How to protect your home/office network?

Using IPTables and Building a Firewall - Background, Motivation and Concepts

Adir Abraham

adir@vipe.technion.ac.il

Do you think that you are alone, connected from your computer to the Internet?

You are wrong. Many nodes (clients and servers), are watching you, getting answered and questioning your computer.

```
13:11:41.045992 132.69.237.98.netbios-ns > 132.69.255.255.netbios-ns:
>>> NBT UDP PACKET(137): QUERY; REQUEST; BROADCAST
13:11:41.056873 arp who-has 132.69.231.8 tell 132.69.229.97
13:11:41.058760 arp who-has 132.69.244.181 tell 132.69.228.93
13:11:41.081228 arp who-has 132.69.217.32 tell 132.69.237.62
13:11:41.086823 arp who-has 132.69.231.72 tell 132.69.238.215
13:11:41.099591 arp who-has 132.69.237.98 tell 132.69.255.195
13:11:41.121914 84440000.00:10:a4:15:0b:b2.453 > 0.ff:ff:ff:ff:ff:ff.453:ipx-rip-req 2219102368/65535.65535
13:11:41.124738 0:a:e6:52:4e:2a > Broadcast sap e0 ui/C
>>> Unknown IPX Data: (43 bytes)
[000] FF FF 00 22 00 00 00 00  00 00 FF FF FF FF FF FF  ...".... ........
[010] 04 52 84 44 00 00 00 0A  E6 52 4E 2A 40 00 00 01  .R.D.... .RN*@...
[020] 00 04 00 FF FF FF FF FF  FF FF FF                 ........ ...
 len=43
….

2004/05/03 13:48:25 80.179.187.43:2632 (80.179.187.43.forward.012.net.il) 80.179.165.105:135 DCE endpoint resolution
2004/05/03 13:47:23 217.132.67.28:3725 (CBL217-132-67-28.bb.netvision.net.il) 172.23.90.237:445 Microsoft-DS
2004/05/03 13:47:17 80.179.41.181:1895 (80.179.41.181.forward.012.net.il) 80.179.165.105:135 DCE endpoint resolution
2004/05/03 13:46:58 80.178.191.187:4009 (80.178.191.187.forward.012.net.il) 80.179.165.105:135 DCE endpoint resolution
2004/05/03 13:46:12 80.179.155.118:4805 (80.179.155.118.forward.012.net.il) 80.179.165.105:135 DCE endpoint resolution
2004/05/03 13:42:00 80.178.148.210:3673 (80.178.148.210.forward.012.net.il) 80.179.14.104:135 DCE endpoint resolution
2004/05/03 13:41:43 80.179.6.193:1401 (80.179.6.193.forward.012.net.il) 80.179.14.104:135 DCE endpoint resolution
2004/05/03 13:40:19 80.186.3.239:1706 (a80-186-3-239.elisa-laajakaista.fi) 80.179.14.104:2904 M2UA
2004/05/03 13:39:44 62.0.106.228:4916 (HFA62-0-106-228.bb.netvision.net.il) 172.23.90.237:445 Microsoft-DS
2004/05/03 13:36:02 217.132.253.29:2089 (CBL217-132-253-29.bb.netvision.net.il) 172.23.90.237:445 Microsoft-DS
2004/05/03 13:34:48 80.219.150.224:0 (80-219-150-224.dclient.hispeed.ch) 80.179.14.104:0 ICMP Ping
2004/05/03 13:33:54 172.23.24.244:1297 (SAM-CRYQ2SRAXIX) 172.23.90.237:445 Microsoft-DS
2004/05/03 13:32:57 212.235.75.165:3358 (DSL212-235-75-165.bb.netvision.net.il) 172.23.90.237:445 Microsoft-DS
2004/05/03 13:31:48 217.132.236.59:3887 (CBL217-132-236-59.bb.netvision.net.il) 172.23.90.237:445 Microsoft-DS
2004/05/03 13:30:32 217.132.159.57:4843 (CBL217-132-159-57.bb.netvision.net.il) 172.23.90.237:445 Microsoft-DS
…
```

What makes a network or specifically your computer vulnerable?

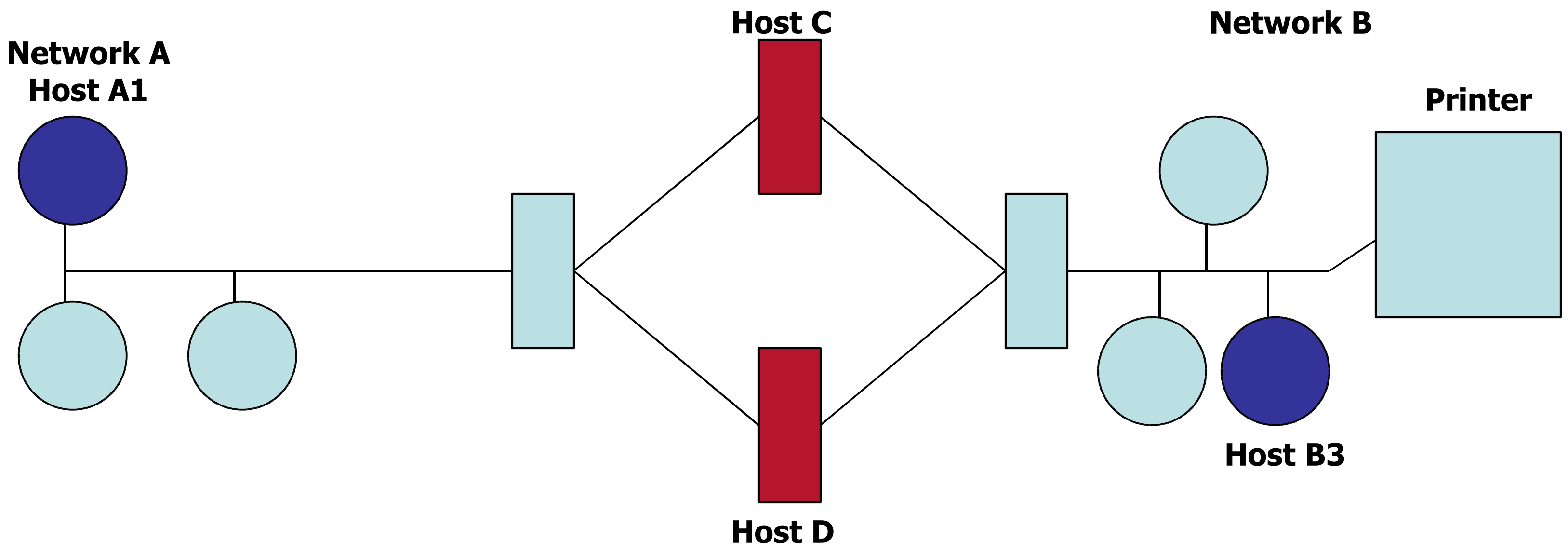
•**Anonymity**: An attacker can mount an attack from thousands of miles away and never come into direct contact with the system itself, its administrators, or users. The potential attacker is thus safe behind an electronic shield. The attack can be passed through many other hosts in an effort to disguise the attack's origin.

•**Many points to attack**: both targets and origins: An attack can come from any host to any host, so that a large network offers many points of vulnerability. A self-contained computer can also get attacked if an attacker scans a wide range of IPs, and those IPs (random ones) reply with the information that he wishes to get.

•**Sharing**: Because networks enable resource and workload sharing (BitTorrent, for example, see http://bitconjurer.org/BitTorrent/introduction.html), more users have the potential to access networked systems than on single computers. Perhaps worse, access is afforded to more systems, so that access controls for single systems may be inadequate in networks.

•**Complexity of system**: Reliable security (which is a separate topic by itself) is difficult, if not impossible, on large operating systems, especially those who are not designed specifically for security. A network usually combines two or more possibly dissimilar operating systems. Therefore, a network operating/control system is likely to be more complex than an operating system for a single computing system.

- **Unknown path**: There may be many paths from one host to another. Suppose that a user from host A1 wants to send a message to a user on host B3. That message might be routed through hosts C or D before arriving at host B3. Host C may provide acceptable security, but not D. Network users seldom have control over the routing of their messages



Thus, a network differs significantly from a stand-alone and local environment. Network characteristics significantly increase the security risk. Also, you can see The differences between a router (all the packets are either forwarded or get into the router itself), and a firewall (which lets packets to pass only if they match the rules of passing).

Port Scan

An easy way to gather network information is to use a port scan. A program that, for a particular IP address, reports which ports respond to messages and which of several known vulnerabilities seem to be present.

Port scanning tells an attacker three things: which standard ports or services are running and responding on the target system, what operating system is installed on the target system, and what applications and versions of applications are present. This information is readily available for the asking from a networked system. It can be obtained quietly, anonymously, without identification or authentication, drawing little or no attention to the scan at all.

Port scanning tools are readily available, and not just to the underground community. The nmap scanner by Fyodor at http://www.insecure.org/nmap is a useful tool that anyone can download. Given an address, nmap will report all open ports, the service they support, and the owner (user ID) of the daemon providing the service.

- **An example of nmap use:**

Suppose we want to want to find out which system services that respond to TCP are operating on machine YYY (IP: y1.y2.y3.y4), on ports 1-1023, to give us some details about the connection and to guess the operating system of that machine. This is the result:

```
[root@vipe root]# nmap -v -sT -O YYY -p 1-1023

Starting nmap V. 2.54BETA22 ( www.insecure.org/nmap/ )
Host YYY (y1.y2.y3.y4) appears to be up ... good.
Initiating Connect() Scan against YYY (y1.y2.y3.y4)
Adding TCP port 53 (state open).
Adding TCP port 22 (state open).
The Connect() Scan took 96 seconds to scan 1023 ports.
For OSScan assuming that port 22 is open and port 21 is closed and neither are firewalled
For OSScan assuming that port 22 is open and port 21 is closed and neither are firewalled
Insufficient responses for TCP sequencing (0), OS detection may be less accurate
WARNING:  OS didn't match until the try #2
Interesting ports on YYY (y1.y2.y3.y4):
(The 1018 ports scanned but not shown below are in state: filtered)
Port       State       Service
21/tcp     closed      ftp
22/tcp     open        ssh
53/tcp     open        domain
80/tcp     closed      http
443/tcp    closed      https

Remote OS guesses: Solaris 2.5, 2.5.1, Solaris 2.6 - 2.7, Solaris 2.6 - 7 X86, Solaris 2.6, Solaris 2.6 - 2.7
with tcp_strong_iss=0, Solaris 2.6 - 2.7 with tcp_strong_iss=2, Sun Solaris 8 early acces beta through
actual release

Nmap run completed -- 1 IP address (1 host up) scanned in 109 seconds
```

It seems that YYY is pretty secured. Only relevant services are open (SSH, DNS), and all the rest are close. We can also see the guess of the OS (the guess was wrong, by the way).

So, there are many intrusions out there. <u>How to protect against (most of) them?</u>

Firewalls – What exactly is a firewall?

A firewall is a device that filters all traffic between a protected, internal network and a less trustworthy, external network. Firewalls usually run on a proprietary or carefully minimized operating system, or on a dedicated hardware device.

The purpose of a firewall is to keep "bad" things outside of the protected environment. Firewalls can implement a security policy that is specifically designed to address what are the "bad" things that might happen.

For example, the policy might be to prevent any access from outside (while still allowing traffic to pass *from* the *inside, to* the *outside*). Alternatively, the policy might permit accesses only from certain places, certain users, or for certain activities.

Part of the challenge of protecting a network with a firewall is determining which security policy meets the needs of the installation. People in the firewall community are divided about a firewalls' default behavior.

- Default permit: "That which is not expressly forbidden, is permitted by default"
- Default deny: "That which is not expressly permitted, is forbidden by default"

Types of Firewalls

Firewalls have a wide range of capabilities. Types of firewalls include:

- Stateful inspection firewalls: A firewall which maintains state information from one packet to another in the input stream. It will be discussed in an advanced lecture.

- Application proxy: Programs which mediate the conversation between a client and a server, rather than merely (and blindly) forwarding packets. Because application proxies can examine much higher-level protocols than is advisable for kernel code, it's possible to use them to protect clients and servers against malice by ensuring that only valid protocol exchanges take place. For example, servers which you would like to protect with an application proxy: IRC and IMAP servers.

- Guard: works like a proxy firewall, but also perform on the user’s behalf in accordance with its available knowledge (for example – a password). Usually, Firewall Proxies can be easily upgraded to Guards.

- Personal firewall: This is an application program that runs on a workstation to block unwanted traffic. A personal firewall can complement the work of a conventional firewall by screening the kind of data a single host will accept.

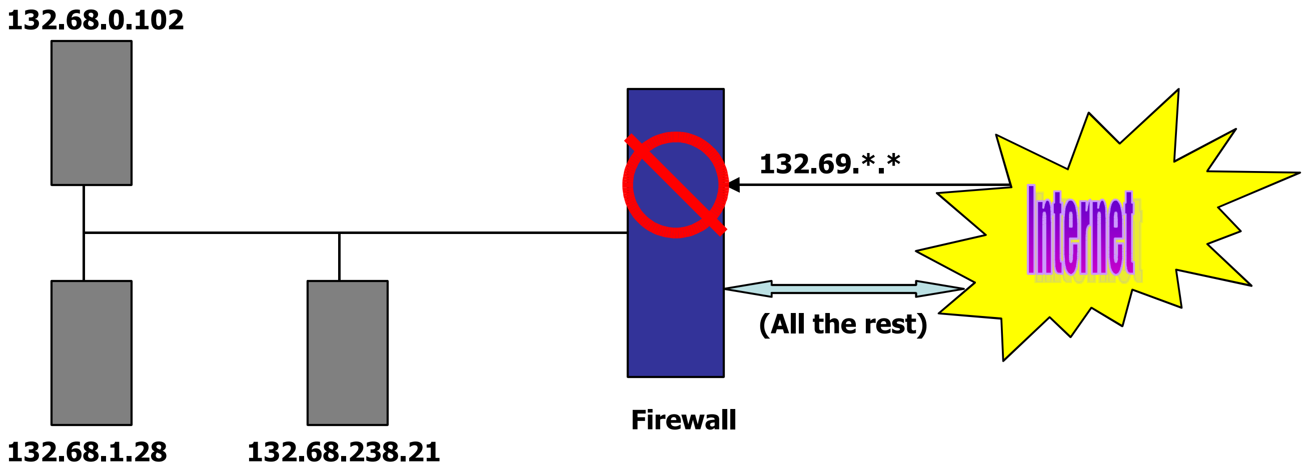
- Packet Filtering Gateway: The main type which is discussed in this lecture.

A type of firewall: Packet Filtering Gateway

- *Packet filtering gateways* or *screening routers*: This type is the simplest, and in some situations is the most effective type of firewall. A packet filtering gateway controls access to packets, based on packet addresses (source and/or destination), or specific transport protocol type (such as HTTP web traffic).

- Packet filters do not analyze the contents of the packet. They block or accept packets solely on the basis of the IP addresses and ports. Thus, any details in the packet's data field (for example, using Telnet command in order to run other blocked services, such as HTTP), is beyond the capability of a packet filter.

- The primary (dis)advantage of packet filtering routers is the combination of simplicity and complexity. The router's inspection is simplistic. To perform sophisticated filtering, the filtering rules set needs to be very detailed. However, a detailed set of rules will become complex and might prone to error. For example, blocking all port 23 traffic (Telnet) is simple and straightforward, but if some Telnet traffic is to be allowed, each IP address from which it is allowed must be specified in the rules. This way, the rule set can become very long.

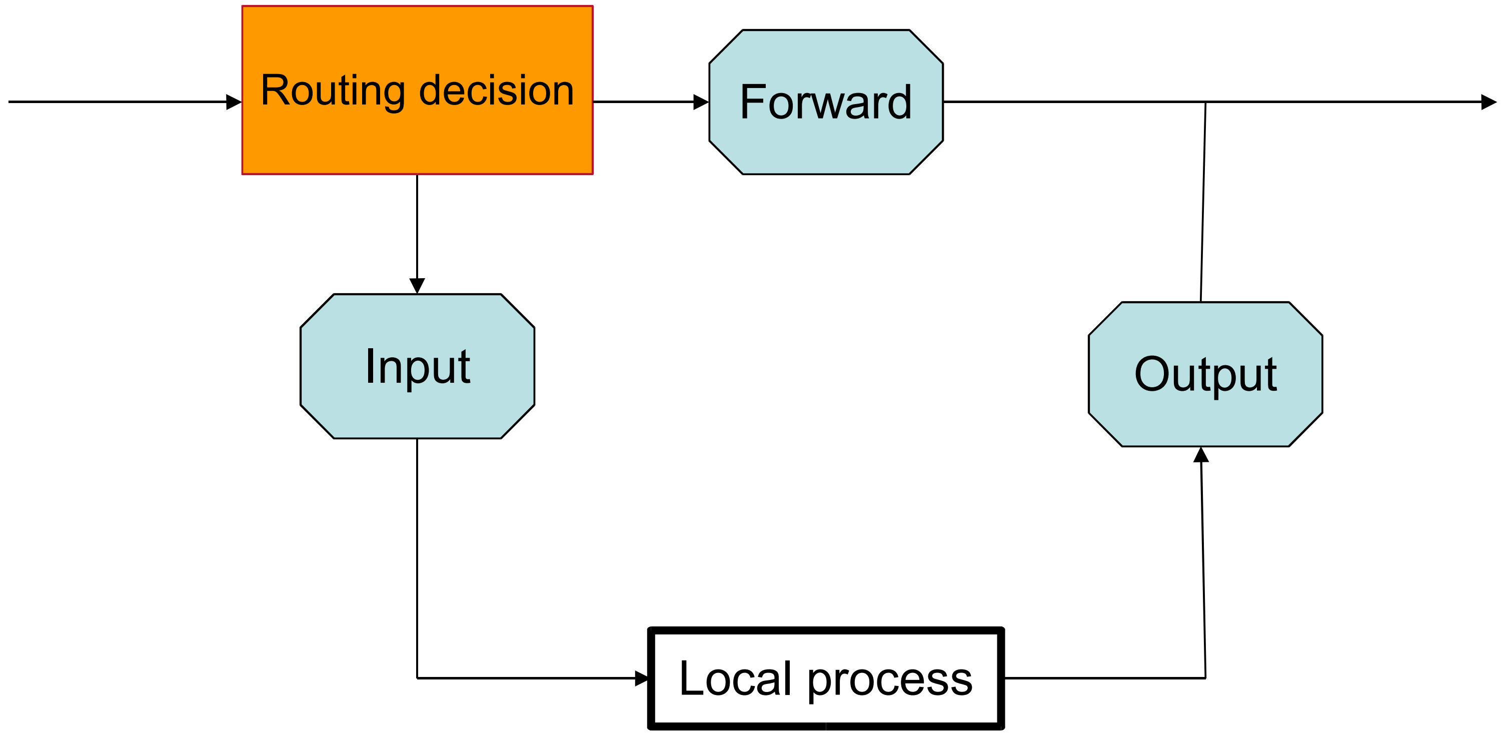


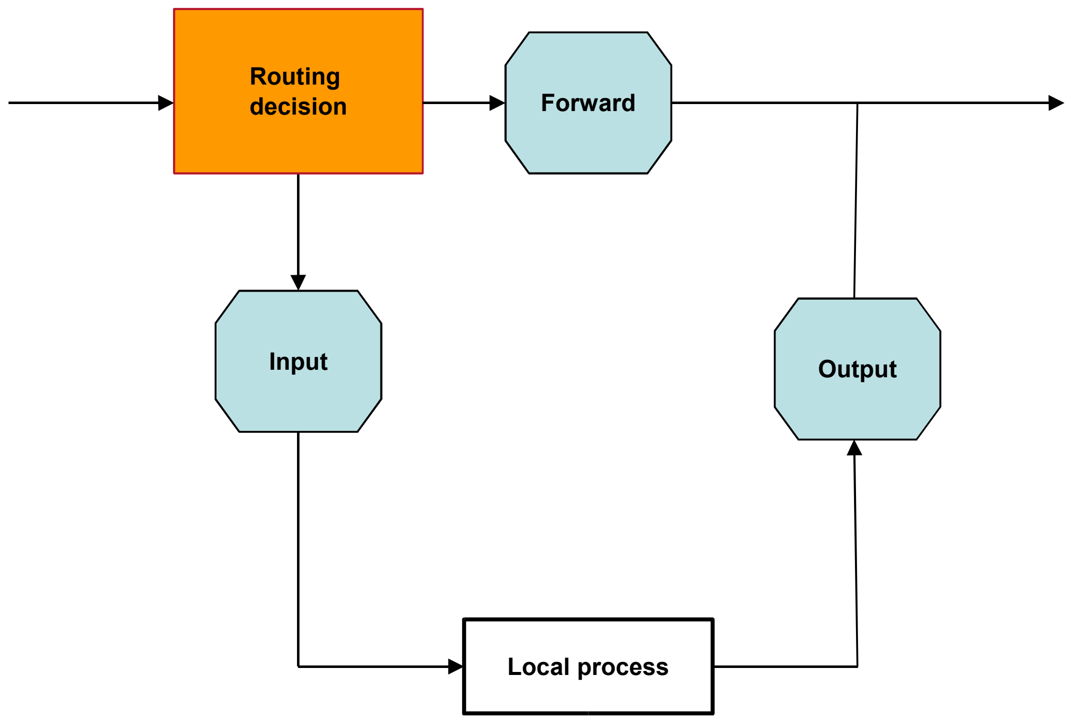
- Here we introduce a packet filtering firewall called IPTables which does all this, and much more. Under Linux, packet filtering is built into the kernel (as a kernel module, or built right in). This lecture doesn't discuss regarding its installation. Details about it you can find in http://iptables-tutorial.frozentux.net/iptables-tutorial.html#PREPARATIONS

The concepts of IPTables (or: How packets traverse the filters)

The kernel starts with three lists of rules in the ‘filter’ table. These lists are called **firewalls chains** (or simply, **chains**). The three chains are called **INPUT**, **OUTPUT** and **FORWARD**.

In kernel 2.4 and 2.6, the chains are arranged like that:

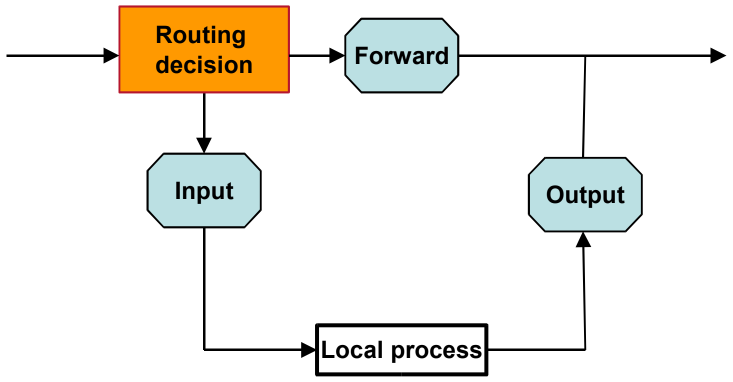




The three octagons represent the three chains mentioned above. When a packet reaches an octagon in the diagram, the chain is examined to decide the fate of the packet. If the chain says to DROP the packet, it is killed there, but if the chain says to ACCEPT the packet, it continues to traverse the diagram.

Chains

What exactly is a chain? A chain is an ordered set of **rules**. Each rule says, “If the packet header looks like this, then here’s what you have to do with the packet”. If the rule doesn’t match the packet, then the next rule in the chain is consulted. The first rule is always the first line in a chain, and it is checked always checked first (it is always recommended to make the rules ordered in the priority of checking. “General” rules you might want to put first). Finally, if there are no more rules to consult, then the kernel looks at the chain **policy** to decide what to do. In a security-conscious system, you actually don’t trust the unmatched-rule packet, and usually tell the kernel to DROP the packet.



1) When a packet comes in (through the Ethernet card, for example), the kernel first looks at the destination of the packet. This is called ‘routing’.
2) If it is destined for this box (if it is into the firewall’s box, for example), then the packet passes downwards in the diagram, to the INPUT chain. If it passes this, any processes waiting for that packet will receive it.
3) Otherwise, if the kernel has no forwarding enabled (you would want to forward a packet if you are a router, or a gateway), or it doesn’t know how to forward the packet, the packet is dropped. If forwarding is enabled, the packet is destined for another network interface (if you have another one), then the packet goes rightwards on our diagram to the FORWARD chain. If it is ACCEPTed, it will be sent out.
4) Finally, a programming running on the box can send network packets. These packets pass through the OUTPUT chain immediately. If it says ACCEPT, then the packet continues out to whatever interface it is destined for.

Using IPTables

Iptables has a fairly detailed manual page (man iptables), if you need more details in particular.

You always start with the three built-in chains INPUT, OUTPUT and FORWARD which you can't delete. There are several things that you can do with iptables:

6) -N – to create a new chain
7) -X – to delete an **empty** chain
8) -P – to change the policy for a built-in chain
9) -L – to list the rules in a chain
10) -F – to flush the rules out of a chain

In order to manipulate rules inside a chain, you may use one of these options:

12) -A – to append a new rule to a chain
13) -I – to insert a new rule at some position in a chain
14) -R – to replace a rule at some position in a chain
15) -D – to delete a rule at some position in a chain, or the first that matches

Filtering Specifications

Specifying Source and Destination IP Addresses

Source (-s, --source or –src) and destination (-d, --destination or --dst) IP addresses can be specified in four ways. The most common way is to use the full hostname, such as ‘localhost’ or ‘www.haifux.org’. The second way is to specify the IP address such as ‘127.0.0.1’ or ‘132.69.253.254’.

The third and fourth ways allow specification of a group of IP addresses, such as ‘200.21.213.0/24’ or ‘200.21.213.0/255.255.255.0’ and will not be discussed here. However, to check your skills, you can try using ‘0/0’ which means, “any address at all”.

This line:

```
iptables –A INPUT –s 0/0 –j DROP
```

Will result in DROPping any packet which arrives to your box. Please, don’t try this at home..

However, if you insisted in doing so, you can remove this rule by writing:

```
iptables –D INPUT –s 0/0 –j DROP
```

or

```
iptables –D INPUT 1
```

(if this is your first rule in the INPUT chain)

Specifying Protocol

The protocol can be specified with the –p (or –protocol) flag. Protocols can be a number (if you know the numeric protocol values for IP), or a name for the special cases of ‘TCP’, ‘UDP’ or ‘ICMP’. Case doesn’t matter. You can write ‘tcp’ instead of ‘TCP’, etc.

Specifying an Interfance

The -i (--in-interface) and ‘o’ (--out-interface) options specify the name of an interface to match. An interface is the physical device the packet came in on

(‘-i’), or is going out on (‘-o’). You can use the ifconfig command to list the interfaces which are working at the moment, and to specify rules for your interfaces.

Naturally, packets traversing the INPUT chain don’t have an output interface, so any rule using ‘-o’ will never match there. Similary, packets which traverse the OUTPUT chain don’t have an input interface, so ‘-i’ will never match there.

Only packets traversing the FORWARD chain have both an input and output interface.

Specifying Inversion

Many flags can have their arguments preceded by ‘!’ (‘not’). For example, ‘-s !localhost’ matches any packet not coming from localhost, or ‘-p !tcp’ matches any protocol which is NOT TCP.

There are more advanced options using IPTables, such as fragments, and extensions to IPTables, such as specifying protocol flags, which will not be discussed in here.

An example of using IPTables

```
#we want to start from clean tables.
#This way, if you rerun this script, you do not double the rules
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
iptables -F INPUT
iptables -F FORWARD
iptables -F OUTPUT
#We allow only an already established connections to get in, and we don't let
#  the outside world to make a new connection (stateful inspection, which is a
# must in this case, but not discussed here)
 iptables -A INPUT -p udp -m state --state ESTABLISHED,RELATED -j ACCEPT
 iptables -A INPUT -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
 iptables -A INPUT -p icmp -m state --state ESTABLISHED,RELATED -j ACCEPT

# then we permit the loopback interface (we assume that we trust it in here)
 iptables -A INPUT -s 127.0.0.1 -j ACCEPT
```

```
#The following 2 rules enable the passive and active ftp port command
#(that opens a data stream).
iptables -A INPUT -p tcp --sport 21 -m state --state ESTABLISHED –j ACCEPT
iptables -A INPUT -p tcp --sport 20 -m state --state ESTABLISHED,RELATED  -j
            ACCEPT

# We also allow HTTP, SMTP, FTP, SSH, POP3 and IMAP requests
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 20:22 -j ACCEPT
iptables -A INPUT -p tcp --dport 110 -j ACCEPT
iptables -A INPUT -p tcp --dport 143 -j ACCEPT

# we allow broadcasts.
iptables -A INPUT -p ALL -d $LAN_BCAST_ADDRESS -j ACCEPT

# we allow outgoing UDP, TCP, ICMP, FTP, SSH, Telnet, SMTP, HTTP and HTTPs
iptables –A OUTPUT –p udp ACCEPT
iptables –A OUTPUT –p tcp --sport 1024: --dport 1024: -j ACCEPT
iptables –A OUTPUT –p icmp –j ACCEPT
iptables –A OUTPUT –p tcp --dport 20:23 –j ACCEPT
iptables –A OUTPUT –p tcp --dport 25 –j ACCEPT
iptables –A OUTPUT –p tcp --dport 80 –j ACCEPT
iptables –A OUTPUT –p tcp --dport 443 –j ACCEPT
```

# set default policies for the INPUT, FORWARD and OUTPUT chains. Since we don’t want any unknown packet to get into our system, the default policy is DROP

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

Some advice on packet filtering design

- Common wisdom in making your computer secured, is to block everything, and then open up holes as necessary. What is not known, or explicitly allowed, is simply prohibited.
- Do not run services which you don't need, even if you have blocked access to them. It can result in false alarms and make your firewall more difficult to configure.
- If you are creating a dedicated firewall, start by running nothing, then block all packets, then add services and let packets get through as required.
- You can combine different firewalls in different levels. For instance – a packet filtering firewall, together with a proxy. This way even when a packet gets in (or out), it passes through the application proxy which checks the packet in the application level (checking your email for viruses in your mail server, before it arrives to your box, for example).
- Logging is useful when setting up a firewall if something isn't working. See the manual of iptables for more details.
- In general, try to be minimalist when you build a firewall. Simple set of rules in a firewall, might be everything you need for your secured home/office network.

Thank you!