

GNU/Linux Firewalls and Network Security



Ports and Addresses

- Both TCP and UDP connections are uniquely identified by four things
- {source port, destination port, source address, destination address}
- The source and destination addresses are stored in the IP header
- The source and destination ports are stored in the TCP or UDP header

TCP Flags

- **SYN**
 - Synchronize to initiate a connection
- **FIN**
 - The sender is finished sending data
- **ACK**
 - Used to acknowledge previously received data
- **RST**
 - Reset the connection
- **URG**
 - We have urgent data to send
- **PSH**
 - The receiver should pass this data up to the application layer as soon as possible

TCP Establishment

- **TCP connections are established as follows:**
 1. Client sends a TCP packet with the SYN flag set to the server
 2. Server responds with a SYN-ACK
 3. Client responds with an ACK
- This is called a **three-way handshake**

Closing TCP Connections

- **TCP Connections are normally closed as follows:**
 1. Side wishing to close connection sends a TCP packet with the FIN flag set (active close)
 2. Other side responds with a FIN of its own....
 - 3....and an ACK of the FIN it just received
 4. Closing side responds with an ACK

Network Address Translation(NAT)

- NAT means altering one of the source or destination IP addresses in a packet (sometimes ports are altered, as well)
- SNAT
 - Source NAT
 - Source IP address is altered
- MASQUERADE
 - Like SNAT, but for use with a dynamic IP
 - Source IP address is altered
- DNAT
 - Destination NAT
 - Destination IP address is altered

Firewalls

- A firewall is merely a router that limits or filters the flow of traffic between two or more networks
- Host firewalls limit the flow of traffic to or from a specific host, and don't ever forward traffic
- Firewalls or routers with more than one interface are **multi-homed**

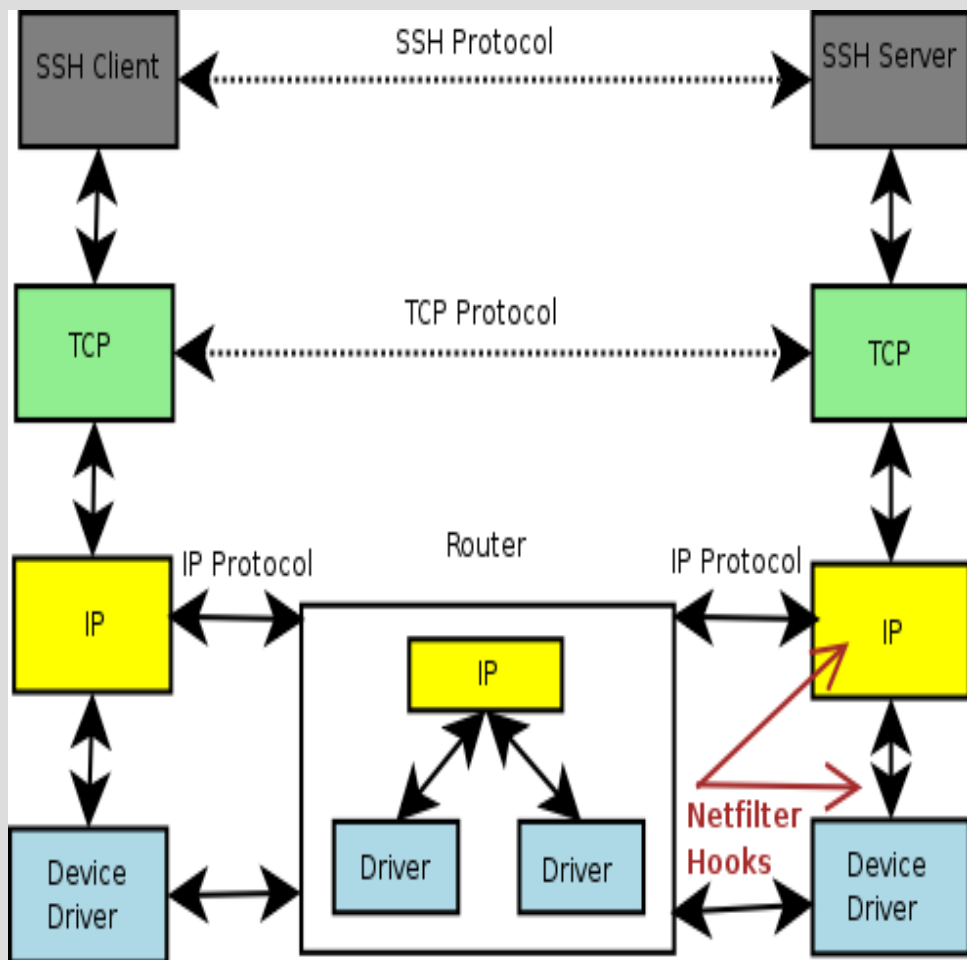
Types of Firewalls

- **Packet filter**
 - Fast and simple
 - ...but they filter packets based on limited information
 - ...and can't handle complex protocols easily
- **Application proxy**
 - Filtering based on application (layer four) data
 - Tend to be slow but secure
 - ...but need one proxy for each application
- **Stateful packet filter**
 - Makes filtering decisions based on the recorded state of a connection
 - Able to handle more complex protocols in a more secure way. The canonical example is active FTP

GNU/Linux Firewalls

- The Linux firewalling toolset is called **iptables**
- It is really a collection of user-space libraries and utilities that interface with the Linux kernel's filtering, NAT, and packet-mangling code
- The in-kernel system is referred to as **netfilter**, for kernels $\geq 2.4.x$
- It can be used as a simple packet filter, or a stateful packet filter
- It has application helpers for protocols that need them

Netfilter in the Network Stack



- Netfilter starts to examine packets after the device driver hands them off to the IP layer
- There are multiple points along the path of a packet where Netfilter will perform some action
- This can be a filter (DROP, etc)
- This can be NAT
- This can be packet mangling

Netfilter Tables & Chains

- **Mangle**
 - Not discussed here, this table contains all five chains
- **NAT**
 - Consists of PREROUTING, OUTPUT, and POSTROUTING chains
- **Filter**
 - Consists of INPUT, OUTPUT, and FORWARD chains

Common Iptables Targets

- **ACCEPT**
 - Allows packet to move to next chain
- **DROP**
 - Drops the packet with no notice to the sender
- **REJECT**
 - Rejects the packet with an ICMP port-unreachable message
 - Valid only in INPUT, FORWARD, OUTPUT
- **RETURN**
 - Returns from the current chain to the calling chain
- **LOG**
 - Logs the packet and continues along the chain

Common Iptables Targets, Continued

- **DNAT**
 - Destination NAT
 - Valid only in PREROUTING,OUTPUT
- **SNAT**
 - Source NAT
 - Valid only in POSTROUTING
- **MASQUERADE**
 - Source NAT for dynamic IP's, the NAT is re-calculated every time the rule matches
 - Valid only in POSTROUTING

Iptables, Useful Syntax

- **iptables [-t {filter | nat}] -A chain {rule}**
 - Appends a rule to a chain
- **iptables -F**
 - Flushes all the chains so the default policy is in effect
- **iptables -X**
 - Deletes all user-defined chains
- **iptables -P {builtin-chain} TARGET**
 - Defines the default policy for the given chain
- **iptables -N {chain name}**
 - Creates a new chain that can then be used as a target (after `-j`)

Iptables Rules

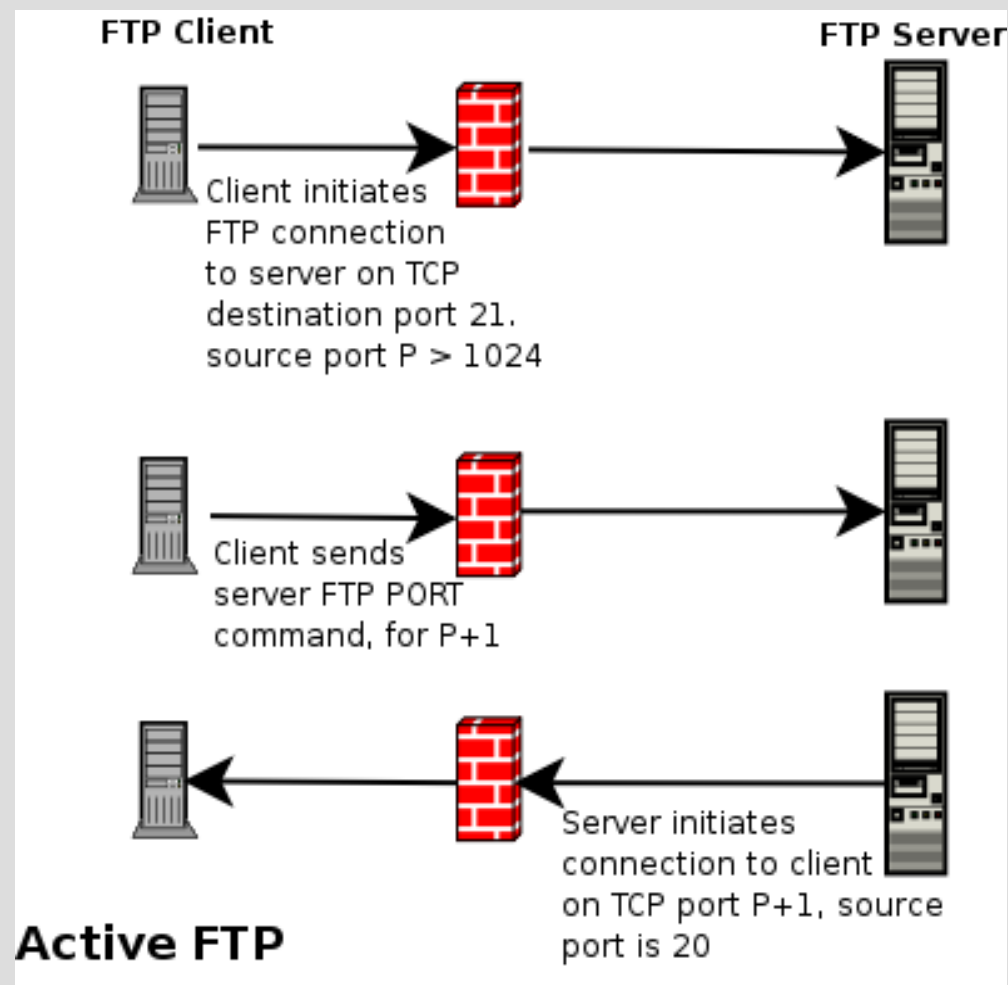
- **-p [!]{protocol}**
 - Protocol can be one of tcp, udp, icmp, or all
- **-s [!]{source}** or **-d [!]{destination}**
 - Source or destination is a hostname, IP address, or IP address/netmask
- **-i [!]{in-interface}**
 - An interface name, as from **ifconfig**
 - INPUT, FORWARD, PREROUTING only
- **-o [!]{out-interface}**
 - OUTPUT, FORWARD, POSTROUTING only

Iptables Rules, Continued

- **-j TARGET**
 - Jump to the given target chain
- **-m {match extension}**
 - Load the given match extension
 - Most common is **-m state**
 - Also have **limit** and **multiport** match extensions

The State Match Extension

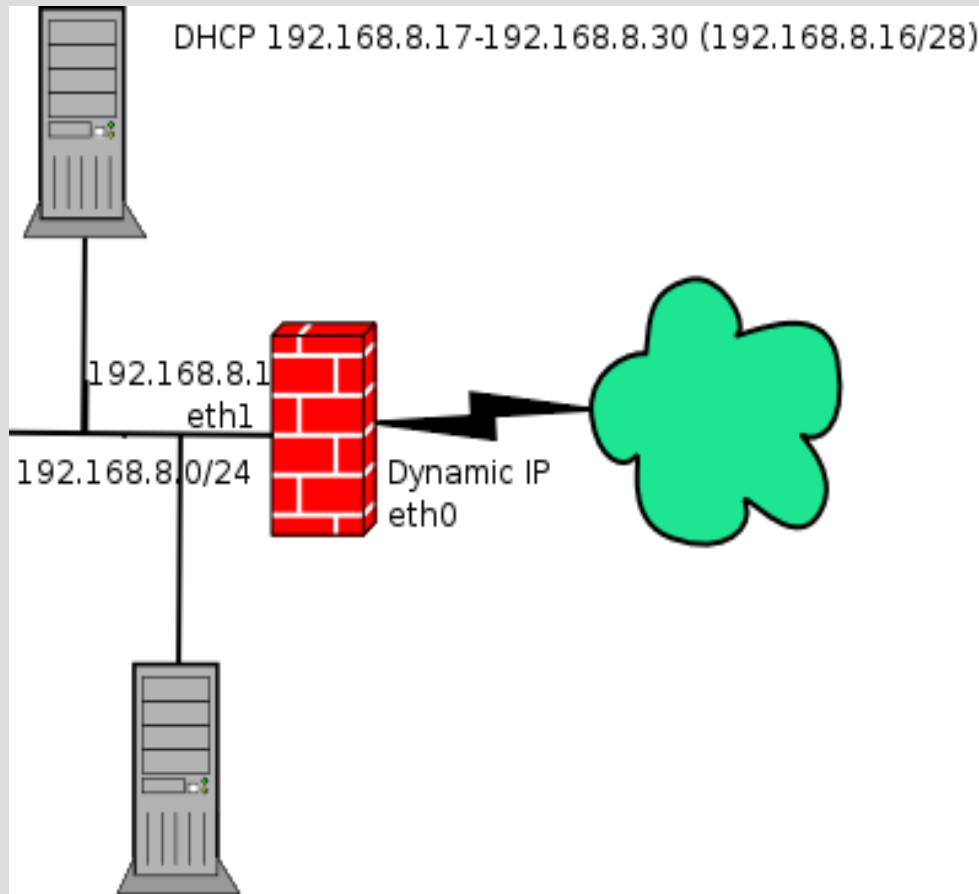
- The state match makes iptables truly useful and enhances security, compared to pure packet filters
- One example is active FTP
- Firewall must open **all** the TCP high ports to the client



The State Match Extension, Continued

- **NEW**
 - A connection that has not yet been seen
- **ESTABLISHED**
 - A packet that is associated with an existing connection
- **RELATED**
 - A packet that represents a new connection, but is related in some way to a prior one
- **INVALID**
 - The packet is invalid in some way

Example Network



- We have some broadband connection to the Internet
- The IP is dynamic
- We use eth0, but this could apply equally well to ppp0 for dial-up access

Default Security Policy

- Default policy is DENY on all filter chains
 - `iptables -P INPUT DROP`
 - `iptables -P OUTPUT DROP`
 - `iptables -P FORWARD DROP`
- Loopback rules
 - `iptables -A INPUT -i lo -j ACCEPT`
 - `iptables -A OUTPUT -o lo -j ACCEPT`
- Drop invalid packets on the input and forward chains
 - `iptables -A INPUT -m state --state INVALID -j DROP`
 - `iptables -A FORWARD -m state --state INVALID -j DROP`

Flush Previous Tables and Chains

```
iptables -F
iptables -X
for table in filter nat mangle
do
    iptables -t $table -F
    iptables -t $table -X
done
```

Input Policy

- **INPUT chain:** What connections do we want to allow to the firewall itself?
- Allow inbound SSH only (from any interface), drop and log everything else
- Make this stateful
 - `iptables -A INPUT -p tcp --dport 22 -m state --state NEW -j ACCEPT`
 - `iptables -A INPUT -m state --state ESTABLISHED, RELATED -j ACCEPT`
 - `iptables -A INPUT -j LOG --log-prefix INPUT DROP:`
- `--log-prefix` will prefix every syslog entry with the given string (29 characters max)

Output Policy

- **OUTPUT chain:** What connections do we want to allow out from the firewall itself?
- Allow outbound SSH and DNS only, to the Internet
- Outbound packets to our internal network are dropped
 - `iptables -A OUTPUT -p tcp --dport 22 -o eth0 -m state --state NEW -j ACCEPT`
 - `iptables -A OUTPUT -p udp --dport 53 -o eth0 -m state --state NEW -j ACCEPT`
 - `iptables -A OUTPUT -p tcp --dport 53 -o eth0 -m state --state NEW -j ACCEPT`

Output Policy, Continued

- `iptables -A OUTPUT -m state --state ESTABLISHED, RELATED -j ACCEPT`
- `iptables -A OUTPUT -j LOG --log-prefix OUTPUT DROP:`

Forward Policy

- **FORWARD chain:** What connections do we want to allow through the firewall?
- Allow outbound SSH, SMTP, FTP, HTTP, HTTPS, and DNS from internal clients, forward nothing inbound
 - `iptables -A FORWARD -p tcp -s 192.168.8.16/28 -m multiport --destination-port 21,22,25,53,80,443 -m match --state NEW -j ACCEPT`
 - `iptables -A FORWARD -p udp -s 192.168.8.16/28 --dport 53 -m match --state NEW -j ACCEPT`
 - `iptables -A FORWARD -m state --state ESTABLISHED, RELATED -j ACCEPT`

Forward Policy, continued

- Remember that the Linux kernel must still be configured to forward packets
 - `echo 1 > /proc/sys/net/ipv4/ip_forward`
OR
 - `sysctl -w net.ipv4.ip_forward=1` OR
 - Put the line `net.ipv4.ip_forward=1` in `/etc/sysctl.conf` and run the command `sysctl -p`

NAT Policy

- Allow internal clients to get out to the Internet, by hiding them behind the firewall's external IP address
- We will use the MASQUERADE target, since we have a dynamic IP address on our external interface
- We must specify the nat table here
 - **iptables -t nat -A POSTROUTING -o eth0 -s 192.168.8.16/28 -j MASQUERADE**
- If we had a static IP, we could use SNAT
 - **iptables -t nat -A POSTROUTING -o eth0 -s 192.168.8.16/28 -j SNAT --to-source \$STATIC_IP**

Rate Limiting

- A good use of the rate-limiting match extension is to limit the amount of logging that we do
 - `iptables -A INPUT -j LOG --log-prefix INPUT DROP: -m limit --limit 10/minute --limit-burst 10`
 - `iptables -A INPUT -j DROP`
- This allows an initial burst of 10 packets to be logged, then rate limiting kicks-in and will only allow 10 packets/minute to be logged

User-Defined Chains

- We can use a user-defined chain with rate-limiting to protect against syn-flood attacks
 1. `iptables -N SYN_PROT`
 2. `iptables -A INPUT -p tcp --syn -j SYN_PROT`
 3. `iptables -A SYN_PROT -p tcp --syn -m limit --limit 3/second --limit-burst 10 -j RETURN`
 4. `iptables -A SYN_PROT -j DROP`
- TCP SYN packets are matched and pulled out of the input chain and sent off to the SYN_PROT chain by **rule #2**
- **Rule #3** limits new connections to 3/second, with an initial burst of 10, sending packets back to the input chain for further processing
- All other TCP SYN packets are dropped by **rule #4**

Network/Host Security

- **Tips for the Linux security administrator**
 - Do not rely on a firewall for all of your network's security
 - Firewalls, by definition, have to allow some traffic through
 - Crackers will typically use these holes and attack application layer vulnerabilities, ignoring your firewall
 - Use the **defense in depth** principle
 - Harden your firewall, DMZ, and internal hosts, or at least quarantine insecure hosts (road warriors)
 - Use host and/or network-based IDS (Snort, Tripwire)
 - Use application proxies or filters as appropriate

Network/Host Security

- **Defense in depth, continued**
 - Restrict outbound traffic from internal hosts
 - Restrict traffic across VPN's
 - Open VPN tunnels are a big source of malware
 - Use strong authentication
 - DSA (SSH v2) keys for SSH, with no password authentication allowed
 - Don't forget about physical security
 - Restrict remote administrative access to certain IP addresses
 - Don't use plain-text protocols where possible
 - Telnet, Berkeley r-tools, FTP, HTTP, POP, IMAP
 - Use SSH, scp, sftp, HTTPS, IMAPS

Network/Host Security

- **Defense in depth, continued**
 - Keep OS security patches up to date
 - **apt-get update && apt-get -u upgrade**
 - **RHN**
 - Keep detailed logs of appropriate traffic
 - Use remote syslog to send firewall log files to another host
 - This works well if you have multiple firewalls
- Get on your distro's security update mailing list
- Read the security guides for your distro
 - Red Hat and Debian have excellent security guides

Copyright & License

This work is Copyright © 2004 Maxwell Consulting Services, LLC (<http://turinglabs.com>) and is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/1.0/> or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.