HPS
https://hps.vi4io.org

Julian Kunkel

Firewalls

Learning Objectives

- Describe main responsibilities of a firewall
- Utilize nftables in Linux to set up basic rules based on a template
- Utilizing tools to test the effectiveness of the firewall

Outline

1 Introduction

2 Firewalls in Linux

3 Summary

Motivation

- System security is vital
- Admins want to restrict access to desired/expected services
  - Maybe only a subset of clients (IPs) shall be able to access
  - Prevent accidental exposure of services to the world
- Admins want to limit rate of network or be notified
- Admins want to block malware and application-layer attacks
- In some scenarios, want to redirect network traffic
  - NAT = Network Address Translation rewrites network addresses/ports

$\Rightarrow$ Firewalls do this for us!

General Architecture

Firewall

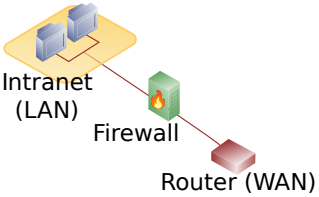

Figure: Source: Wikipedia [3] (heavily modified)

- Network packets pass through the firewall
- A firewall can be local to the computer system
- Packets can be accepted, rejected, forwarded
- Packets can be modified and redirected...

DMZ = Demilitarized Zone

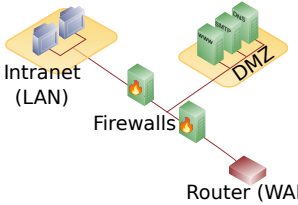

Figure: Source: Wikipedia [3] (modified)

- Typically employs two firewalls
- Exposes externally facing services to untrusted network
- Protects the local network by isolating Internet and private network

Differentiating Firewalls

Types of Firewalls [1]

- **Packet filtering**

  A small amount of data is analyzed and distributed according to the filter's standards.

- **Proxy service**

  Network security system that protects while filtering messages at the application layer.

- **Stateful inspection**

  Dynamic packet filtering monitors active connections to determine routing.

- **Next Generation Firewall/Deep Packet Inspection**

  Deep packet inspection Firewall with application-level inspection.

Visibility

- **Visible -** firewall is between client/target system - client must be configured to use firewall
- **Transparent -** communication through firewall is ensured via network configuration

Outline

1 Introduction

2 Firewalls in Linux

3 Summary

Interaction of Netfilter components in Linux

- There exist various user-space tools that allow to modify network packets on different levels

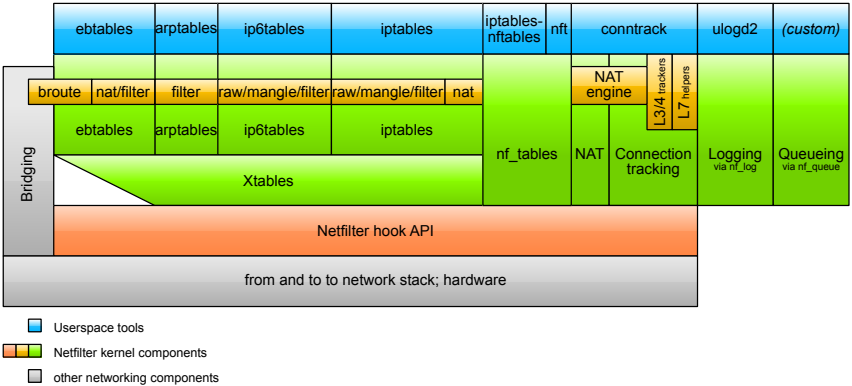

Figure: Source: Jan Engelhardt, Netfilter-components.svg, Wikipedia [4]

nftables [5]

- VM executing byte-code to inspect a network packet
- Make decisions on how that packet is handled
  - Based upon data from packet, associated metadata (e.g. interface), and connection-tracking
  - May use arithmetic, bitwise and comparison operators
- May manipulate sets of data (typically, IP/Port)
- The netfilter tool (nft) can be used to manipulate them
  - Example: `$ nft add rule inet filter output ip daddr 1.2.3.4 drop`
  - May log our count packets for which a rule applies

Organisation of rules

- Rules are uniquely identified by a table, a chain and the specification
- Rules can also be uniquely identified via a handle
- A table belongs to one network family (ip, ip6, inet (ip+ip6), arp, bridge)
- A chain can be linked to a network hook (interfaces with traffic)

Chains

- Filter hook: INPUT (local tgt), OUTPUT (local send), FORWARD (routing)
- NAT hook: used to mangle packets (before or after routing)
- Root can create custom chains for better management

Netfilter Packet Flow

- Processing of a packet is complex, there are many paths and chains

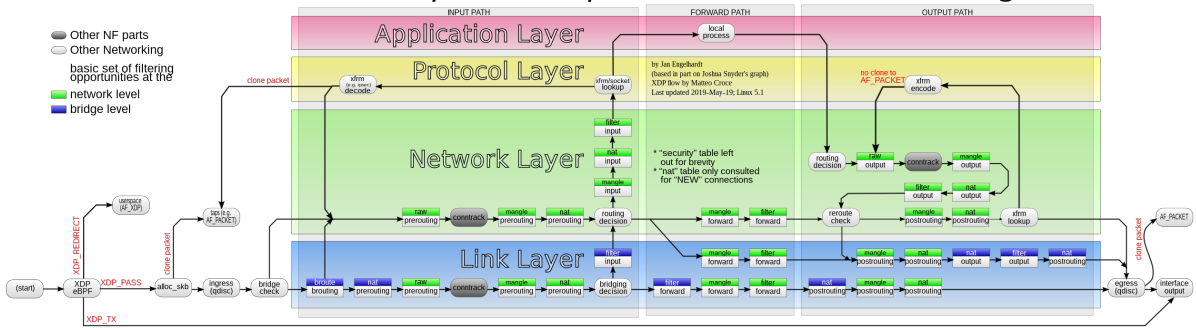

Figure: Source: Jan Engelhardt, Netfilter-packet-flow.svg, Wikipedia [4]

Basic NFT Commands [6,7]

- Show current firewall rules

```
$ nft list tables       % available tables
$ nft -a list ruleset % all rules, -a shows the handles
```

- Removing all rules – flush rules, beware of loosing connection to ssh!

```
$ nft flush ruleset
```

- Load rules from a file (-c check only the validity, then remove -c)

```
$ nft -c -f /etc/nftables.conf
```

- nft can be used to manipulate/add/remove individual rules
  Example: Drop packets with destination IP address 1.2.3.4

```
$ nft add rule inet filter output ip daddr 1.2.3.4 drop
```

- To ensure persistency, I advise using `$ nft -f`

Stateful inspection via connection tracking [4]

- Goal: Keep track of logical connections - i.e., across multiple packets
  - Useful for higher-level protocols such as FTP, TCP and even UDP
- NEW: trying to create a new connection
- ESTABLISHED: part of an already-existing connection
- RELATED: new connection that has been expected (e.g., for FTP)
- INVALID: invalid state, e.g., not valid according to TCP state diagram
- UNTRACKED: used by admin to bypass connection tracking

Example Session

- We must test that a firewall works as intended... Let's try this:

```
# Retrieve data from gwdg.de webserver
$ wget 134.76.9.48
# Block any outgoing IP to host with the IP of gwdg.de
$ nft add rule inet filter output ip daddr 134.76.9.48 drop
# This should not work:
$ wget 134.76.9.48
# List the current rules, with -a we get a handle
$ nft -a list ruleset
# Remove the rule again, in our case the rule handle is 4
$ nft delete rule inet filter output handle 4
```

Example Server Configuration (based upon [8])

```
flush ruleset # remove all existing rules
table inet firewall {
  chain input_ipv4 {
    # Accept ping (icmp-echo-request) for diagnostic purposes. Allows discover if host is alive. Accept with rate limit
    icmp type echo-request limit rate 5/minute burst 20 packets counter accept
  }
  chain input_ipv6 {
    # accept neighbour discovery otherwise connectivity breaks count the number of hits to this rule
    icmpv6 type { nd-neighbor-solicit, nd-router-advert, nd-neighbor-advert } counter accept
    # accepting ping (icmpv6-echo-request) for diagnostic purposes.
    icmpv6 type echo-request limit rate 5/second accept
  }
  chain input {
    # By default, drop all traffic unless it meets a filter criteria specified by the rules that follow below.
    type filter hook input priority 0; policy drop;
    # Allow traffic from established and related packets, drop invalid, keep this!
    ct state vmap { established : accept, related : accept, invalid : drop }
    # Only for new connections
    iifname lo counter accept   # Allow loopback traffic use name for counter
    # Jump to chain according to layer 3 protocol using a verdict map
    meta protocol vmap { ip : jump input_ipv4, ip6 : jump input_ipv6 }
    tcp dport { 22, 80, 443} counter accept  # Allow SSH on port TCP/22 and allow HTTP(S) TCP/80 and TCP/443
    limit rate over 10/minute counter drop   # Drop packets with rate > 10/minute, needed to limit logging rate
    log prefix "[nftables] input Denied: " counter drop   # Enable logging of remaining input traffic
  }
  chain forward {
    type filter hook forward priority 0; policy drop; # Drop everything (assumes this device is not a router)
  } # no need to define output chain, default policy is accept if undefined.
```

Sidequest: Scanning ports using nmap

- More testing of the firewall
- With nmap, we can scan open ports
  `nmap -A localhost`
- Note: A scan is often an indicator for an upcoming attack
  - Do only scan a host/network if this is agreed by the owner!
- To create a service for testing, you can service from a TCP port, e.g. using
  `$ nc -l PORT`
- Logfiles: Use `$ cat /var/log/syslog` to show output created from nftables

Summary

- The Netfilter hook API allows implementing firewalls on all levels
- Connection allows tracking the logical connection state
- List the rules via: `$ nft -a list ruleset`
- Recommendation:
  - Use a file to store and work on the rules
  - When working on a remote server, have a backup rule to login
    Save rules only after testing them to prevent lock-out or connect via IPMI
- Exercise: We utilize the Linux firewall!

References

1 https://www.checkpoint.com/cyber-hub/network-security/what-is-firewall/

2 https://en.wikipedia.org/wiki/Firewall_(computing)

3 https://en.wikipedia.org/wiki/DMZ_(computing)

4 https://en.wikipedia.org/wiki/Netfilter

5 https://lwn.net/Articles/564095/

6 https://wiki.nftables.org/wiki-nftables/index.php/What_is_nftables%3F

7 https://wiki.nftables.org/wiki-nftables/index.php/Quick_reference-nftables_in_10_minutes

8 https://wiki.nftables.org/wiki-nftables/index.php/Simple_ruleset_for_a_server

9 https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks