Security principles

Firewalls and NAT

Core Concepts

Host security vs Network security

What is a firewall?

What does it do?

Where does one use it?

At what level does it function?

What kinds of firewalls are there?

What about NAT?

Host vs Network Security

- Host security
  - Rules, policies and practices that are applied to the host itself
    - Passwords, ACLs, roles and groups, system integrity (checksum), ressource audit, encryption
  - If not automated, doesn't scale
  - No global way of enforcing which services can be reached from the network
- Threats: buffer overflow, brute force, social engineering

Host vs Network Security

- Network security
  - Rules, policies and practices that target network traffic and services
    - Rules/Filters, traffic analysis, penetration testing, anti-spoofing, encryption
  - Doesn't concern with local security *on the host*
  - Global way of protecting access to the resources of a network
- Threats: DoS, portscan, buffer overflow, spoofing, sniffing

What is a Firewall?

Device (software, hardware) enforcing selective access (allow, deny) between different security domains, based on rules

- In plain speak: traffic police

What Does a Firewall Do?

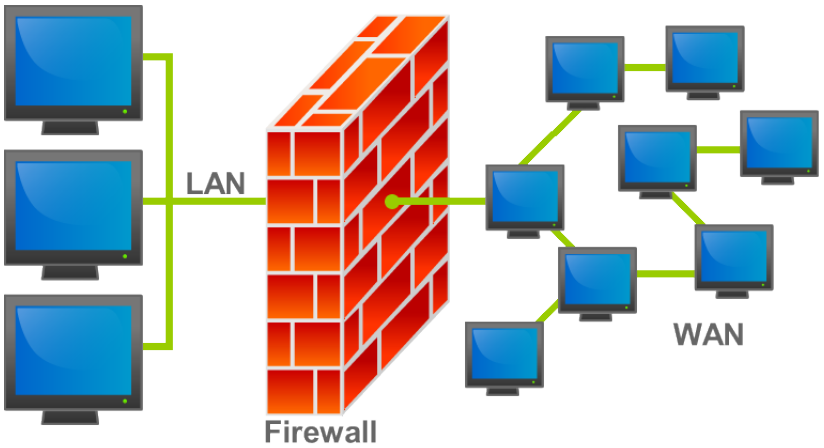
- Selectively grant or reject access to network traffic between hosts or networks belonging to different security domains
- Domains can be local or remote (LAN, Internet)
- The rules can apply to the traffic at different levels
  - The rules may be explicit or implicit (difference between stateful and stateless)

What Does a Firewall Do?

- The goal is to protect your network from undesired traffic
- It doesn't matter if an attack originates from the inside or outside...
  - You have a responsibility to protect the rest of the Internet from *your* systems
  - Maybe your users are well behaved
    - But you may have been hacked, or infected by a virus or spyware sending spam

Where Does One Use a Firewall?

The firewall must be located between security domains



Example above: between Internet and LAN

Where Does One Use a Firewall?

The firewall acts as a ”choke point” for all traffic (just like a router in a simple network setup)

- Enforces traffic rules in one location

Advantage: single point of control, no need to configure rules on every machine

Downside: single point of *failure*!

What About NAT?

NAT is a kind of transformation performed on packets to-and-from a network

NAT requires state keeping

Typically, one uses PAT (Port Address Translation)

- # of private IPs > # of public IPs
- ”overload” the public IPs by using multiple source ports to keep track of the private IPs

What About NAT?

- NAT is not synonymous with anonymity and security!
  - They are a side effect
  - NAT alone does not protect from attacks if the attacker is on your external network, and sending packets to your inside hosts via the firewall
  - If the firewall doesn't explicitly reject this traffic, it might get through!

Summary

- Firewalls are located between different parts of a network
  - Can be ”inside” / ”outside”, but it can also be ”sales” / ”engineering” / ”production”
- Firewalls can operate at different levels
  - They can be more or less aware of what's going on inside the packets
  - Stateful firewalls are to be preferred over simple packet filters

Questions

?

A Firewall for Linux: iptables

Current (2010) Linux distributions come with the ip_tables packet filter either built-in to the kernel (versions 2.4 and 2.6) or available as a module.

The command line interface is:

`iptables` (IPv4)

`iptables6` (IPv6)

What Can iptables Do?

With iptables you can:

- Create firewall rules
- Configure Network Address Translation
- “Mangle” packets on the fly
- Can be configured “by hand,” via scripts, using third party tools
- Block packets until a user is authenticated
- Etc…

For now we will concentrate on firewalls

iptables Rulesets: What to Filter

As you create iptables rules remember you must decided what protocols you are filtering:

- tcp
- udp
- icmp
- and, specific port numbers

iptables has many protocol specific options

iptables Complexity

The first thing you should do to understand iptables after this class is:

```
man iptables
```

Really! Do this.

There are many, many, many options available.

There are many, many, many custom modules available.

The Three iptables Tables

“filter” table

The iptables *filter* table is the default table for rules, unless otherwise specified.

“nat” table

The *network address translation or nat* table is used to translate the source or destination field in packets.

“mangle” table

The *mangle* table is used to alter certain fields in the headers of IP packets.

iptables filter Table Chains

The *filter* table has three built-in chains that packets traverse:

1. **INPUT:** Packets destined for the host.
2. **OUTPUT:** Packets created by the host to send to another system
3. **FORWARD:** Packets received by the host that are destined for another host

You can create your own chains as well. We'll do this later.

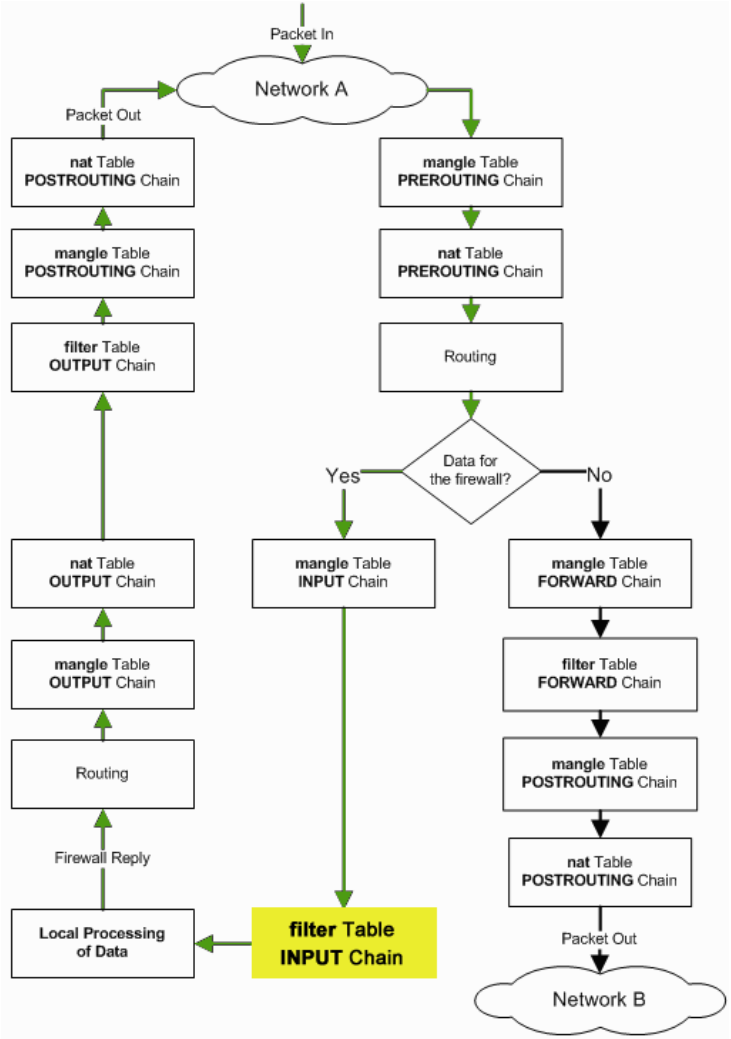
Packet Traversal of iptables

On the next slide you can see where the *filter* table and the INPUT, FORWARD and OUTPUT chains reside within iptables.

If you don't specify any *nat* or *mangle* table rules, then packets traverse these tables with no affect.

For initial firewalls we concentrate on applying *packet filtering* rules to packets traversing the *filter* table, INPUT chain.

iptables Packet Traversal



- For this introduction to iptables we spend most of our time applying rules in the yellow box – or, for packets going in to our host destined for local processes.
- For packets leaving from our host we would filter these in the filter table, OUTPUT chain.
- For packets passing through our host to another network (such as using NAT) we filter these in the filter table, FORWARD chain.

Diagram courtest of:
http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch14_:_Linux_Firewalls_Using_iptables

iptables Packet Traversal cont.

As you can see we are just getting started with iptables.

It's important to understand what the other tables and chains are for.

In the next few slides we will describe the complete steps a packet takes as it is examined by the Linux kernel depending on its final destination.

Tables courtesy of http://www.faqs.org/docs/iptables/traversingoftables.html

Packets Destined for Local Host

Step	Table	Chain	Comment
1			On the wire (e.g., Internet)
2			Comes in on the interface (e.g., eth0)
3	mangle	PREROUTING	This chain is normally used for mangling packets, i.e., changing TOS and so on.
4	nat	PREROUTING	This chain is used for DNAT mainly. Avoid filtering in this chain since it will be bypassed in certain cases.
5			Routing decision, i.e., is the packet destined for our local host or to be forwarded and where.
6	mangle	INPUT	At this point, the mangle INPUT chain is hit. We use this chain to mangle packets, after they have been routed, but before they are actually sent to the process on the machine.
7	filter	INPUT	This is where we do filtering for all incoming traffic destined for our local host.
8			Local process/application (i.e., server/client program)

Packets Coming from Local Host

Step	Table	Chain	Comment
1			Local process/application (i.e., server/client program)
2			Routing decision. What source address to use, what outgoing interface to use, and other necessary information that needs to be gathered.
3	mangle	OUTPUT	This is where we mangle packets, it is suggested that you don't filter in this chain as it can have side effects.
4	nat	OUTPUT	This chain can be used to NAT outgoing packets from the firewall itself.
5	filter	OUTPUT	This is where we filter packets going out from the local host.
6	mangle	POSTROUTING	The POSTROUTING chain in the mangle table is mainly used when we want to do mangling on packets before they leave our host, but after the actual routing decisions. This chain will be hit by both packets just traversing the firewall, as well as packets created by the firewall itself.
7	nat	POSTROUTING	This is where we do SNAT. You should not filter here.
8			Goes out on some interface (e.g., eth0)
9			On the wire (e.g., Internet)

Forwarded Packets

Step	Table	Chain	Comment
1			On the wire (i.e., Internet)
2			Comes in on the interface (i.e., eth0)
3	mangle	PREROUTING	This chain is normally used for mangling packets, i.e., changing TOS and so on.
4	nat	PREROUTING	This chain is used for DNAT mainly. SNAT is done further on. Avoid filtering in this chain since it will be bypassed in certain cases.
5			Routing decision, i.e., is the packet destined for our local host or to be forwarded and where.
6	mangle	FORWARD	The packet is then sent on to the FORWARD chain of the mangle table. This can be used for very specific needs, where we want to mangle the packets after the initial routing decision, but before the last routing decision made just before the packet is sent out.

7 Continued on the following slide ➔

Forwarded Packets cont.

Step	Table	Chain	Comment
7	filter	FORWARD	The packet gets routed onto the FORWARD chain. Only forwarded packets go through here, and here we do all the filtering. Note that all traffic that's forwarded goes through here (not only in one direction), so you need to think about it when writing your rule-set.
8	mangle	POSTROUTING	This chain is used for specific types of packet mangling that we wish to take place after all kinds of routing decisions has been done, but still on this machine.
9	nat	POSTROUTING	This chain should first and foremost be used for SNAT. Avoid doing filtering here, since certain packets might pass this chain without ever hitting it. This is also where Masquerading is done.
10			Goes out on the outgoing interface (i.e., eth1).
11			Out on the wire again (i.e., LAN).

iptables Summary

As you build rules for iptables be mindful of how incoming, outgoing and forwarded packets traverse the various tables and chains.

iptables is a very powerful tool. Starting simple and building as you learn and understand more about iptables is a good strategy.

Questions

?

Building a Firewall Ruleset

Two basic approaches:

1. Allow everything by default, filter the ”bad” things
   - Very quickly unmanageable!
   - What is ”bad” ?
2. Block everything by default, allow only what you know should be allowing
   - More work in the beginning
   - Easier in the long run

Building a Firewall Ruleset

Some firewalls have a ”first match” principle, others ”last match”:

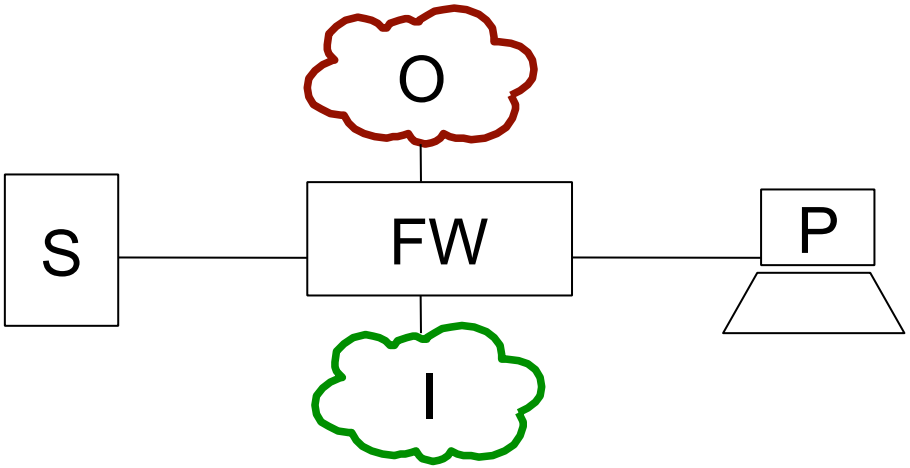
```
1. allow ip from A to B
2. deny ip from any to any
```

In the above example, ip traffic from A to B will be allowed if the firewall software stops on the first match (rule 1).

If the firewall is last match, the traffic will be denied (last rule to match is 2)

Building a Firewall Ruleset

Be careful with order and logical operators




```
allow tcp from not S to I

allow tcp from P to I
```

You have just opened for traffic from O to I!

Building a Firewall Ruleset

- Be careful not too be too conservative when filtering certain protocols
  - Many ICMP messages should be allowed as they can carry important information about network status (congestion, reachability)
  - Most stateful firewalls automatically allow ICMP messages that are related to a known active ”connection”
- DNS is much more than ”512 byte UDP packets on port 53”

Remember...

- A firewall with very strict rules doesn't help if users are allowed to ssh from computer to computer
  - Once an evildoer is inside the network, it can be too late...
- ”A hard crunchy shell around a soft chewy centre” – Bill Cheswick / Steve Bellovin
- It's not enough to *only* focus on network security!

Questions

?

Building an iptables Ruleset

There are so many ways to build rulesets with iptables, many available tools and even more opinions about what's best!

But, in general…

1. Create an initial iptables ruleset using the `iptables` command line interface (CLI).
2. Save your ruleset out to a file.
3. Configure your box to use the ruleset at system start.
4. Edit the saved ruleset file to create more complex rulesets, make updates, etc.
5. Test your ruleset! Critical. Be sure it works as expected.

Complexity and Power

A nice feature of iptables is the ability to filter on complex and dynamic protocols and actions, such as ftp, irc, number of failed attempts, connection attempts by ip or ranges and much more.

A Simple Example

Block ping (icmp echo request) locally:

```
iptables -A INPUT -p icmp --icmp-type echo-request -i lo -j DROP
```

What's going on?

1. -A → Append this rule to the INPUT chain
2. -p → protocol
3. --icmp-type echo-request
4. -i → input interface
5. -j DROP → jump to the target DROP

A Simple Example cont.

Remove our ping blocking rule:

```
iptables -D INPUT -p icmp --icmp-type echo-request -i lo -j DROP
```

What's going on?

- -D: "Delete" the following specification

How to test this:

```
ping 127.0.0.1
```

By the way – should you block ping? (NO!!!)

A More Complex Example

Block SSH login attempts after three failures in five minutes:

```
iptables -N SSHSCAN
iptables -A INPUT -p tcp --dport 22 -m state --state NEW -j SSHSCAN
iptables -A SSHSCAN -m recent --set --name SSH
iptables -A SSHSCAN -m recent --update --seconds 300 --hitcount 3 --name \
 SSH -j DROP
```

What's going on here?

This works because iptables is a *stateful* firewall. It remembers packets coming from the same origin address.

A More Complex Example cont.

1. `iptables -N SSHSCAN`

   Create a New chain named “SSHSCAN”

2. `iptables -A INPUT -p tcp --dport 22 -m state --state NEW -j SSHSCAN`

   For the tcp protocol packets connecting on port 22 (SSH) load the “state” module and look for new connections – if this matches, then jump to the next SSHSCAN target.

A More Complex Example cont.

3. `iptables -A SSHSCAN -m recent --set --name SSH`

   For the SSHSCAN chain load the *recent* module which will set and check work based on user-definable fields and timers, then add the source address of the associated packets (--set), and finally specify a list name to use for commands (--name SSH).

A More Complex Example cont.

4. `iptables -A SSHSCAN -m recent --update --seconds 300 --hitcount 3 --name SSH -j DROP`

   Scan the SSH list of IP addresses and see if there have been three separate connection attempts within the last 300 seconds (5 minutes). If there is a match, drop the packet and update the timestamp on the packet.

Complex Rulesets

The last example can be refined to:

Allow certain addresses to be excluded:

```
iptables -A INPUT -p tcp --dport 22 -s $WHITE_LIST_IP -j ACCEPT
```

To log connection attempts:

```
iptables -A SSHSCAN -m recent --update --seconds 300 --hitcount 3
--name SSH -j LOG --log-level info --log-prefix "SSH SCAN blocked: ”
```

More Details available at:

http://www.ducea.com/2006/06/28/using-iptables-to-block-brute-force-attacks/

Basic iptables Commands

A more complete list of commands will be provided during your lab.

Step-by-step instructions for using iptables with Ubuntu will be part of your lab.

`iptables -F`

Flush *all* iptables rules

`iptables -L`

List *all* iptables rules

Basic iptables Commands cont.

```
iptables –L INPUT
```

View all INPUT chain rules

```
iptables -I INPUT -s "201.128.33.200" -j DROP
```

Block an IP address

```
iptables -I INPUT -s "201.128.33.0/24" -j DROP
```

Block a range of IP addresses

```
iptables -I INPUT -s "201.128.33.200” –j ACCEPT
```

Unblock an IP address

```
iptables -A INPUT -p tcp --dport 25 -j DROP
iptables -A INPUT -p udp --dport 25 -j DROP
```

Block access to a port (SMTP) for both tcp and udp

iptables Connection Tracking

The iptables connection tracking feature is the ability to maintain connection information in memory.

It can remember connection states such as established and new connections along with protocol types, source and destination ip address.

You can allow or deny access based upon state.

iptables Connection Tracking cont.

Connection tracking uses four states:

NEW - A Client requesting new connection via firewall host

ESTABLISHED - A connection that is part of already established connection

RELATED - A connection that is requesting a new request but is part of an existing connection.

INVALID - If none of the above three states can be referred or used then it is an INVALID state.

We may use this feature of iptables in our firewall lab.

Questions

?

Some Food for Thought

Complex firewall rule sets need to be broken down and modular. Don't just add!

Tables, groups, macros and variables

Check with your ISP to know what they filter
- for example, it does not help to filter nefarious traffic on your side (downstream) of the connection, if it is a denial of Service - it is too late!

A Firewall for Linux: iptables

The iptables project is located here:

http://www.netfilter.org/projects/iptables/

Extensive documentation is available:

http://www.netfilter.org/documentation/

An Ubuntu iptables HowTo

https://help.ubuntu.com/community/IptablesHowTo

A CentOS (RedHat) iptables HowTo

http://wiki.centos.org/HowTos/Network/IPTables