Security principles

Firewalls and 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), resource 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?

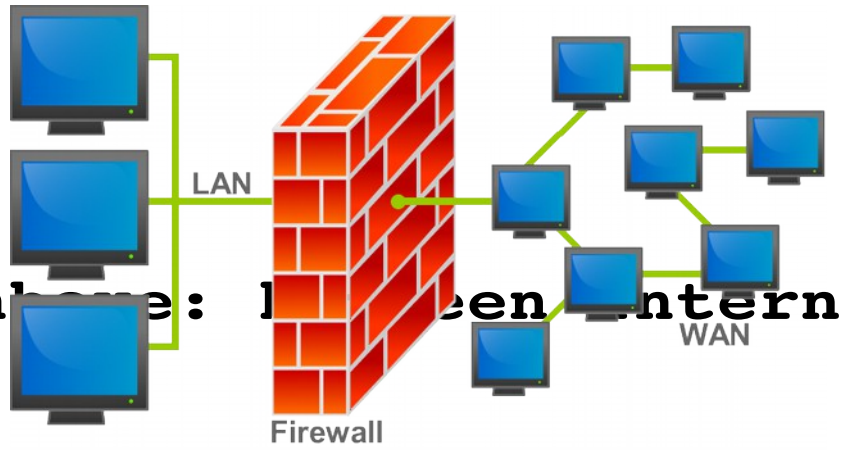
The goal is to protect your network from undesired traffic

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*!

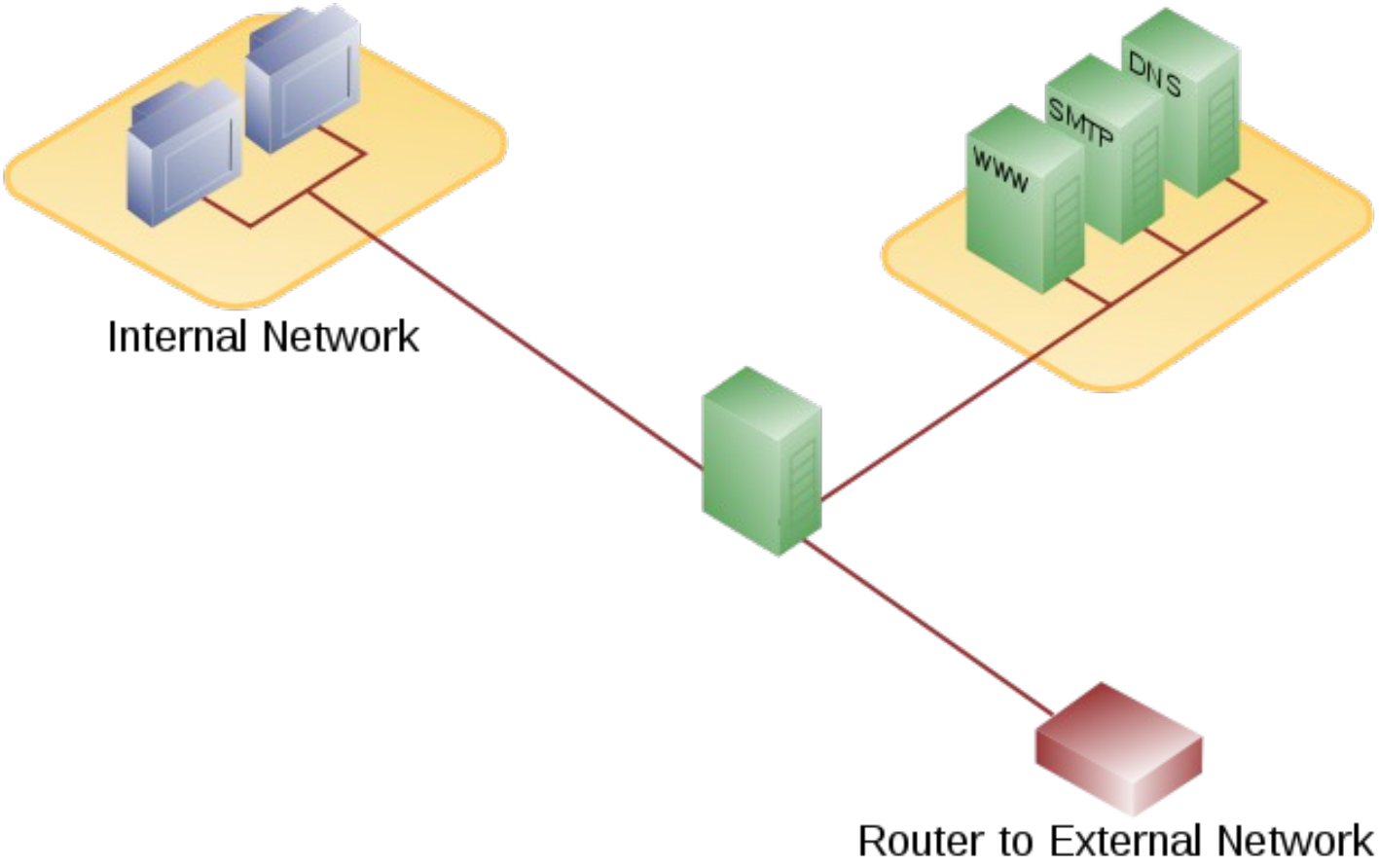
What is a Firewall DMZ?

- **Demilitarized zone (military term)**

  An isolated network that is separated from the internal (trusted) and the external (untrusted) networks.

- **It is used to place hosts which provide services to both the internal and external networks**
- **Hosts in the DMZ can be considered as "semi-trusted"**

DMZ using a three-legged firewall



Why is the DMZ a good idea?

- **If your external-facing servers are compromised, you don't expose your internal trusted network**
- **Provides a clear separation of traffic flows**
  - Reduces the complexity of filtering

What About NAT?

NAT translates source and destination addresses on packets flowing between two networks

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!**
  - Those 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 (with static translations)
  - If the firewall doesn't explicitly reject this traffic, it might get through!

What About NAT?

- **NAT has problems**
  - It breaks certain applications
    - FTP, SIP, H.323, etc.
  - It makes forensic work difficult
    - If somone from your internal network attacks an external host, and you are required to provide the identity of the attacker, you will need to maintain copies of your NAT translation states

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 look not only at the IP headers, but also at the TCP/UDP headers, application headers and contents
- Stateful firewalls are to be preferred over simple packet filters

Questions

?

A Firewall for Linux: iptables

Current Linux distributions come with the ip_tables packet filter either built-in to the kernel (versions 2.4 and up) 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 decide 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.

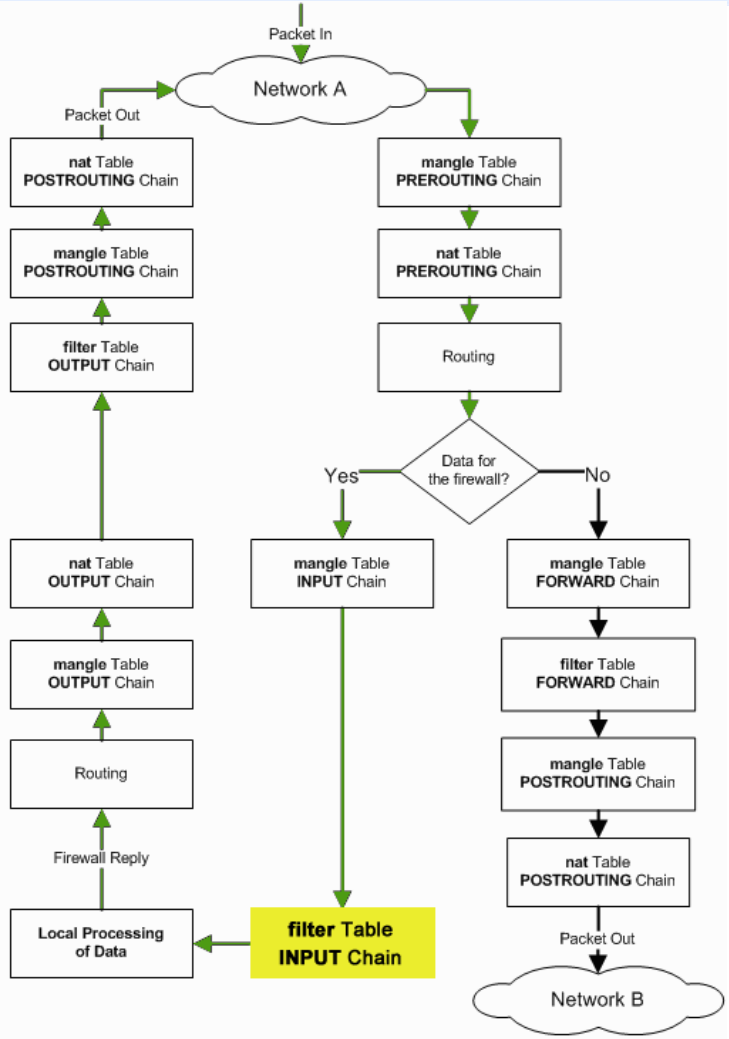
Packet Traversal of iptables

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 effect.

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 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 you 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 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?

- -A → Append this rule to the INPUT chain
- -p → protocol
- --icmp-type echo-request
- -i → input interface
- -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

iptables -F

Flush *all* iptables rules

iptables -L

List *all* iptables rules

iptables –I <chain> [num] <rule>

Insert rules in the given chain, as the given rule number. If no number is specified, insert at the head of the chain

iptables –A <chain> <rule>

Append rules to the end of the selected chain

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.

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