Firewalls

Linux System Administration
Fall 2022

The purpose of Firewalls

- Control packet processing
  - Input - packets destined for a local address
  - Output - packets with source a local address
  - Forward - packets with source and destination that are not local
- Enforce connectivity policy

Defense in Depth

- Filter at the router
  - Limit access to DMZ and LAN
- Filter at each machined
  - Limit DMZ machine functionality

Security is hard

- Don't write firewall rules if you don't know what you are doing
  - Don't assume that after this class, you can write firewall rules!!!
- Be paranoid
  - Don't trust the network

Linux Firewalls

- `ipchains`
  - The original (stateless)
- `iptables`
  - Mainstay of most firewalls
- `nftables`
  - new hotness
- `netfilter`
  - Kernel module that implements packet filtering

Wrappers around iptables/nftables

- `firewalld` Redhat
- `ufw` uncomplicated firewall Debian
- `shorewall` simplified rule generatore
- `pfSense` popular BSD firewall
- `fail2ban` dynamic blocking

Testing firewalls

- `ping` & `traceroute` basic connectivity
- `nmap` network scanner
  - scan for TCP on 100.64.42.0/24
    ```
    nmap -sS 100.64.42.0/24
    ```
  - fast scan all ports on 100.64.42.3
    ```
    nmap -T4 -n -Pn 100.64.42.3
    ```
- ***It matters where you run the test from***

nmap -n -sS -sU 100.64.41.0/24

```
Nmap scan report for 100.64.41.1
Host is up (0.040s latency).
Not shown: 1997 closed ports
PORT    STATE         SERVICE
22/tcp  open          ssh
67/udp  open|filtered dhcps
123/udp open|filtered ntp

Nmap scan report for 100.64.41.2
Host is up (0.037s latency).
Not shown: 1997 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
111/tcp open  rpcbind
111/udp open  rpcbind

Nmap scan report for 100.64.41.3
Host is up (0.040s latency).
Not shown: 1995 closed ports
PORT    STATE         SERVICE
22/tcp  open          ssh
80/tcp  open          http
111/tcp open          rpcbind
68/udp  open|filtered dhcpc
111/udp open          rpcbind

Nmap scan report for 100.64.41.4
Host is up (0.034s latency).
Not shown: 1996 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
111/tcp open  rpcbind
111/udp open  rpcbind

Nmap scan report for 100.64.41.6
Host is up (0.037s latency).
Not shown: 1994 closed ports
PORT    STATE         SERVICE
22/tcp  open          ssh
53/tcp  open          domain
111/tcp open          rpcbind
53/udp  open          domain
68/udp  open|filtered dhcpc
111/udp open          rpcbind
```

Stateless vs. Statefull

- Stateless
  - Packets define what is allowed
  - Simplifies work for developers and improves performance
- Statefull (connection tracking)
  - Decision may depend on previous packets
  - Greatly simplifies writing firewall for user
    - Allow out what was allowed in
    - Absolutely required for complex protocols like ftp

nftables vs. iptables

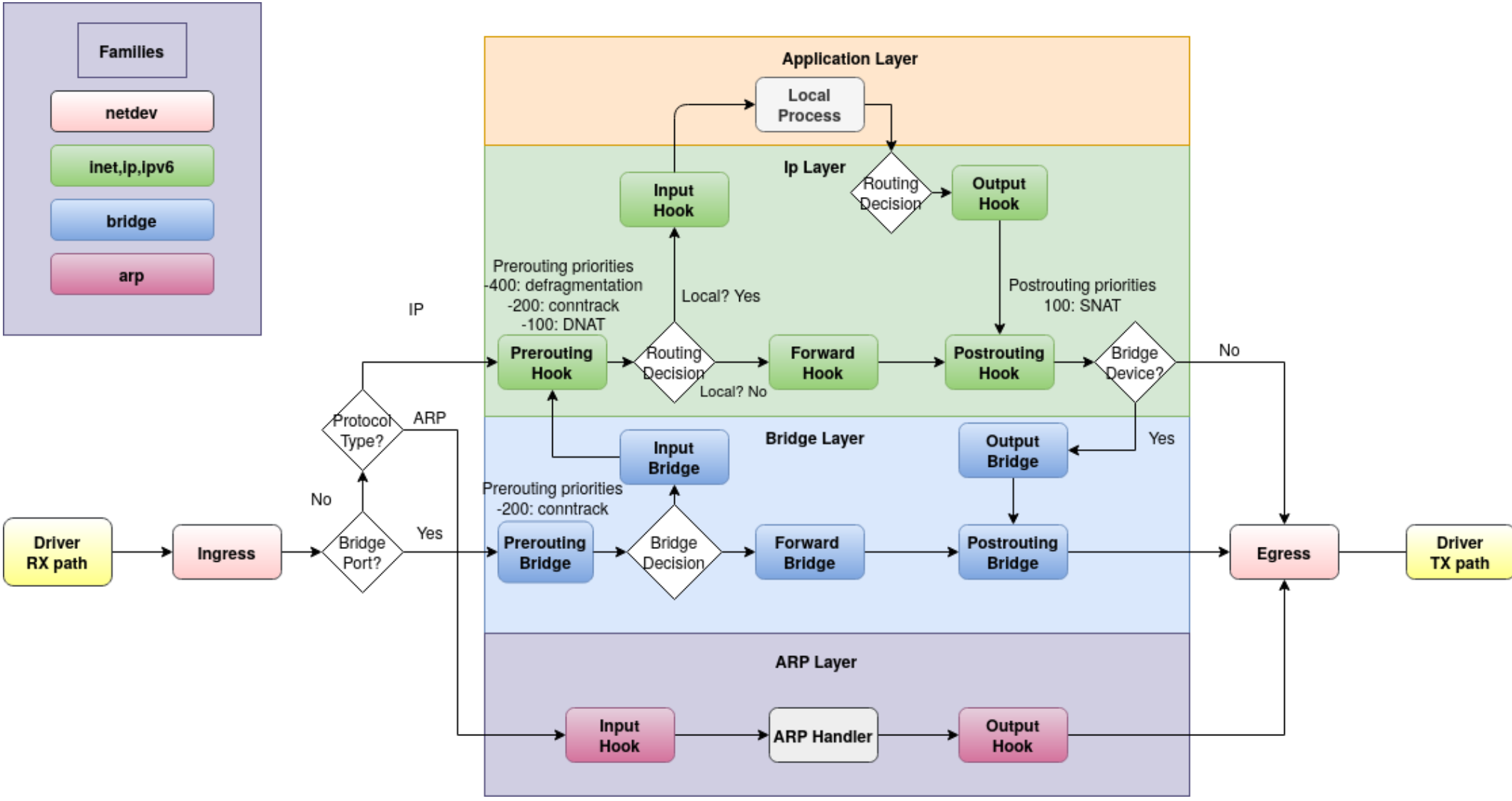
- Same team of developers
  - Same kernel module
  - Same terminology
- New syntax
  - More flexible rules
- No default chains
  - Improves performance and scalability
- Comprehensive
  - Includes ARP, IPv6, ethernet

`iptables` terminology

- `table` container for chains
  - Predefined names `filter`, `nat`, `mangle`, `raw`
- `chain` list of rules executed in sequence
  - `filter` table has `INPUT`, `OUTPUT` and `FORWARD`
  - `nat` table has `PREROUTING, POSTROUTING` and `OUTPUT`
  - Default action is called policy
- `rule` defined action
  - Actions are `ACCEPT`, `DROP`, `QUEUE` or `RETURN`

`nftables` terminology

- `table` container for chains
- `chain` list of rules
  - `type` filter, route or nat
  - `hook` where this chain fits
  - `priority` used to order chains
  - `policy` default action
- `rule` defined action


Families
netdev
inet,ip,ipv6
bridge
arp
Application Layer
Local Process
Ip Layer
Routing Decision
Output Hook
Input Hook
Prerouting priorities
-400: defragmentation
-200: conntrack
-100: DNAT
Local? Yes
Postrouting priorities
100: SNAT
IP
Prerouting Hook
Routing Decision
Local? No
Forward Hook
Postrouting Hook
Bridge Device?
No
Yes
Bridge Layer
Input Bridge
Output Bridge
Protocol Type?
ARP
No
Prerouting priorities
-200: conntrack
Driver RX path
Ingress
Bridge Port?
Yes
Prerouting Bridge
Bridge Decision
Forward Bridge
Postrouting Bridge
Egress
Driver TX path
ARP Layer
Input Hook
ARP Handler
Output Hook

Example: local access to ssh server

- Accept only TCP/IP connections on port 22 from LAN and DMZ
- Allow any outgoing connections
- iptables
  -
```
iptables -A INPUT -p tcp --dport 22 -s 100.64.42.0/24 -j ACCEPT
iptables -A INPUT -p tcp --dport 22 -s 10.21.32.0/24  -j ACCEPT
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
```
- nftables

```
nft -i
table ip saclass {
  chain incoming {
    type filter hook input priority 0; policy drop;
    ip saddr {10.21.32.0/24,172.30.40.0/24} tcp dport 22 accept
  }
}
```

-

Understanding iptables/nftables

- Packets mapped to table&chains based on source
  - filter input, output, forward, etc
- Rules in chain are executed in order
  - check for match of address, port, etc
    - action taken if match
  - rest of rules are not evaluated
    - except if the action was `jump`
- You have to allow replies to establish a connection

tables

- The table select the packet family
- `iptables` predefines `filter`, `nat`, `mangle`, `raw`
- `nftables` user defined `table` *`family`* `name`
  - `ip` IPv4
  - `ip6` IPv6
  - `inet` Internet (IPv4 or IPv6)
  - `arp` IPv4 ARP
  - `bridge` bridge (ethernet)
  - Example: `nft add table ip saclass`

base chains

- `iptables` predefined `filter` chains `INPUT`, `OUTPUT` and `FORWARD`
  - Policy set by `iptables -P chain action`
    Example: `iptables -P INPUT DROP`
- `nftables` user defined `table chain name`
  - `type` chain type `filter`, `nat`, `route`
  - `hook` where it fits `input`, `forward`, `output`, `prerouting`, `postrouting`
  - `priority` order of chains (low to high) numeric or
  - `policy` default action `accept` or `drop`
  - Example: Incoming packet filter drop except what is allowed

```
nft add chain saclass incoming
   '{type filter hook input priority 0; policy drop;}'
```

rules

- Rules are executed in order
- Execution stops when a rule matches and action is taken
  - iptables actions are `ACCEPT`, `DROP`, `REJECT`
  - nftables actions are `accept`, `drop`, `reject`
- Example: Allow incoming to port 22 (ssh)
  - iptables
    `iptables -A INPUT` **`-p tcp --dport 22 -j ACCEPT`**
  - nftables
    `nft add rule saclass incoming` **`tcp dport 22 accept`**
- All the conditions in the rule must match

iptables

- `-F` flush (clear)
- `-A` append rule
- `-I` insert rule
- `-D` delete rule
- `-N` new chain
- `-P` set policy

- Set rules on boot
  - Redhat
    - `/etc/sysconfig/iptables`
  - Debian
    - `/etc/iptables/rules.v4`
- Show rules
  - `iptables -nvL`

Example /etc/sysconfig/iptables

```
#iptables for dundermifflin web server
*filter
# Default drop INPUT and FORWard, ACCEPT output
:INPUT   DROP   [0:0]
:FORWARD DROP   [0:0]
:OUTPUT  ACCEPT [0:0]
# Allow established OR RELATED connections
-A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
# Allow selected icmp traffic
-A INPUT -p icmp --icmp-type echo-request            -j ACCEPT
-A INPUT -p icmp --icmp-type echo-reply              -j ACCEPT
-A INPUT -p icmp --icmp-type time-exceeded           -j ACCEPT
-A INPUT -p icmp --icmp-type destination-unreachable -j ACCEPT
-A INPUT -p icmp                                     -j DROP
# Allow any inbound http
-A INPUT -p tcp --dport 80  -j ACCEPT
# Permit selected inbound ssh
-A INPUT -p tcp -s 198.18.0.0/16 --dport 22 -j ACCEPT
-A INPUT -p tcp -s 100.64.0.0/16 --dport 22 -j ACCEPT
-A INPUT -p tcp -s 10.21.32.0/24 --dport 22 -j ACCEPT
# Deny outbound to facebook
-A OUTPUT -d 157.240.28.35 -j DROP
COMMIT
```

`nft` (nftables)

- `list` show content
  - `ruleset`
  - `table`
  - `chain`
- `add` item
- `delete` item
- `flush` empty

- Set rules on boot
  - Redhat
    - `/etc/sysconfig/nftables.conf`
  - Debian
    - `/etc/iptabbles/rules.v4`
- Show all rules
  - `nft list ruleset`

Example /etc/sysconfig/nftables.conf

```
#!/usr/sbin/nft -f

#  Flush all previous rules
flush ruleset

#  IPv4 input chain - allow ssh from LAN and VPN only
#  Block everything else
table ip saclass {
  chain incoming {
    type filter hook input priority 0; policy drop;
    ip saddr {10.21.32.2/24,198.18.0.0/16} tcp dport 22 accept
  }
}
#  No output chain which allows everything
```

Connection Tracking `ct`

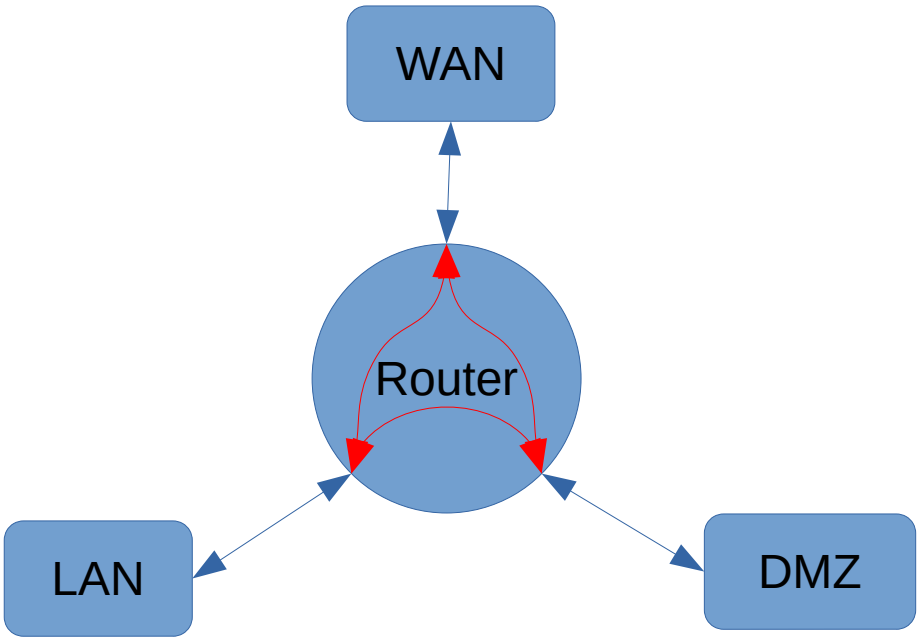
- Statefull firewall
- Allows replies to permitted connections
- Critical for complex protocols like `ftp`
- Typical usage
  - `ct state established,related accept`
  - `ct state invalid drop`

Machine A Firewall

- Inbound connections
  - ssh from LAN, DMZ, WAN and VPN
  - DHCP and NTP from LAN and DMZ
  - returns from outbound connections
- Outbound connections
  - Any except facebook

Machine A Forwarding

- Three interfaces
  - LAN, DMZ, WAN
- Six paths
  - WAN to LAN
  - WAN to DMZ
  - DMZ to WAN
  - DMZ to LAN
  - LAN to WAN
  - LAN to DMZ



Machine A Forward Chains

- WAN to DMZ
  - ICMP
  - UDP DNS
  - TCP ssh, http, grader
  - return packets
- WAN to LAN
  - return packets
- LAN to WAN
  - anything except facebook
- LAN to DMZ
  - anything
- DMZ to WAN
  - ICMP
  - UDP DNS
  - TCP DNS, http, https
  - return packets
- DMZ to LAN
  - ICMP
  - TCP ssh,NFS
  - return packets