network, eth1 is internet
#
echo 0 > /proc/sys/net/ipv4/ip_forward
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A FORWARD -i eth0 -o eth1 -s 10.1.1.0/24 -p tcp \
--dport 80 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -d 10.1.1.0/24 -p tcp \
--sport 80 -j ACCEPT
iptables -t nat -A POSTROUTING -o eth1 -s 10.1.1.0/24 -j SNAT \
--to-source 11.12.13.14
echo 1 > /proc/sys/net/ipv4/ip_forward
```

2.3.4. IP masquerading

IP masquerading is very similar to SNAT, but is meant for dynamic interfaces. Typical example are broadband 'router/modems' connected to the internet and receiving a different ip-address from the isp, each time they are cold-booted.

The only change needed to convert the SNAT script to a masquerading is one line.

```
iptables -t nat -A POSTROUTING -o eth1 -s 10.1.1.0/24 -j MASQUERADE
```

2.3.5. DNAT (Destination NAT)

DNAT is typically used to allow packets from the internet to be redirected to an internal server (in your DMZ) and in a private address range that is inaccessible directly form the internet.

This example script allows internet users to reach your internal (192.168.1.99) server via ssh (port 22).

```
#!/bin/bash
#
# iptables script for DNAT
# eth0 is internal network, eth1 is internet
#
echo 0 > /proc/sys/net/ipv4/ip_forward
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -A FORWARD -i eth0 -o eth1 -s 10.1.1.0/24 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -p tcp --dport 22 -j ACCEPT
iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 22 \
-j DNAT --to-destination 192.168.1.99
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Chapter 3. Cloning a Linux Server

3.1. About cloning

You can have distinct goals for cloning a server. For instance a clone can be a cold iron backup system used for manual disaster recovery of a service. Or a clone can be created to serve in a test environment. Or you might want to make an almost identical server. Let's take a look at some offline and online ways to create a clone of a Linux server.

3.2. About offline cloning

The term offline cloning is used when you power off the running Linux server to create the clone. This method is easy since we don't have to consider open files and we don't have to skip virtual file systems like **/dev** or **/sys** . The offline cloning method can be broken down into these steps:

```
1. Boot source and target server with a bootable CD
2. Partition, format and mount volumes on the target server
3. Copy files/partitions from source to target over the network
```

The first step is trivial. The second step is explained in the Disk Management chapter. For the third step, you can use a combination of **ssh** or **netcat** with **cp**, **dd**, **dump** and **restore**, **tar**, **cpio**, **rsync** or even **cat**.

3.3. Offline cloning example

We have a working Red Hat Enterprise Linux 5 server, and we want a perfect copy of it on newer hardware. First thing to do is discover the disk layout.

```
[root@RHEL5 ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda2              15G  4.5G  9.3G  33% /
/dev/sda1              99M   31M   64M  33% /boot
```

The **/boot** partition is small but big enough. If we create an identical partition, then **dd** should be a good cloning option. Suppose the / partition needs to be enlarged on the target system. The best option then is to use a combination of **dump** and **restore**. Remember that dd copies blocks, whereas dump/restore copies files.

The first step to do is to boot the target server with a live CD and partition the target disk. To do this we use the Red Hat Enterprise Linux 5 install CD. At the CD boot prompt we type "linux rescue". The cd boots into a root console where we can use fdisk to discover and prepare the attached disks.

When the partitions are created and have their filesystem, then we can use dd to copy the /boot partition.

```
ssh root@192.168.1.40 "dd if=/dev/sda1" | dd of=/dev/sda1
```

Then we use a dump and restore combo to copy the / partition.

```
mkdir /mnt/x
mount /dev/sda2 /mnt/x
cd /mnt/x
ssh root@192.168.1.40 "dump -0 -f - /" | restore -r -f -
```

Chapter 4. Introduction to Samba

4.1. Verify installed version

To see the version of samba installed on RedHat, use rpm -qa. Looks like **samba** 3 in the screenshot here, version 3.0.10.

```
[paul@RHEL4b ~]$  rpm -qa | grep samba
samba-common-3.0.10-1.4E.9
samba-client-3.0.10-1.4E.9
system-config-samba-1.2.21-1
samba-swat-3.0.10-1.4E.9
samba-3.0.10-1.4E.9
[paul@RHEL4b ~]$
```

Use dpkg -l on Debian or Ubuntu. Our Feisty Fawn here uses Samba 3.0.24

```
paul@laika:~$ dpkg -l | grep samba
ii  samba-common    3.0.24-2ubuntu1.2    Samba common files used by both the...
paul@laika:~$
```

4.2. Installing Samba

Samba is installed by default on Red Hat Enterprise Linux. If Samba is not yet installed, then the easiest way is to use the graphical menu (Applications -- System Settings -- Add/Remove Applications) and select "Windows File Server" in the Server section. The non-graphical way is to either use rpm -i followed by the samba-version.rpm file.

```
[paul@RHEL4b ~]$  rpm -i samba-3.0.10-1.4E.9.rpm
```

Or if you have a subscription to RHN, then **up2date** is the tool to use.

```
[paul@RHEL4b ~]$  up2date -i samba
```

Ubuntu and Debian users can use the aptitude program.

```
paul@laika:~$ aptitude install samba-server
```

4.3. Documentation

Obviously there are manual pages for Samba. Don't forget **man smb.conf**.

```
[root@RHEL4b samba]# apropos samba
cupsaddsmb       (8)  - export printers to samba for windows clients
lmhosts          (5)  - The Samba NetBIOS hosts file
net              (8)  - Tool for administration of Samba and remote CIFS servers
pdbedit          (8)  - manage the SAM database (Database of Samba Users)
samba            (7)  - A Windows SMB/CIFS fileserver for UNIX
smb.conf [smb]   (5)  - The configuration file for the Samba suite
smbpasswd        (5)  - The Samba encrypted password file
smbstatus        (1)  - report on current Samba connections
swat             (8)  - Samba Web Administration Tool
tdbbackup        (8)  - tool for backing up and ... of samba .tdb files
[root@RHEL4b samba]#
```

Samba comes with excellent documentation in html and pdf format (and also as a free download from Samba.org and are for sale as a printed book). Red Hat Enterprise Linux installs the html and pdf version in /usr/share/doc by default.

```
[paul@RHEL4b ~]$ locate Samba-HOWTO-Collection.pdf
/usr/share/doc/samba-3.0.10/Samba-HOWTO-Collection.pdf
```

Ubuntu packages the docs as a seperate package from Samba.

```
root@laika:~# aptitude search samba | grep -i documentation
i   samba-doc                       - Samba documentation
i   samba-doc-pdf                   - Samba documentation (PDF format)
root@laika:~# find /usr/share/doc/samba-doc-pdf | grep -i howto
/usr/share/doc/samba-doc-pdf/Samba3-HOWTO.pdf.gz
```

Besides the howto, there is also an excellent book called **Samba by example** (again available as book in shops, and as a free pdf and html).

4.4. smb.conf

Samba configuration is done in the **smb.conf** file. The file can be edited manually, or you can use a web based interface like webmin or swat to manage it. The file is usually located in /etc/samba. You can find the exact location with **smbd -b**.

```
[root@RHEL4b ~]# smbd -b | grep CONFIGFILE
CONFIGFILE: /etc/samba/smb.conf
[root@RHEL4b ~]#
```

The default smb.conf file contains a lot of examples with explanations.

```
[paul@RHEL4b ~]$ ls -l /etc/samba/smb.conf
-rw-r--r--  1 root root 10836 May 30 23:08 /etc/samba/smb.conf
(...)
paul@laika:~$ ls -l /etc/samba/smb.conf
-rw-r--r-- 1 root root 10515 2007-05-24 00:21 /etc/samba/smb.conf
```

Below is an example of a very minimalistic smb.conf. It allows samba to start, and to be visible to other computers (Microsoft shows computers in Network Neighborhood or My Network Places).

```
[paul@RHEL4b ~]$ cat /etc/samba/smb.conf
[global]
workgroup = WORKGROUP
[firstshare]
path = /srv/samba/public
[paul@RHEL4b ~]$
```

Below is a screenshot of the **net view** command on Microsoft Windows XP sp2. It shows how the Samba server with the minimalistic smb.conf is visible to Microsoft computers nearby.

```
C:\Documents and Settings\paul>net view
Server Name            Remark

-------------------------------------------------------------------------------
\\RHEL4B               Samba 3.0.10-1.4E.9
\\W2000
\\WINXP
The command completed successfully.
```

Some parameters in smb.conf can get a long list of values behind them. You can continue a line (for clarity) on the next by ending the line with a backslash.

```
valid users = Serena, Venus, Lindsay \
              Kim, Justine, Sabine \
              Amelie, Marie, Suzanne
```

Curious but true, smb.conf accepts synonyms like **create mode** and **create mask**, and sometimes minor spelling errors like **browsable** and **browseable**. And on occasion you can even switch words, the **guest only** parameter is identical to **only guest**.

4.5. testparm

To verify the syntax of the smb.conf file, you can use testparm.

```
[paul@RHEL4b ~]$ testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[firstshare]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions

[paul@RHEL4b ~]$
```

An interesting option is **testparm -v**, which will output all the global options with their default value. The remark seen by the **net view** command is the default value for the "server string" option. Simply adding this value to the global section in smb.conf and restarting samba will change the option. After a while, the changed option is visible on the Microsoft computers.

```
C:\Documents and Settings\paul>net view
Server Name            Remark
```

```
-------------------------------------------------------------------------------
\\RHEL4B               Public File Server
\\W2000
\\WINXP
The command completed successfully.
```

The samba daemons are constantly (once every 60 seconds) checking the smb.conf file, so it is good practice to keep this file small. But it is also good practice to document your samba configuration, and to explicitly set options that have the same default values. The **testparm -s** option allows you to do both. It will output the smallest possible samba configuration file, while retaining all your settings. The idea is to have your samba configuration in another file (like smb.conf.full) and let testparm parse this for you. The screenshot below shows you how. First the smb.conf.full file with the explicitly set option workgroup to WORKGROUP.

```
[root@RHEL4b samba]# cat smb.conf.full
[global]
workgroup = WORKGROUP

# This is a demo of a documented smb.conf
# These two lines are removed by testparm -s

server string = Public Test Server

[firstshare]
path = /srv/samba/public
```

Next, we execute testparm with the -s option, and redirect stdout to the real smb.conf file.

```
[root@RHEL4b samba]# testparm -s smb.conf.full > smb.conf
Load smb config files from smb.conf.full
Processing section "[firstshare]"
Loaded services file OK.
```

And below is the end result. The two comment lines and the default option are no longer there.

```
[root@RHEL4b samba]# cat smb.conf
# Global parameters
[global]
server string = Public Test Server

[firstshare]
path = /srv/samba/public
[root@RHEL4b samba]#
```

4.6. Samba daemons

Samba 3 consists of three daemons, they are named **nmbd**, **smbd** and **winbind**. The **nmbd** daemon takes care of all the names and naming. It registers and resolves names, and handles browsing. It should be the first daemon to start. The **smbd** daemon manages file transfers and authentication. It should be started after nmbd. The **winbind** daemon is only started to handle Microsoft Windows domain membership.

You can start the daemons by invoking **/etc/init.d/smb start** (some systems use **/etc/init.d/samba**) on any linux. Red Hat derived systems are happy with **service smb start**.

```
[root@RHEL4b ~]# /etc/init.d/smb start
Starting SMB services:                                      [  OK  ]
Starting NMB services:                                      [  OK  ]
[root@RHEL4b ~]# service smb restart
Shutting down SMB services:                                 [  OK  ]
Shutting down NMB services:                                 [  OK  ]
Starting SMB services:                                      [  OK  ]
Starting NMB services:                                      [  OK  ]
[root@RHEL4b ~]#
```

4.7. smbclient

With **smbclient** you can see browsing and share information from your smb server. It will display all your shares, your workgroup, and the name of the Master Browser. The -N switch is added to avoid having to enter an empty password. The -L switch is followed by the name of the host to check.

```
[root@RHEL4b init.d]# smbclient -NL rhel4b
Anonymous login successful
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]

Sharename       Type      Comment
---------       ----      -------
firstshare      Disk
IPC$            IPC       IPC Service (Public Test Server)
ADMIN$          IPC       IPC Service (Public Test Server)
Anonymous login successful
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]

Server               Comment
---------            -------
RHEL4B               Public Test Server
WINXP

Workgroup            Master
---------            -------
WORKGROUP            WINXP
```

The screenshot below uses smbclient to display information about a remote smb server (in this case a Windows XP machine).

```
[root@RHEL4b init.d]# smbclient -NL winxp
Anonymous login successful
Domain=[WORKGROUP] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]

Sharename       Type      Comment
---------       ----      -------
Error returning browse list: NT_STATUS_ACCESS_DENIED
Anonymous login successful
Domain=[WORKGROUP] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]
```

```
Server               Comment
---------            -------
RHEL4B               Public Test Server
W2000
WINXP

Workgroup            Master
---------            -------
WORKGROUP            WINXP
```

4.8. smbtree

Another useful tool to troubleshoot Samba or simply to browse the SMB network is **smbtree**. In its simplest form, smbtree will do an anonymous browsing on the local subnet. displaying all SMB computers and (if authorized) their shares.

Let's take a look at two screenshots of smbtree in action (with blank password). The first one is taken immediately after booting four different computers (one MS Windows 2000, one MS Windows XP, one MS Windows 2003 and one RHEL 4 with Samba 3.0.10).

```
[paul@RHEL4b ~]$ smbtree
Password:
WORKGROUP
PEGASUS
 \\WINXP
 \\RHEL4B                        Pegasus Domain Member Server
Error connecting to 127.0.0.1 (Connection refused)
cli_full_connection: failed to connect to RHEL4B<20> (127.0.0.1)
 \\HM2003
[paul@RHEL4b ~]$
```

The information displayed in the previous screenshot looks incomplete. The browsing elections are still ongoing, the browse list is not yet distributed to all clients by the (to be elected) browser master. The next screenshot was taken about one minute later. And it shows even less.

```
[paul@RHEL4b ~]$ smbtree
Password:
WORKGROUP
 \\W2000
[paul@RHEL4b ~]$
```

So we wait a while, and then run smbtree again, this time it looks a lot nicer.

```
[paul@RHEL4b ~]$ smbtree
Password:
WORKGROUP
 \\W2000
PEGASUS
 \\WINXP
 \\RHEL4B                        Pegasus Domain Member Server
  \\RHEL4B\ADMIN$                 IPC Service (Pegasus Domain Member Server)
  \\RHEL4B\IPC$                   IPC Service (Pegasus Domain Member Server)
  \\RHEL4B\domaindata             Active Directory users only
 \\HM2003
[paul@RHEL4b ~]$ smbtree --version
Version 3.0.10-1.4E.9
[paul@RHEL4b ~]$
```

I added the version number of smbtree in the previous screenshot, to show you the difference when using the latest version of smbtree (below a screenshot taken from Ubuntu Feisty Fawn). The latest version shows a more complete overview of machines and shares.

```
paul@laika:~$ smbtree --version
Version 3.0.24
paul@laika:~$ smbtree
Password:
WORKGROUP
 \\W2000
  \\W2000\firstshare
  \\W2000\C$             Default share
  \\W2000\ADMIN$         Remote Admin
  \\W2000\IPC$           Remote IPC
PEGASUS
 \\WINXP
cli_rpc_pipe_open: cli_nt_create failed on pipe \srvsvc to machine WINXP.
Error was NT_STATUS_ACCESS_DENIED
 \\RHEL4B                        Pegasus Domain Member Server
  \\RHEL4B\ADMIN$                 IPC Service (Pegasus Domain Member Server)
  \\RHEL4B\IPC$                   IPC Service (Pegasus Domain Member Server)
  \\RHEL4B\domaindata             Active Directory users only
 \\HM2003
cli_rpc_pipe_open: cli_nt_create failed on pipe \srvsvc to machine HM2003.
Error was NT_STATUS_ACCESS_DENIED
paul@laika:~$
```

The previous screenshot also provides useful errors on why we cannot see shared info on computers winxp and w2003. Let us try the old smbtree version on our RHEL server, but this time with Administrator credentials (which are the same on all computers).

```
[paul@RHEL4b ~]$ smbtree -UAdministrator%Stargate1
WORKGROUP
  \\W2000
PEGASUS
  \\WINXP
    \\WINXP\C$             Default share
    \\WINXP\ADMIN$         Remote Admin
    \\WINXP\share55
    \\WINXP\IPC$           Remote IPC
  \\RHEL4B                 Pegasus Domain Member Server
    \\RHEL4B\ADMIN$        IPC Service (Pegasus Domain Member Server)
    \\RHEL4B\IPC$          IPC Service (Pegasus Domain Member Server)
    \\RHEL4B\domaindata    Active Directory users only
  \\HM2003
    \\HM2003\NETLOGON      Logon server share
    \\HM2003\SYSVOL        Logon server share
    \\HM2003\WSUSTemp       A network share used by Local Publishing ...
    \\HM2003\ADMIN$         Remote Admin
    \\HM2003\tools
    \\HM2003\IPC$           Remote IPC
    \\HM2003\WsusContent    A network share to be used by Local ...
    \\HM2003\C$             Default share
[paul@RHEL4b ~]$
```

As you can see, this gives a very nice overview of all SMB computers and their shares.

4.9. Samba Web Administration Tool (SWAT)

Samba comes with a web based tool to manage your samba configuration file. The tool is accessible with a web browser on port 901 of the host system. To enable the tool, first find out whether your system is using the inetd or the xinetd superdaemon.

```
 [root@RHEL4b samba]# ps fax | grep inet
 15026 pts/0    S+     0:00                      \_ grep inet
  2771 ?        Ss     0:00 xinetd -stayalive -pidfile /var/run/xinetd.pid
 [root@RHEL4b samba]#
```

Then edit the inetd.conf or change the disable = yes line in /etc/xinetd.d/swat to disable = no.

```
[root@RHEL4b samba]# cat /etc/xinetd.d/swat
# default: off
# description: SWAT is the Samba Web Admin Tool. Use swat \
#              to configure your Samba server. To use SWAT, \
#              connect to port 901 with your favorite web browser.
service swat
{
 port            = 901
 socket_type     = stream
 wait            = no
 only_from       = 127.0.0.1
 user            = root
 server          = /usr/sbin/swat
 log_on_failure  += USERID
 disable         = no
}
[root@RHEL4b samba]# /etc/init.d/xinetd restart
Stopping xinetd:                                           [  OK  ]
Starting xinetd:                                           [  OK  ]
[root@RHEL4b samba]#
```

Be careful when using SWAT, it erases alle your manually edited comments in smb.conf.

4.10. Practice

0. !! Make sure you know your student number, anything *ANYTHING* you name must include your student number!

1. Verify that you can logon to a Linux/Unix computer. Write down the name and ip address of this machine.

2. Do the same for all the other (virtual) machines available to you.

3. Verify networking by pinging the machines, if you like names, edit the appropriate hosts files.

4. Make sure Samba is installed, write down the version of Samba.

5. Open the Official Samba-3 howto pdf file that is installed on your computer. How many A4 pages is this file ? Then look at the same pdf on samba.org, it is updated regularly.

6. Take a backup copy of the original smb.conf, name it smb.conf.orig

7. Enable SWAT and take a look at it.

8. Stop the Samba server.

9. Create a minimalistic smb.conf.minimal and test it with testparm.

10. Start Samba with your minimal smb.conf.

11. Verify with smbclient that your Samba server works.

12. Verify that another (Microsoft) computer can see your Samba server.

13. Browse the network with net view and smbtree.

14. Change the "Server String" parameter in smb.conf. How long does it take before you see the change (net view, smbclient, My Network Places,...) ?

15. Will restarting Samba after a change to smb.conf speed up the change ?

16. Which computer is the master browser master in your workgroup ? What is the master browser ?

17. If time permits (or if you are waiting for other students to finish this practice), then install a sniffer (ethereal/wireshark) and watch the browser elections.

Chapter 5. Simple Samba File Servers

5.1. Read Only File Server

Let's start with setting up a very simple read only file server with Samba. Everyone (even anonymous guests) will receive read access.

The first step is to create a directory and put some test files in it.

```
[root@RHEL4b samba]# mkdir -p /srv/samba/readonly
[root@RHEL4b samba]# ls -l /srv/samba/
total 4
drwxr-xr-x  2 root root 4096 Jun 22 11:07 readonly
[root@RHEL4b samba]# cd /srv/samba/readonly/
[root@RHEL4b readonly]# echo "It is cold today." > winter.txt
[root@RHEL4b readonly]# echo "It is hot today." > summer.txt
[root@RHEL4b readonly]# ll
total 8
-rw-r--r--  1 root root 17 Jun 22 11:13 summer.txt
-rw-r--r--  1 root root 18 Jun 22 11:13 winter.txt
[root@RHEL4b readonly]#
```

Linux will always require a user account before giving access to files (the files in our example above are owned by root). So we will create a user for our readonly file server and make this user the owner of the directory and all of its files. (Strictly speaking, you can setup a Samba read only file server without creating an extra user account).

```
[root@RHEL4b ~]# useradd -c "Anonymous Samba Access" -p secret -s /bin/false Samba_nobod
[root@RHEL4b samba]# chown Samba_nobody.Samba_nobody /srv/samba/readonly/
[root@RHEL4b samba]# chmod 777 /srv/samba/readonly/
[root@RHEL4b samba]# ls -l /srv/samba/
total 4
drwxrwxrwx  2 Samba_nobody Samba_nobody 4096 Jun 22 11:09 readonly
[root@RHEL4b samba]# cd /srv/samba/readonly/
[root@RHEL4b readonly]# chown Samba_nobody.Samba_nobody *
[root@RHEL4b readonly]# ll
total 8
-rw-r--r--  1 Samba_nobody Samba_nobody 17 Jun 22 11:13 summer.txt
-rw-r--r--  1 Samba_nobody Samba_nobody 18 Jun 22 11:13 winter.txt
[root@RHEL4b samba]#
```

It is time to create the smb.conf file (feel free to test it with testparm). We put our file server in the default workgroup, give it a descriptive server string, and set the security to share level (more on this later). The share is called pubread, and access to the share is enforced by Samba (remember we gave 777 to the directory).

```
[root@RHEL4b samba]# cat smb.conf
[global]
workgroup = WORKGROUP
server string = Public Anonymous File Server
security = share
```

```
[pubread]
path = /srv/samba/readonly
comment = files to read
read only = Yes
guest ok = Yes
[root@RHEL4b samba]#
```

After testing with testparm, restart the samba server and verify the existence of the share with smbclient.

```
[root@RHEL4b readonly]# service smb restart
Shutting down SMB services:                                [  OK  ]
Shutting down NMB services:                                [  OK  ]
Starting SMB services:                                     [  OK  ]
Starting NMB services:                                     [  OK  ]
[root@RHEL4b readonly]# smbclient -L 127.0.0.1
Password:
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]

Sharename       Type      Comment
---------       ----      -------
pubread         Disk      files to read
IPC$            IPC       IPC Service (Public Anonymous File Server)
ADMIN$          IPC       IPC Service (Public Anonymous File Server)
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]

Server               Comment
---------            -------
RHEL4B               Public Test Server
WINXP

Workgroup            Master
---------            -------
WORKGROUP            WINXP
[root@RHEL4b readonly]#
```

The final test is to go to a Microsoft Windows computer and read a file on the Samba server. First we use the **net use** command to mount the pubread share on the driveletter k.

```
C:\Documents and Settings\paul>net use k: \\rhel4b\pubread
The command completed successfully.
```

Then we test looking at the contents of the share, and reading the files.

```
C:\Documents and Settings\paul>k:

K:\>dir
Volume in drive K is pubread
Volume Serial Number is 0D56-11F2

Directory of K:\

06/22/2007  11:13 AM    <DIR>          .
06/22/2007  11:09 AM    <DIR>          ..
06/22/2007  11:13 AM                18 winter.txt
```

```
06/22/2007  11:13 AM                17 summer.txt
2 File(s)             35 bytes
2 Dir(s)   2,763,522,048 bytes free

K:\>type winter.txt
It is cold today.

K:\>
```

Just to be on the safe side, let us try writing.

```
K:\>echo very cold > winter.txt
Access is denied.

K:\>
```

5.2. Practice

1. Create a directory in a good location (FHS) to share files for everyone to read.

2. Make sure the directory is owned properly, put a textfile in it, then share it with Samba.

3. Verify from your own and from another computer (smbclient, net use, ...) that the share is accessible for reading.

4. Make a backup copy of your smb.conf, name it smb.conf.ReadOnlyFileServer.

5.3. Writable File Server

In this second example, we will create a share where everyone can create files and write to files. Similar to before, we start by creating a directory, and setting ownership to our Samba_nobody user account.

```
[root@RHEL4b samba]# mkdir /srv/samba/writable
[root@RHEL4b samba]# chown Samba_nobody.Samba_nobody /srv/samba/writable/
[root@RHEL4b samba]# chmod 777 /srv/samba/writable/
```

Then we simply add a share to our file server by editing smb.conf. Below the check with testparm.

```
[root@RHEL4b samba]# testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[pubread]"
Processing section "[pubwrite]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions

# Global parameters
[global]
```

```
server string = Public Anonymous File Server
security = SHARE

[pubread]
comment = files to read
path = /srv/samba/readonly
guest ok = Yes

[pubwrite]
comment = files to read and write
path = /srv/samba/writable
read only = No
guest ok = Yes
```

Restart Samba, then onto the Windows XP machine and test our writing skills.

```
C:\Documents and Settings\paul>net use w: \\rhel4b\pubwrite
The command completed successfully.

C:\Documents and Settings\paul>w:

W:\>echo This is a write test > hello.txt

W:\>dir
Volume in drive W is pubwrite
Volume Serial Number is 0D56-272A

Directory of W:\

06/22/2007  12:29 PM    <DIR>          .
06/22/2007  12:26 PM    <DIR>          ..
06/22/2007  12:31 PM                23 hello.txt
1 File(s)             23 bytes
2 Dir(s)   2,763,522,048 bytes free

W:\>type hello.txt
type hello.txt
This is a write test

W:\>
```

There is one little issue though; the linux owner of the files created through this writable share is the linux guest account (usually named nobody).

```
[root@RHEL4b samba]# ls -l /srv/samba/writable/
total 4
-rwxr--r--  1 nobody nobody 23 Jun 22 12:31 hello.txt
-rwxr--r--  1 nobody nobody  0 Jun 22 12:33 test.txt
[root@RHEL4b samba]#
```

So this is not the cleanest solution. We will improve this in the next topic.

5.4. Forcing a User Owner

The Samba_nobody user account that we created in the previous examples is actually not used by Samba. It just owns the files and directories that we created for our shares.

The goal of this section is to force ownership of files created through the Samba share to belong to our Samba_nobody user. Remember, our server is still accessible to everyone, nobody needs to know this user account or password. We just want a clean linux server.

To accomplish this, we first have to tell Samba about this user. We can do this by adding the account to **smbpasswd**.

```
[root@RHEL4b samba]# smbpasswd -a Samba_nobody
New SMB password:
Retype new SMB password:
Added user Samba_nobody.
[root@RHEL4b samba]#
```

To find out where Samba keeps this information, use **smbd -b**. The PRIVATE_DIR variable will show you where the smbpasswd database is located.

```
[root@RHEL4b samba]# smbd -b | grep -i private
PRIVATE_DIR: /etc/samba
```

You can use a simple cat to see the contents of the smbpasswd database. The nobody user does not have a password, the Samba_nobody user does have one (it is secret).

```
[root@RHEL4b samba]# cat /etc/samba/smbpasswd
nobody:99:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:XXXXXXXXXXXXXXXXXXXXXXXXXXXXX...
Samba_nobody:502:552902031BEDE9EFAAD3B435B51404EE:878D8014606CDA29677A44...
[root@RHEL4b samba]#
```

Now that Samba knows about this user, we can adjust our writable share to force the ownership of files created through it. For this we use the **force user** and **force group** options. Now we can be sure that all files in the Samba writable share are owned by the same Samba_nobody user.

```
[root@RHEL4b samba]# testparm -s smb.conf 2>/dev/null | tail -7
[pubwrite]
comment = files to read and write
path = /srv/samba/writable
force user = Samba_nobody
force group = Samba_nobody
read only = No
guest ok = Yes
[root@RHEL4b samba]#
```

5.5. More about smbclient

Instead of going to the Microsoft machines, we can do the same tests from within linux with **smbclient**. This first screenshot shows how to verify that Samba is running on your localhost, how to list all the Samba shares, who is the Master Browser of the workgroup and some more information.

```
[paul@RHEL4b ~]$ smbclient -NL localhost
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]

Sharename       Type      Comment
---------       ----      -------
pubread         Disk      files to read
pubwrite        Disk      files to read and write
authwrite       Disk      authenticated users only
IPC$            IPC       IPC Service (Public Anonymous File Server)
ADMIN$          IPC       IPC Service (Public Anonymous File Server)
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]

Server               Comment
---------            -------
RHEL4B               Public Anonymous File Server
WINXP

Workgroup            Master
---------            -------
WORKGROUP            WINXP
[paul@RHEL4b ~]$
```

It can also be used to test user access to a Samba share. First an example of how to test anonymous access to our pubread share. If the connection is established, then we get an smb prompt. You can use exit or q to return to bash.

```
[paul@RHEL4b ~]$ smbclient //rhel4b/pubread -U%
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]
smb: \> dir
.                                    D        0  Fri Jun 22 11:13:15 2007
..                                   D        0  Fri Jun 22 13:03:54 2007
winter.txt                                   18  Fri Jun 22 11:13:11 2007
summer.txt                                   17  Fri Jun 22 11:13:15 2007

45734 blocks of size 262144. 10541 blocks available
smb: \> exit
[paul@RHEL4b ~]$
```

5.6. NetBIOS name resolving

If your clients are spread across multiple subnets, then it is likely there is a WINS (Microsoft Windows Internet Naming Service) or NBNS (NetBIOS Name Server) available to resolve NetBIOS names. You should then point Samba to the wins server with the **wins server** parameter.

```
wins server = 10.0.0.42
```

You can set the resolving order that Samba should use with the **name resolve order** parameter.

```
name resolve order = wins lmhosts host bcast
```

5.7. Practice

1. Create a directory and share it with Samba.

2. Make sure everyone can read and write files, test writing with smbclient and from a Microsoft computer.

3. Verify the ownership of files created by various users.

4. Use the "force user" and "force group" directives to force ownership of files created in this shared directory.

5. Test that Samba properly registers in a WINS server.

6. Test the working of force user with smbclient and/or net use and/or the MS Windows Explorer.

Chapter 6. Samba Servers with authentication and restrictions

6.1. Authenticated User Access

The goal of this example is to set up a file share accessible to a number of different users. The users will need to authenticate with their password before access to this share is granted. We will first create three randomly named users, each with their own password. First we add these users to linux.

```
[root@RHEL4b samba]# useradd -c "Serena Williams" -p SerenaW Serena
[root@RHEL4b samba]# useradd -c "Kim Clijsters" -p KimC Kim
[root@RHEL4b samba]# useradd -c "Martina Hingis" -p MartinaH Martina
```

Then we add them to the smbpasswd file, with the same password.

```
[root@RHEL4b samba]# smbpasswd -a Serena
New SMB password:
Retype new SMB password:
Added user Serena.
[root@RHEL4b samba]# smbpasswd -a Kim
New SMB password:
Retype new SMB password:
Added user Kim.
[root@RHEL4b samba]# smbpasswd -a Martina
New SMB password:
Retype new SMB password:
Added user Martina.
```

We add the following section to our smb.conf (and create the directory /srv/samba/authwrite).

```
[authwrite]
path = /srv/samba/authwrite
comment = authenticated users only
read only = No
guest ok = No
```

After restarting Samba, we test with different users from within Microsoft computers. First Kim from Windows XP.

```
C:\>net use m: \\rhel4b\authwrite /user:Kim KimC
The command completed successfully.

C:\>m:

M:\>echo greetings from Kim > greetings.txt
```

The next screenshot is Martina on a Windows 2000 computer, she succeeds in writing her files, but fails to overwrite the file from Kim.

```
C:\>net use k: \\rhel4b\authwrite /user:Martina MartinaH
The command completed successfully.

C:\>k:

K:\>echo greetings from martina > Martina.txt

K:\>echo test overwrite > greetings.txt
Access is denied.
```

You can also test connecting with authentication with smbclient, first we a wrong password, then with the correct one.

```
[paul@RHEL4b ~]$ smbclient //rhel4b/authwrite -UMartina%wrongpass
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]
tree connect failed: NT_STATUS_WRONG_PASSWORD
[paul@RHEL4b ~]$ smbclient //rhel4b/authwrite -UMartina%MartinaH
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]
smb: \> more Martina.txt
getting file \Martina.txt of size 25 as /tmp/smbmore.Uv6c86 (24.4 kb/s) (average 24.4 kb
greetings from martina
smb: \> q
[paul@RHEL4b ~]$
```

Congratulations, you now have a simple standalone Samba file server with authenticated access. And the files in the shares belong to their proper owners.

```
[root@RHEL4b samba]# ls -l /srv/samba/authwrite/
total 8
-rwxr--r--  1 Kim     Kim     17 Jun 22 13:05 greetings.txt
-rwxr--r--  1 Martina Martina 25 Jun 22 13:08 Martina.txt
```

6.2. Frequently used share settings

6.2.1. valid users

To restrict users per share, you can use the **valid users** parameter. In the example below, only the users listed as valid will be able to access the tennis share.

```
[tennis]
 path = /srv/samba/tennis
 comment = authenticated and valid users only
 read only = No
 guest ok = No
 valid users = serena, kim, venus, justine
```

6.2.2. invalid users

If you are paranoia, you can also use **invalid users** to explicitely deny the listed users access. When a user is in both lists, the user has no access!

```
[tennis]
 path = /srv/samba/tennis
 read only = No
 guest ok = No
 valid users = kim, serena, venus, justine
 invalid users = venus
```

6.2.3. create mask and inherit permissions

Similar to umask (but not inverted), you can use the **create mask** and **directory mask** to set default permissions for newly created files and directories.

```
[tennis]
 path = /srv/samba/tennis
 read only = No
 guest ok = No
 create mask = 644
```

With **inherit permissions = Yes** you can force newly created files and directories to inherit permissions from their parent directory, overriding the create mask and directory mask settings.

6.2.4. hosts allow

The **hosts allow** or **allow hosts** parameter is one of the key advantages of Samba. It allows access control of shares on the ip-address level. To allow only specific hosts to access a share, list the hosts, seperated by comma's.

```
allow hosts = 192.168.1.5, 192.168.1.40
```

Allowing entire subnets is done by ending the range with a dot.

```
allow hosts = 192.168.1.
```

Subnet masks can be added in the classical way.

```
allow hosts = 10.0.0.0/255.0.0.0
```

You can also allow an entire subnet with exceptions.

```
hosts allow = 10. except 10.0.0.12
```

6.2.5. hosts deny

The **hosts deny** or **deny hosts** parameter is the logical counterpart of the previous. The syntax is the same as for hosts allow.

```
hosts deny = 192.168.1.55, 192.168.1.56
```

6.2.6. hide unreadable

Setting **hide unreadable** to yes will prevent users from seeing files that cannot be read by them.

```
hide unreadable = yes
```

6.2.7. read list

One more setting before we go on to the next topic. Even on a writable share, you can set a list of read only users with the **read list** parameter.

```
[authwrite2]
 path = /srv/samba/authwrite2
 comment = authenticated users only
 read only = No
 guest ok = No
 read list = Martina, Roberto
```

6.3. Practice

0. Make sure you have properly named backups of your smb.conf of the previous practices.

1. Create three users (on the Unix and on the Samba), remember their passwords!

2. Set up a shared directory that is only accessible to authenticated users.

3. Verify that files created by these users belong to them.

4. Limit access to the sales share to Sandra, Ann and Veronique. Make sure that Roberto cannot access the share.

5. Even though the share is writable, Ann should only have read access.

6. Set the create mask for files to read and write for everyone, test that it works.

7. Limit one shared directory to the 192.168.1.0/24 subnet, and another share to the two computers with ip-addresses 192.168.1.33 and 172.17.18.19.

8. Make sure users can only see files and directories that they can read. Test that it works!!

9. If time permits (or if you are waiting for other students to finish this practice), then combine the "read only" and "writable" statements to check which one has priority. Then combine them with "read list", "write list", "hosts allow" and "hosts deny". Then

combine them with file permissions on the linux filesystem (chmod,chown) and make a table of minimal mandatory settings for readonly/readwrite shared directories.

Chapter 7. Samba Domain Member Server

7.1. smb.conf

The **workgroup** option in the global section should match the netbios name of the Active Directory domain. Authentication will not be handled by Samba now, but by the Active Directory Domain Controllers, so we set the **security** option to domain. Since linux requires a user account for every user accessing its file system, we need to provide Samba with a range of uid's and gid's that it can use to create these user accounts. The first Active Directory user to connect will receive linux uid 20000. Below is our new global section in smb.conf.

```
[global]
 workgroup = PEGASUS
 server string = Pegasus Domain Member Server
 security = Domain
 idmap uid = 20000-22000
 idmap gid = 20000-22000
 winbindd use default domain = Yes
```

Nothing special is required for the share section in smb.conf. Remember, we do not manually create users in smbpasswd or on the linux (/etc/passwd). Only Active Directory users are allowed access.

```
[domaindata]
 path = /srv/samba/domaindata
 comment = Active Directory users only
 read only = No
```

7.2. Joining the Active Directory Domain

While the Samba server is stopped, you can use **net rpc join** to join the Active Directory Domain.

```
[root@RHEL4b samba]# net rpc join -UAdministrator%Stargate1
Joined domain PEGASUS.
[root@RHEL4b samba]#
```

Time to start Samba followed by **winbind**.

```
[root@RHEL4b samba]# service smb start
Starting SMB services:                                     [  OK  ]
Starting NMB services:                                     [  OK  ]
[root@RHEL4b samba]# service winbindd start
Starting winbindd services:                                [  OK  ]
[root@RHEL4b samba]#
```

7.3. nsswitch.conf

We need to update the **/etc/nsswitch.conf** file now, so user group and host names can be resolved against the winbindd daemon.

```
[root@RHEL4b samba]# vi /etc/nsswitch.conf
[root@RHEL4b samba]# grep winbindd /etc/nsswitch.conf
passwd:     files winbindd
group:      files winbindd
hosts:      files dns winbindd
[root@RHEL4b samba]#
```

7.4. winbind

The **winbind** daemon is talking with the Active Directory domain. With **wbinfo** you can provide winbindd with credentials to talk to Active Directory.

```
[root@RHEL4b samba]# wbinfo --set-auth-user=Administrator%Stargate1
```

We can also use **wbinfo -a** to verify authentication of a user against Active Directory. Assuming a user account Venus with password VenusW is just created on the Active Directory, we get the following screenshot.

```
[root@RHEL4b samba]# wbinfo -a Venus%VenusW
plaintext password authentication succeeded
challenge/response password authentication succeeded
[root@RHEL4b samba]#
```

We can use **getent** to verify that winbindd is working and actually adding the Active directory users to /etc/passwd. The screenshot below shows that Kim and Serena are normal linux users in /etc/passwd, and that the Active Directory user Venus received uid 20000 in /etc/passwd.

```
[root@RHEL4b samba]# getent passwd Kim
Kim:x:504:504:Kim Clijsters:/home/Kim:/bin/bash
[root@RHEL4b samba]# getent passwd Serena
Serena:x:503:503:Serena Williams:/home/Serena:/bin/bash
[root@RHEL4b samba]# getent passwd Venus
venus:*:20000:20000::/home/PEGASUS/venus:/bin/false
```

Not all Active Directory user accounts added to /etc/passwd by winbindd, only those that have been used.

```
[root@RHEL4b samba]# getent passwd Justine
[root@RHEL4b samba]# wbinfo -a Justine%JustineH
plaintext password authentication succeeded
challenge/response password authentication succeeded
[root@RHEL4b samba]# getent passwd Justine
justine:*:20001:20000::/home/PEGASUS/justine:/bin/false
[root@RHEL4b samba]#
```

All the Active Directory users can now easily connect to the Samba share. Files created by them, belong to them.

```
[root@RHEL4b samba]# ll /srv/samba/domaindata/
total 0
-rwxr--r--  1 justine 20000 0 Jun 22 19:54 created_by_justine_on_winxp.txt
-rwxr--r--  1 venus   20000 0 Jun 22 19:55 created_by_venus.txt
-rwxr--r--  1 maria   20000 0 Jun 22 19:57 Maria.txt
```

7.5. Practice

1. Verify that you have a working Active Directory (AD) domain.

2. Setup Samba as a member server in the domain.

3. Verify the creation of a computer account in AD for your Samba server.

4. Verify the automatic creation of AD users in /etc/passwd with wbinfo and getent.

5. Connect to Samba shares with AD users, and verify ownership of their files.

Chapter 8. Samba Domain Controller

8.1. About Domain Controllers

8.1.1. Samba 3

Samba 3 can act as a domain controller in its own domain. In a Windows NT4 domain, with one Windows NT4 PDC and zero or more BDC's, Samba 3 can only be a member server. The same is valid for Samba 3 in an Active Directory Domain with Windows 2000 and/or Windows 2003 DC's. In short, a Samba 3 domain controller can not share domain control with Windows domain controllers.

8.1.2. Samba 4

Samba 4 can be a domain Controller in an Active Directory domain, but as of this writing, Samba 4 is not released for production!

8.1.3. About password backends

The example below uses the **tdbsam** password backend. Another option would be to use LDAP. Larger domains will benefit from using LDAP instead of the not so scalable tdbsam. When you need more than one Domain Controller, then the Samba team advises to not use tdbsam.

8.2. smb.conf

Now is a good time to start adding comments in your smb.conf. First we'll take a look at the naming of our domain and server in the **[global]** section, and at the domain controlling parameters. The security must be set to user (which is the default). Our Samba server is the most stable system in the network, so it should win all browser elections (**os level** above 32) to become the **browser master**, and it should accept domain logons (**domain logons = Yes**).

```
[global]
# names
 workgroup = SPORTS
 netbios name = DCSPORTS
 server string = Sports Domain Controller
# domain control parameters
 security = user
 os level = 80
 preferred master = Yes
 domain master = Yes
 domain logons = Yes
```

Then we create some sections for file shares, to test our Samba server. Users can all access the general sports file share, but only group members can access their own sport share.

```
[sports]
comment = Information about all sports
path = /srv/samba/sports
valid users = @ntsports
read only = No

[tennis]
comment = Information about tennis
path = /srv/samba/tennis
valid users = @nttennis
read only = No

[football]
comment = Information about football
path = /srv/samba/football
valid users = @ntfootball
read only = No
```

Part of the Microsoft definition of a domain controller is that it should have a **netlogon share**. This is the relevant part of smb.conf to create this netlogon share on Samba.

```
[netlogon]
comment = Network Logon Service
path = /srv/samba/netlogon
admin users = root
guest ok = Yes
browseable = No
```

8.3. Users and Groups

To be able to use users and groups in Samba, we have to set up some users and groups on the Linux computer.

```
[root@RHEL4b samba]# groupadd ntadmins
[root@RHEL4b samba]# groupadd ntsports
[root@RHEL4b samba]# groupadd nttennis
[root@RHEL4b samba]# groupadd ntfootball
[root@RHEL4b samba]# useradd -m -G ntadmins -p Stargate1 Administrator
[root@RHEL4b samba]# useradd -m -G ntsports,nttennis -p stargate Venus
[root@RHEL4b samba]# useradd -m -G ntsports,nttennis -p stargate Serena
[root@RHEL4b samba]# useradd -m -G ntsports,nttennis -p stargate Kim
[root@RHEL4b samba]# useradd -m -G ntsports,ntfootball -p stargate Figo
[root@RHEL4b samba]# useradd -m -G ntsports,ntfootball -p stargate Pfaff
```

Next we must make these users known to Samba with the smbpasswd tool. When you add the first user to **tdbsam**, the file **/etc/samba/passdb.tdb** will be created.

```
[root@RHEL4b samba]# smbpasswd -a Administrator
New SMB password:
```

```
Retype new SMB password:
Unable to open/create TDB passwd
pdb_getsampwnam: Unable to open TDB passwd (/etc/samba/passdb.tdb)!
TDBSAM version too old (0), trying to convert it.
TDBSAM converted successfully.
Added user Administrator.
[root@RHEL4b samba]#
```

Adding the second user generates less output.

```
[root@RHEL4b samba]# smbpasswd -a root
New SMB password:
Retype new SMB password:
Added user root.
```

8.4. About Computer Accounts

Every NT computer (Windows NT, 2000, XP, Vista) can become a member of a domain. Joining the domain (by right-clicking on My Computer) means that a computer account will be created in the domain. This computer account also has a password (but you cannot know it) to prevent other computers with the same name from accidentally becoming member of the domain. The computer account created by Samba is visible in the **/etc/passwd** file on linux. Computer accounts appear as a normal user account, but end their name with a dollar sign. Below a screenshot of the winxp$ computer account, created by Samba 3.

```
[root@RHEL4b samba]# tail -5 /etc/passwd
Serena:x:508:512::/home/Serena:/bin/bash
Kim:x:509:513::/home/Kim:/bin/bash
Figo:x:510:514::/home/Figo:/bin/bash
Pfaff:x:511:515::/home/Pfaff:/bin/bash
winxp$:x:512:516::/home/nobody:/bin/false
```

To be able to create the account, you will need to provide credentials of an account with the permission to create accounts (by default only root can do this on Linux). And we will have to tell Samba how to to this, by adding an **add machine script** to the global section of smb.conf.

```
add machine script = /usr/sbin/useradd -s /bin/false -d /home/nobody %u
```

You can now join a Microsoft computer to the sports domain (with the root user). After reboot of the Microsoft computer, you will be able to logon with Administrator (password Stargate1), but you will get an error about your roaming profile. We will fix this in the next section.

8.5. Roaming Profiles

For your information, if you want to force local profiles instead of roaming profiles, then simply add the following two lines to the global section in smb.conf.

```
logon home =
logon path =
```

Microsoft computers store a lot of User Metadata and application data in a user profile. Making this profile available on the network will enable users to keep their Desktop and Application settings across computers. User profiles on the network are called **roaming profiles** or **roving profiles**. The Samba domain controller can manage these profiles. First we need to add the relevant section in smb.conf.

```
[Profiles]
 comment = User Profiles
 path = /srv/samba/profiles
 readonly = No
 profile acls = Yes
```

Besides the share section, we also need to set the location of the profiles share (this can be another Samba server) in the global section.

```
 logon path = \\%L\Profiles\%U
```

The **%L** variable is the name of this Samba server, the **%U** variable translates to the username. After adding a user to smbpasswd and letting the user log on and off, the profile of the user will look like this.

```
[root@RHEL4b samba]# ll /srv/samba/profiles/Venus/
total 568
drwxr-xr-x  4 Venus Venus   4096 Jul  5 10:03 Application Data
drwxr-xr-x  2 Venus Venus   4096 Jul  5 10:03 Cookies
drwxr-xr-x  3 Venus Venus   4096 Jul  5 10:03 Desktop
drwxr-xr-x  3 Venus Venus   4096 Jul  5 10:03 Favorites
drwxr-xr-x  4 Venus Venus   4096 Jul  5 10:03 My Documents
drwxr-xr-x  2 Venus Venus   4096 Jul  5 10:03 NetHood
-rwxr--r--  1 Venus Venus 524288 Jul  5  2007 NTUSER.DAT
-rwxr--r--  1 Venus Venus   1024 Jul  5  2007 NTUSER.DAT.LOG
-rw-r--r--  1 Venus Venus    268 Jul  5 10:03 ntuser.ini
drwxr-xr-x  2 Venus Venus   4096 Jul  5 10:03 PrintHood
drwxr-xr-x  2 Venus Venus   4096 Jul  5 10:03 Recent
drwxr-xr-x  2 Venus Venus   4096 Jul  5 10:03 SendTo
drwxr-xr-x  3 Venus Venus   4096 Jul  5 10:03 Start Menu
drwxr-xr-x  2 Venus Venus   4096 Jul  5 10:03 Templates
[root@RHEL4b samba]#
```

8.6. Groups in NTFS acls

We have users on Unix, we have groups on Unix that contain those users.

```
[root@RHEL4b samba]# grep nt /etc/group
...
ntadmins:x:506:Administrator
ntsports:x:507:Venus,Serena,Kim,Figo,Pfaff
nttennis:x:508:Venus,Serena,Kim
ntfootball:x:509:Figo,Pfaff
```

```
[root@RHEL4b samba]#
```

We already added Venus to the **tdbsam** with **smbpasswd**.

```
smbpasswd -a Venus
```

Does this mean that Venus can access the tennis and the sports shares ? Yes, all access works fine on the Samba server. But the nttennis group is not available on the windows machines. To make the groups available on windows (like in the ntfs security tab of files and folders), we have to map unix groups to windows groups. To do this, we use the **net groupmap** command.

```
[root@RHEL4b samba]# net groupmap add ntgroup="tennis" unixgroup=nttennis type=d
No rid or sid specified, choosing algorithmic mapping
Successully added group tennis to the mapping db
[root@RHEL4b samba]# net groupmap add ntgroup="football" unixgroup=ntfootball type=d
No rid or sid specified, choosing algorithmic mapping
Successully added group football to the mapping db
[root@RHEL4b samba]# net groupmap add ntgroup="sports" unixgroup=ntsports type=d
No rid or sid specified, choosing algorithmic mapping
Successully added group sports to the mapping db
[root@RHEL4b samba]#
```

Now you can use the Samba groups on all NTFS volumes on members of the domain.

8.7. logon scripts

Before testing a logon script, make sure it has the proper carriage returns that DOS files have.

```
[root@RHEL4b netlogon]# cat start.bat
net use Z: \\DCSPORTS0\SPORTS
[root@RHEL4b netlogon]# unix2dos start.bat
unix2dos: converting file start.bat to DOS format ...
[root@RHEL4b netlogon]#
```

Then copy the scripts to the netlogon share, and add the following parameter to smb.conf.

```
logon script = start.bat
```

8.8. Practice

1. Setup Samba as a domain controller.

2. Create the shares salesdata, salespresentations and meetings. Salesdata must be accessible to all sales people and to all managers. SalesPresentations is only for all sales people. Meetings is only accessible to all managers. Use groups to accomplish this.

3. Join a Microsoft computer to your domain. Verify the creation of a computer account in /etc/passwd.

4. Setup and verify the proper working of roaming profiles.

5. Find information about home directories for users, set them up and verify that users receive their home directory mapped under the H:-drive in MS Windows Explorer.

6. Use a couple of samba domain groups with members to set acls on ntfs. Verify that it works!

7. Knowing that the %m variable contains the computername, create a seperate log file for every computer(account).

8. Knowing that %s contains the client operating system, include a smb.%s.conf file that contains a share. (The share will only be visible to clients with that OS).

9. If time permits (or if you are waiting for other students to finish this practice), then combine "valid users" and "invalid users" with groups and usernames with "hosts allow" and "hosts deny" and make a table of which get priority over which.

Chapter 9. Samba Print Servers

9.1. Simple CUPS Print Server

Let us start by setting up a Samba print server that serves two printers which are set up with the CUPS web interface (http://localhost:631). We make these printers available to everyone for printing. We set up the CUPS printers without a driver (raw printing device). The **lpstat** tool will see the printers like this.

```
[root@RHEL4b samba]# lpstat -t
scheduler is running
system default destination: HPColor
device for HPBlack: socket://192.168.1.244:9100
device for HPColor: parallel:/dev/lp0
HPBlack accepting requests since Jan 01 00:00
HPColor accepting requests since Jan 01 00:00
printer HPBlack is idle.  enabled since Jan 01 00:00
printer HPColor is idle.  enabled since Jan 01 00:00
```

The windows clients need to install the correct printer driver themselves, so the spooler just sends the jobs to the print device (without any kind of processing or interpreting of the print jobs). Our smb.conf looks like this.

```
[global]
 server string = Public Anonymous Print Server
 security = share
 disable spoolss = No
 printing = cups

[printers]
 path = /var/spool/samba
 read only = Yes
 printable = Yes
 use client driver = Yes
```

Let's do a quick check with smbclient.

```
[root@RHEL4b samba]# smbclient -NL 127.0.0.1
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10-1.4E.9]

Sharename       Type      Comment
---------       ----      -------
IPC$            IPC       IPC Service (Public Anonymous Print Server)
ADMIN$          IPC       IPC Service (Public Anonymous Print Server)
HPBlack         Printer   Local Raw Printer
HPColor         Printer   Local Raw Printer
...
```

That looks ok. Now you can add the printer to windows computers in the workgroup, just browse to your Samba server in the add printer wizard. Or you can connect with the **net use** command as shown below.

```
C:\shov>net use lpt1: \\rhel4b\HPColor
The command completed successfully.

C:\shov>net use
New connections will be remembered.


Status       Local     Remote                   Network

-------------------------------------------------------------------------------
OK           LPT1      \\rhel4b\HPColor         Microsoft Windows Network
The command completed successfully.

C:\shov>print shovel.bat
C:\shov\shovel.bat is currently being printed
```

After printing a test page (by rightclicking on the printer icon in windows and then clicking on the print test page button of the properties dialog box) and issuing the print command from within Firefox, the print queue looks like this.

```
[root@RHEL4b samba]# lpq -a
Rank    Owner   Job     File(s)                         Total Size
active  nobody  4       smbprn.00000001 Test Page       112640 bytes
1st     nobody  5       smbprn.00000002 Mozilla Firefox 120832 bytes
```

For troubleshooting, it can be useful to stop (pause) the printer. This way the jobs stay in the queue.

```
[root@RHEL4b samba]# lpstat -t
scheduler is running
system default destination: HPColor
device for HPBlack: socket://192.168.1.244:9100
device for HPColor: parallel:/dev/lp0
HPBlack accepting requests since Jan 01 00:00
HPColor accepting requests since Jan 01 00:00
printer HPBlack disabled since Jan 01 00:00 -
Paused
printer HPColor is idle.  enabled since Jan 01 00:00
HPBlack-4           nobody       112640   Sat 07 Jul 2007 07:59:33 AM CEST
HPBlack-5           nobody       120832   Sat 07 Jul 2007 08:00:04 AM CEST
```

9.2. Simple BSD Print Server

The default BSD style print commands (also refered to as LPD/LPR) are defined in rfc 1179. The smb.conf file is similar to the one for CUPS printing, except that CUPS is the default. The file now looks like this.

```
[global]
 server string = Public Anonymous Print Server
 printing = bsd
 load printers = yes

[printers]
 path = /var/spool/samba
 writable = no
 printable = Yes
 public = yes
```

Testparm however gives us some more information on values used for the print commands.

```
[root@RHEL4b samba]# testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[printers]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions

# Global parameters
[global]
 server string = Public Anonymous Print Server
 printing = bsd
 print command = lpr -r -P'%p' %s
 lpq command = lpq -P'%p'
 lprm command = lprm -P'%p' %j

[printers]
 path = /var/spool/samba
 guest ok = Yes
 printable = Yes
 browseable = No
[root@RHEL4b samba]#
```

9.3. Simple Unix SysV Print Server

SystemV style printing uses the lp command in this form.

```
lp -dprinter -s file
```

Since by default this command does not remove the file, we have to add this removal to smb.conf. So here is a simple smb.conf to share Unix System V type printers with Samba.

```
[global]
 server string = Public Anonymous Print Server
 printing = sysv
 load printers = yes

[printers]
 path = /var/spool/samba
 writable = no
 printable = Yes
 public = yes
 print command = lp -d%p -s %s ; rm %s
```

9.4. Samba Prining tips

The **printable = Yes** line must always be present in Samba printer shares, even in the **[printers]** section. It is also important to have a naming convention that prevents printers from having the same name as users. The **[homes]** section automatically creates a share for each user with that username, so it cannot be also a printer share.

To troubleshoot the print command, you can da a little trick in smb.conf. Instead of the actual print command, construct the printers section in smb.conf like this.

```
[printers]
 path = /var/spool/samba
 writable = no
 printable = Yes
 public = yes
 print command = echo "lpr -r -P'%p' %s" >> /tmp/bsdprint.log
```

Nothing will be printed, but you can test the print command that is generated by Samba. In this case, the log file looks like this.

```
[root@RHEL4b samba]# cat /tmp/bsdprint.log
lpr -r -P'HP400' smbprn.00000012.ARQtkM
lpr -r -P'HP400' smbprn.00000013.YbFkuN
lpr -r -P'HP400' smbprn.00000017.NeDuGj
[root@RHEL4b samba]#
```

Here is a list variables that are used by Samba for printing.

```
%s filename with path (of the file to be printed)
%f filename without path
%p name of the destination unix printer
%j print job number
```

9.5. Practice

1. Create two printers (with lpadmin or with the cups web interface) and pause(stop) them.

2. Serve these printers with Samba. Connect with a Microsoft computer and test printing.

3. Make sure only Isabelle and Caroline can access one of the printers.

4. Make sure they have to be on the 10.5.0.0/16 subnet to access the printer.

5. If time permits... There are some issues with a BSD printer. Your manager asks you to log the lpr command syntax, its stdout and its stderr to three different files.

Chapter 10. Introduction to Apache

10.1. About Apache

According to NetCraft (http://news.netcraft.com/archives/web_server_survey.html) about seventy percent of all web servers are running on Apache. Some people say that the name is derived from **a patchy** web server, because of all the patches people wrote for the NCSA httpd server.

10.2. Is Apache installed ?

To verify whether Apache is installed, use the proper tools (rpm, dpkg, ...) and grep for apache or httpd.

This Red Hat Enterprise 4 Server has apache installed.

```
[paul@rhel4 ~]$ rpm -qa | grep -i httpd
httpd-2.0.52-25.ent
httpd-manual-2.0.52-25.ent
system-config-httpd-1.3.1-1
httpd-devel-2.0.52-25.ent
httpd-suexec-2.0.52-25.ent
```

This Ubuntu also has apache installed.

```
paul@laika:~$ dpkg -l | grep apache
ii  apache2                 2.2.3-3.2build1      Next generation, scalable, ...
ii  apache2-mpm-prefork     2.2.3-3.2build1      Traditional model for Apach...
ii  apache2-utils           2.2.3-3.2build1      utility programs for webser...
ii  apache2.2-common        2.2.3-3.2build1      Next generation, scalable, ...
ii  libapache2-mod-php5     5.2.1-0ubuntu1.2     server-side, HTML-embedded ...
```

10.3. Is Apache running ?

This is how apache looks when it is installed on Red Hat Enterprise Linux 4, running named as **httpd**.

```
[root@RHELv4u3 ~]# /etc/init.d/httpd status
httpd is stopped
[root@RHELv4u3 ~]# service httpd start
Starting httpd:                                           [  OK  ]
[root@RHELv4u3 ~]# ps -C httpd
PID TTY          TIME CMD
4573 ?       00:00:00 httpd
4576 ?       00:00:00 httpd
4577 ?       00:00:00 httpd
4578 ?       00:00:00 httpd
4579 ?       00:00:00 httpd
4580 ?       00:00:00 httpd
```

```
4581 ?        00:00:00 httpd
4582 ?        00:00:00 httpd
4583 ?        00:00:00 httpd
[root@RHELv4u3 ~]#
```

And here is Apache running on Ubuntu, named as **apache2**.

```
root@laika:~# ps -C apache2
PID TTY          TIME CMD
6170 ?        00:00:00 apache2
6248 ?        00:00:01 apache2
6249 ?        00:00:01 apache2
6250 ?        00:00:00 apache2
6251 ?        00:00:01 apache2
6252 ?        00:00:01 apache2
7520 ?        00:00:01 apache2
8943 ?        00:00:01 apache2
root@laika:~# /etc/init.d/apache2 status
* Usage: /etc/init.d/apache2 {start|stop|restart|reload|force-reload}
root@laika:~#
```

To verify that apache is running, open a web browser on the web server, and browse to http://localhost. An Apache test page should be shown. The http://localhosts/manual url will give you an extensive Apache manual. The second test is to connect to your Apache from another computer.

10.4. Apache configuration

Configuring Apache changed a bit the past couple of years. But it still takes place in **/etc/httpd** or **/etc/apache**.

```
[root@RHELv4u3 ~]# cd /etc/httpd/
[root@RHELv4u3 httpd]# ll
total 32
lrwxrwxrwx  1 root root   25 Jan 24 09:28 build -> ../../usr/lib/httpd/build
drwxr-xr-x  7 root root 4096 Jan 24 08:48 conf
drwxr-xr-x  2 root root 4096 Jan 24 09:29 conf.d
lrwxrwxrwx  1 root root   19 Jan 24 08:48 logs -> ../../var/log/httpd
lrwxrwxrwx  1 root root   27 Jan 24 08:48 modules -> ../../usr/lib/httpd/modules
lrwxrwxrwx  1 root root   13 Jan 24 08:48 run -> ../../var/run
[root@RHELv4u3 httpd]#
```

The main configuration file for the Apache server on RHEL is **/etc/httpd/conf/httpd.conf**, on Ubuntu it is **/etc/apache2/apache2.conf**. The file explains itself, and contains examples for how to set up virtual hosts or configure access.

10.5. Virtual hosts

Virtual hosts can be defined by ip-address, by port or by name (host record). (The new way of defining virtual hosts is through seperate config files in the conf.d directory.) Below is a very simple virtual host definition.

```
[root@rhel4 conf]# tail /etc/httpd/conf/httpd.conf
#
# This is a small test website
#
<VirtualHost testsite.local:80>
ServerAdmin webmaster@testsite.local
DocumentRoot /var/www/html/testsite/
ServerName testsite.local
ErrorLog logs/testsite.local-error_log
CustomLog logs/testsite.local-access_log common
</VirtualHost>
[root@rhel4 conf]#
```

Should you put this little index.html file in the directory mentioned in the above screenshot, then you can access this humble website.

```
[root@rhel4 conf]# cat /var/www/html/testsite/index.html
<html>
 <head><title>Test Site</title></head>
 <body>
  <p>This is the test site.</p>
 </body>
</html>
```

Below is a sample virtual host configuration. This virtual hosts overrules the default Apache **ErrorDocument** directive.

```
<VirtualHost 83.217.76.245:80>
ServerName cobbaut.be
ServerAlias www.cobbaut.be
DocumentRoot /home/paul/public_html
ErrorLog /home/paul/logs/error_log
CustomLog /home/paul/logs/access_log common
ScriptAlias /cgi-bin/ /home/paul/cgi-bin/
<Directory /home/paul/public_html>
 Options Indexes IncludesNOEXEC FollowSymLinks
 allow from all
</Directory>
ErrorDocument 404 http://www.cobbaut.be/cobbaut.php
</VirtualHost>
```

10.6. Aliases and redirects

Apache supports aliases for directories, like this example shows.

```
Alias /paul/ "/home/paul/public_html/"
```

Similarly, content can be redirected to another website or web server.

```
Redirect permanent /foo http://www.foo.com/bar
```

10.7. Securing directories with htpasswd and .htaccess

You can secure files and directories in your website with a userid/password. First, enter your website, and use the **htpasswd** command to create a **.htpasswd file** that contains a userid and an (encrypted) password.

```
[root@rhel4 testsite]# htpasswd -c .htpasswd pol
New password:
Re-type new password:
Adding password for user pol
[root@rhel4 testsite]# cat .htpasswd
pol:x5vZlyw1V6KXE
[root@rhel4 testsite]#
```

You can add users to this file, just don't use the -c switch again.

```
[root@rhel4 testsite]# htpasswd .htpasswd kim
New password:
Re-type new password:
Adding password for user kim
[root@rhel4 testsite]# cat .htpasswd
pol:x5vZlyw1V6KXE
kim:6/RbvugwsgOI6
[root@rhel4 testsite]#
```

You have now defined two users. Next create a subsdirectory that you want to protect with these two accounts. And put the following .htaccess file in that subdirectory.

```
[root@rhel4 kimonly]# pwd
/var/www/html/testsite/kimonly
[root@rhel4 kimonly]# cat .htaccess
AuthUserFile /var/www/html/testsite/.htpasswd
AuthGroupFile /dev/null
AuthName "test access title"
AuthType Basic

<Limit GET POST>
require valid-user
</Limit>
[root@rhel4 kimonly]#
```

Finally, don't forget to verify that AllowOverride is set to All in the general Apache configuration file.

```
# AllowOverride controls what directives may be placed in .htaccess files.
# It can be "All", "None", or any combination of the keywords:
#   Options FileInfo AuthConfig Limit
#
AllowOverride All
```

From now on, when a user accesses a file in that subdirectory, that user will have to provide a userid/password combo that is defined in your .htpasswd.

10.8. more on .htaccess

You can do much more with **.htaccess**. One example is to use .htaccess to prevent people from certain domains to access your website. Like in this case, where a number of referer spammers are blocked from the website.

```
paul@lounge:~/cobbaut.be$ cat .htaccess
# Options +FollowSymlinks
RewriteEngine On
RewriteCond %{HTTP_REFERER} ^http://(www\.)?buy-adipex.fw.nu.*$ [OR]
RewriteCond %{HTTP_REFERER} ^http://(www\.)?buy-levitra.asso.ws.*$ [NC,OR]
RewriteCond %{HTTP_REFERER} ^http://(www\.)?buy-tramadol.fw.nu.*$ [NC,OR]
RewriteCond %{HTTP_REFERER} ^http://(www\.)?buy-viagra.lookin.at.*$ [NC,OR]
...
RewriteCond %{HTTP_REFERER} ^http://(www\.)?www.healthinsurancehelp.net.*$ [NC]
RewriteRule .* - [F,L]
paul@lounge:~/cobbaut.be$
```

10.9. Traffic

Apache keeps a log of all visitors. The **webalizer** is often used to parse this log into nice html statistics.

10.10. Practice

1. Verify that Apache is installed and running.

2. Browse to the Apache HTML manual from another computer.

3. Create a virtual hosts that listens to port 8247.

4. Create a virtual hosts that listens on another ip-address.

5. Test from another computer that all virtual hosts work.

6. Protect a subdirectory of a website with .htpasswd and .htaccess.

Chapter 11. Introduction to squid

11.1. about proxy servers

11.1.1. usage

A **proxy server** is a server that caches the internet. Clients connect to the proxy server with a request for an internet server. The proxy server will connect to the internet server on behalf of the client. The proxy server will also cache the pages retrieved from the internet server. A proxy server may provide pages from his cache to a client, instead of connecting to the internet server to retrieve the (same) pages.

A proxy server has two main advantages. It improves web surfing speed when returning cached data to clients, and it reduces the required bandwidth (cost) to the internet.

Smaller organizations sometimes put the proxy server on the same physical computer that serves as a NAT to the internet. In larger organizations, the proxy server is one of many servers in the DMZ.

When web traffic passes via a proxy server, it is common practice to configure the proxy with extra settings for access control. Access control in a proxy server can mean user account access, but also website(url), ip-address or dns restrictions.

11.1.2. open proxy servers

You can find lists of open proxy servers on the internet that enable you to surf anonymously. This works when the proxy server connects on your behalf to a website, without logging your ip-address. But be careful, these (listed) open proxy servers could be created in order to eavesdrop upon their users.

11.1.3. squid

This chapter is an introduction to the **squid** proxy server (http://www.squid-cache.org). The version used is 2.5.

```
[root@RHEL4 ~]# rpm -qa | grep squid
squid-2.5.STABLE6-3.4E.12
[root@RHEL4 ~]#
```

11.2. squid proxy server

11.2.1. /etc/squid/squid.conf

Squid's main configuration file is **/etc/squid/squid.conf**. The file explains every parameter in great detail. It can be a good idea to start by creating a backup of this file.

```
[root@RHEL4 /etc/squid/]# cp squid.conf squid.conf.original
```

11.2.2. /var/spool/squid

The **squid** proxy server stores its cache by default in **/var/spool/squid**. This setting is configurable in /etc/squid/squid.conf.

```
[root@RHEL4 ~]# grep "^# cache_dir" /etc/squid/squid.conf
# cache_dir ufs /var/spool/squid 100 16 256
```

It is possible that in a default setup where squid has never run, that the /var/spool/squid directories do not exist.

```
[root@RHEL4 ~]# ls -al /var/spool/squid
ls: /var/spool/squid: No such file or directory
```

Running **squid -z** will create the necessary squid directories.

```
[root@RHEL4 ~]# squid -z
2008/09/22 14:07:47| Creating Swap Directories
[root@RHEL4 ~]# ls -al /var/spool/squid
total 80
drwxr-x---   18 squid squid 4096 Sep 22 14:07 .
drwxr-xr-x   26 root  root  4096 May 30  2007 ..
drwxr-xr-x  258 squid squid 4096 Sep 22 14:07 00
drwxr-xr-x  258 squid squid 4096 Sep 22 14:07 01
drwxr-xr-x  258 squid squid 4096 Sep 22 14:07 02
...
```

11.2.3. port 3128 or port 8080

By default the squid proxy server will bind to port 3128 to listen to incoming requests.

```
[root@RHEL4 ~]# grep "default port" /etc/squid/squid.conf
#       The default port number is 3128.
```

Many organizations use port 8080 instead.

```
[root@RHEL4 ~]# grep 8080 /etc/squid/squid.conf
http_port 8080
```

11.2.4. /var/log/squid

The standard log file location for squid is **/var/log/squid**.

```
[root@RHEL4 ~]# grep "/var/log" /etc/squid/squid.conf
# cache_access_log /var/log/squid/access.log
# cache_log /var/log/squid/cache.log
# cache_store_log /var/log/squid/store.log
```

11.2.5. access control

The default squid setup only allows localhost access. To enable access for a private network range, look for the "INSERT YOUR OWN RULE(S) HERE..." sentence in squid.conf and add two lines similar to the screenshot below.

```
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS

acl company_network src 192.168.1.0/24
http_access allow company_network
```

Restart the squid server, and now the local private network can use the proxy cache.

11.2.6. testing squid

First, make sure that the server running squid has access to the internet.

```
[root@RHEL4 ~]# wget -q http://linux-training.be/index.html
[root@RHEL4 ~]# ls -l index.html
-rw-r--r--  1 root root 2269 Sep 18 13:18 index.html
[root@RHEL4 ~]#
```

Then configure a browser on a client to use the proxy server. OR you could set the HTTP_PROXY (sometimes http_proxy) variable to point command line programs to the proxy.

```
[root@fedora ~]# export HTTP_PROXY=http://192.168.1.39:8080
[root@ubuntu ~]# export http_proxy=http://192.168.1.39:8080
```

Testing a client machine can then be done with wget (wget -q is used to simplify the screenshot).

```
[root@RHEL5 ~]# > /etc/resolv.conf
```

```
[root@RHEL5 ~]# wget -q http://www.linux-training.be/index.html
[root@RHEL5 ~]# ls -l index.html
-rw-r--r-- 1 root root 2269 Sep 18  2008 index.html
[root@RHEL5 ~]#
```

11.2.7. name resolution

You need name resolution working on the squid server, but you don't need name resolution on the clients.

```
[paul@RHEL5 ~]$ wget http://grep.be
--14:35:44--  http://grep.be
Resolving grep.be... failed: Temporary failure in name resolution.
[paul@RHEL5 ~]$ export http_proxy=http://192.168.1.39:8080
[paul@RHEL5 ~]$ wget http://grep.be
--14:35:49--  http://grep.be/
Connecting to 192.168.1.39:8080... connected.
Proxy request sent, awaiting response... 200 OK
Length: 5390 (5.3K) [text/html]
Saving to: `index.html.1'

100%[===============================>] 5,390       --.-K/s   in 0.1s

14:38:29 (54.8 KB/s) - `index.html' saved [5390/5390]

[paul@RHEL5 ~]$
```

Chapter 12. Introduction to bind

12.1. DNS History

Today, **DNS** or **Domain Name System** is a worldwide distributed hierarchical database. It's primary function is to resolve names to ip addresses, and to point to internet servers providing SMTP or LDAP services.

In the seventies, only a few hundred computers were connected to the internet. To resolve names, computers had a flat file that contained a table to resolve hostnames to ip-addresses. This local file was downloaded from hosts.txt on an ftp server in Stanford.

In 1984 **Paul Mockapetris** created DNS, a distributed treelike hierarchical database.

ICANN..............

12.2. DNS Structure

12.2.1. root

DNS is a tree structure. The top of the tree is called the **root**. There are thirteen root servers on the internet, they are named A to M. Journalist often refer to these servers as **the master servers of the internet**, because if these servers go down, then nobody can (use names to) connect to websites.

The root servers are not thirteen physical machines, in fact ... (expand later with mirror info...)

12.2.2. top level domains (TLD)

Below the root level are the **top level domains** or **TLD's**. Originally there were only seven defined:.com(mercial),.edu(cational),.gov(ernment),.int(ernational),.mil(ilitary),.net(work) and .org for non-commercial organizations. The .arpa domain was also used, it will be explained later.

Country TLD's were defined for individual countries, like .be for Belgium and .fr for France.

In the 21st century new TLD's were defined like .museum .info .biz and .aero.

12.2.3. domains

One level below the top level domains are the **domains**. Examples of domain names are google.com or linux-training.be. Domains can have subdomains (also called child domains).

12.2.4. fully qualified domain name

The **fully qualified domain name** or **FQDN** is the combination of the hostname of a machine appended with its domain name.

If for example a system is called **wolf** and it is in the domain **stargate.local**, then the FQDN of this system is **wolf.stargate.local**.

12.3. How DNS works

12.3.1. zone

A zone is a portion of the DNS tree. A DNS server that is controlling a zone, is said to be the **authoritative** DNS server for that zone. A zone is a collection of **resource records**. There are several types of resource records, for example A, PTR, NS, MX, SOA and CNAME.

12.3.1.1. A record

The **A record**, which is also called a **host record** contains the ipv4-address of a computer. When a DNS client queries a DNS server for an A record, then the DNS server will resolve the hostname in the query to an ip-address. An **AAAA record** is similar but contains an ipv6 address instead of ipv4.

12.3.1.2. PTR record

A **PTR record** is the reverse of an A record. It contains the name of a computer and can be used to resolve an ip-address to a hostname.

12.3.1.3. NS record

A **NS record** or **nameserver record** is a record that points to a DNS name server (in this zone). You can list all your name servers for your DNS zone in distinct NS records.

12.3.1.4. SOA record

The SOA record of a zone contains meta information about the zone itself. The contents of the SOA record is explained in detail in the section about zone transfers. There is exactly one SOA record for each zone.

12.3.1.5. CNAME record

A **CNAME record** maps a hostname to a hostname, creating effectively an alias for an existing hostname. The name of the mail server is often aliased to **mail** or **smtp**, and the name of a web server to **www**.

12.3.1.6. MX record

The **MX** points to an SMTP server. When you send an email to another domain, then your mail server will need the MX record of the target domain's mail server.

12.3.2. master and slave

There are several reasons to create more than one name server in a zone. One server might not be able to answer to all queries, or you might want some fault tolerance to mitigate the impact of hardware failure. When adding a **secondary DNS server** to a zone, then you will configure this server as a **slave server** to the **primary server**. The primary server then becomes the **master server** of the slave server.

Very often the primary DNS server is the master server of all slaves. Sometimes a slave server is master server for a second line slave server.

12.3.3. zone transfers

The slave server receives a copy of the zone database using a **zone transfer**. Zone transfers are requested by the slave servers at regular intervals. Those intervals are defined in the **SOA record**.

The SOA record contains a **refresh** value. If this is set to 30 minutes, then the slave server will request a copy of the zone file every 30 minutes. There is also a **retry** value. The retry value is used when the master server did not reply to the last zone transfer request. The value for **expiry time** says how long the slave server will answer to queries, without receiving a zone update.

Zone transfers only occur when the zone database was updated (meaning when one or more resource records were added, removed or changed on the master server). The slave server will compare the **serial number** of its own copy of the SOA record with the serial number of its master's SOA record. When both serial numbers are the same, then no update is needed (because no records were added, removed or deleted). When the slave has a lower serial number than its master, then a zone transfer is requested.

12.3.4. full or incremental zone transfers

When a zone tranfer occurs, this can be either a full zone transfer or an incremental zone transfer. The decision depends on the size of the transfer that is needed to completely update the zone on the slave server. An incremental zone transfer is prefered when the total size of changes is smaller than the size of the zone database. Full zone transfers use the **axfr** protocol, incremental zone transfer use the **ixfr** protocol.

12.3.5. DNS cache

DNS is a caching protocol. When a client queries its local DNS server, and the local DNS server is not authoritative for the query, then this server will go looking for an authoritative name server in the DNS tree. The local name server will first query a root server, then a TLD server and then a domain server. When the local name server resolves the query, then it will relay this information to the client that submitted the query, and it will also keep a copy of these queries in its cache. So when a(nother) client submits the same query to this name server, then it will retrieve this information form its cache.

For example, a client queries for the A record on www.linux-training.be to its local server. This is the first query ever received by this local server. The local server checks that it is not authoritative for the linux-training.be domain, nor for the .be TLD, and it is also not a root server. So the local server will use the root hints to send an **iterative** query to a root server. The root server will reply with a reference to the server that is authoritative for the .be domain (root DNS servers do not resolve fqdn's, and root servers do not respond to recursive queries). The local server will then sent an iterative query to the authoritative server for the .be TLD. This server will respond with a reference to the name server that is authoritative for the linux-training.be domain. The local server will then sent the query for www.linux-training.be to the authoritative server (or one of its slave servers) for the linux-training.be domain. When the local server receives the ip-address for www.linux-training.be, then it will provide this information to the client that submitted this query. Besides caching the A record for www.linux-training.be, the local server will also cache the NS and A record for the linux-training.be name server and the .be name server.

12.3.6. caching only server

A DNS server that is set up without its own zone, but that is connected to other name servers and caches the queries is called a **caching only name server**.

12.3.7. iterative or recursive query

A **recursive query** is a DNS query where the client that is submitting the query expects a complete answer. An **iterative query** is a DNS query where the client does not expect a complete answer. Iterative queries usually take place between name servers. The root name servers do not respond to recursive queries.

12.4. old stuff....work in progress

Forward lookup zones are most common, they contain host or A records to translate hostnames or Fully Qualified Domain Names (FQDN) to ip addresses. Reverse lookup zones contain PTR records, they translate ip addresses to hostnames or FQDN's.

The internet contains thirteen logical DNS servers for the top of the hierarchy. This top is called the root, and is represented with a dot. Below the root are the Top Level Domains (TLD's). There are common TLD's like .com, .net. .info. aero. .museum, .gov, .mil, .edu and others. And there are country TLD's, like .be for Belgium and .fr for France.

The internet root name servers will only answer iterative queries, most local DNS servers will answer to recursive queries.

12.5. bind

One of the most common name servers on Linux is the Berkeley Internet Name Domain (bind) server. Use rpm or dpkg to verify whether it is installed.

```
[root@RHEL4b etc]# rpm -qa | grep -i bind
ypbind-1.17.2-8
bind-chroot-9.2.4-16.EL4
bind-utils-9.2.4-16.EL4
bind-devel-9.2.4-16.EL4
bind-libs-9.2.4-16.EL4
bind-9.2.4-16.EL4
```

12.6. named

The software is called 'bind', the daemon runs as 'named' ! So look for the named daemon, the named manual pages and /etc/named.conf to work with bind.

```
[root@RHEL4b etc]# apropos named | grep -i domain
named                (8)  - Internet domain name server
```

12.7. Caching only Name Server

A caching only name server is a DNS server that is not authoritative for any zone. It forwards queries to other DNS servers and locally caches the results.

The default /etc/named.conf on RHEL is a caching only name server.

12.8. Our first zone

The way to set up zones in /etc/named.conf is to create a zone entry with a reference to another file located in /var/named.

Here is an example of such an entry in /etc/named.conf

```
zone "classdemo.local" IN {
 type master;
 file "classdemo.local.zone";
 allow-update { none; };
};
```

To create the zone file, the easy method is to copy an existing zone file (this is easier than writing from scratch).

```
[root@RHEL4b named]# cd /var/named/
[root@RHEL4b named]# pwd
/var/named
[root@RHEL4b named]# cp localhost.zone classdemo.local.zone
[root@RHEL4b named]#
```

Here is an example of a zone file.

```
[root@RHEL4b named]# cat classdemo.local.zone
$TTL    86400
$ORIGIN classdemo.local.
@       IN SOA  rhel4b.classdemo.local.   admin.classdemo.local. (
                         2007083100      ; serial
                         3H              ; refresh
                         900             ; retry
                         1W              ; expiry
                         1D )            ; minimum

                IN NS           rhel4b.classdemo.local.
                IN MX     10    mail.classdemo.local.
                IN A            192.168.1.191

rhel4b          IN      A       192.168.1.191
mail            IN      A       192.168.1.191
www             IN      A       192.168.1.191
ftp             IN      A       192.168.1.191
server2         IN      A       192.168.1.1
```

12.9. Starting the name server

When starting the name server, don't forget to look at the log file to verify that all your zones are properly configured.

```
[root@RHEL4b etc]# service named restart
Stopping named:                                            [  OK  ]
Starting named:                                            [  OK  ]
[root@RHEL4b etc]# service named status
number of zones: 9
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is OFF
server is up and running
[root@RHEL4b etc]#
```

12.10. Practice DNS

1. Set up a working DNS server with your own zone. Test that it works.

2. Set up a master and a slave server.

Index