

# TCP/IP Firewalling Basics

by

Harald Welte <laforge@sunbeam.franken.de>

# Contents

---

- Introduction
- Networking Basics
- Potential Security Problems
- Solution 1: Packet Filters
- Solution 2: Proxies
- Comparison
- Summary

Firewalling Basics

# Introduction

# Networking Basics

---

- 7 layer OSI model used to abstract networking protocols
  - layer 7: application layer: e.g. telnet/ftp
  - layer 6: presentation layer:
  - layer 5: session layer:
  - layer 4: transport layer: e.g. TCP/UDP
  - layer 3: network layer: e.g. IP
  - layer 2: data link layer: e.g. Ethernet
  - layer 1: physical layer: e.g. Wire
- Layer 1 + 2 embedded in hardware
- Layer 3 + 4 implemented in operating system
- Layer 5+ embedded in application program

# Networking Basics

---

- Layer 2: Ethernet
  - enables two hosts within same physical net to exchange packets
  - unreliable
  - addressing granularity: host
  - fixed hardware addresses (MAC address, 48bit)
  
- Layer 3: Internet Protocol (IP)
  - enables two hosts in different physical networks to exchange packets
  - unreliable, best effort
    - ▷ packet reordering
    - ▷ packet loss
  - addressing granularity: host
  - logical addresses (IP Address, 32bit)
  - checksum protects only IP header

# Networking Basics

---

- Layer 4: User Datagram Protocol (UDP)
  - unreliable, best effort
  - addressing granularity: ports (16bit = 65535)
  - optional payload checksum
  
- Layer 4: Transmission Control Protocol (TCP)
  - provides connection abstraction
  - reliable
    - ▷ ordering guarantee
    - ▷ retransmissions correct packet loss
    - ▷ flow control
    - ▷ payload checksum protects payload from data corruption
  
- Layer 4: Internet Control Message Protocol (ICMP)
  - used internally by TCP/IP protocol suite
    - ▷ error messages (e.g. host unreachable)
    - ▷ diagnostics (e.g. ping/pong)

# Potential Security Problems

---

- Security issues arise at interconnection of two networks
  - Traditional Case: IP Router connecting an organization internal network to the Internet
  
- What Security Problem?
  - organization-internal services exposed to outside network
  - spoofed (forged) packets to circumvent 'security by address'
  - even if all internal services secured by authentication, difficult to guarantee security on all internal hosts
  
- Why Firewalling?
  - to restrict which internal services are exposed to the outside
  - to restrict which outside services are used by internal users

# Solution 1: Packet Filters

- Filter individual packets at network interconnection (Router)
  
- Filter criteria traditionally include
  - IP source + destination address
  - TCP/UDP source + destination port
  - TCP header flags
  
- Filtering rules determine if
  - packet is allowed to transit interconnection
  - packet is silently dropped
  - packet is dropped and error message returned to sender

# Solution 1: Packet Filters

## □ Capabilities

- disallow communication between certain IP addresses
- disallow communication between certain port numbers
- disallow malicious packets, like packets
  - ▷ using source routing IP option
  - ▷ impossible combination of features, like tcp xmas scan
- generate log of malicious and/or filtered packets

## □ Limitations

- scope limited to individual packets
- no ability to look inside packet payload (HTTP 1.1 virtual hosts)
- no abstraction of connection, filtering rules needed for both directions

# Solution 1: Packet Filters

- Extensions
  - stateful packet filters (connection tracking)
    - ▷ filtering only needed for connection-initiating packets
    - ▷ all other packets within connection are accepted as part of an already established connection
  - TCP window tracking
    - ▷ allow filtering not only on source/dest port but also on TCP sequence number
  - NAT (Network Address Translation)
    - ▷ manipulation of source / destination address
    - ▷ redirect packets to other hosts
    - ▷ 'share' one ip address at dialup accounts (masquerading)
    - ▷ connect two networks with overlapping address ranges

## Solution 2: Proxies

---

- A proxy operates at layer 5 and above
  
- Mode of operation
  - client connects to proxy instead of server
  - proxy initiates a second, separate connection to server
  
- Proxies are just normal programs implementing a server and a client for a particular application protocol (e.g. HTTP) using operating system mechanisms (like sockets API, winsock, ...)

# Solution 2: Proxies

## □ Capabilities

- disallow communication between certain IP addresses
- disallow communication between certain ports
- disallow communication based on packet payload
  - ▷ e.g. pathnames / filenames within HTTP and FTP
  - ▷ e.g. email-addresses within SMTP
  - ▷ e.g. hostnames within DNS ([www.netzensur.de](http://www.netzensur.de))
  - ▷ e.g. badwords ('sex' and 'teen' within same file)
- manipulation of packet payload
  - ▷ everything possible...

## □ Limitations

- somebody needs to tell client app to connect to proxy instead of server
- separate proxies for all used protocols needed
- not possible to filter on packet options, etc.

# Solution 2: Proxies

- Extensions
  - Transparent Proxies
    - ▷ accept connections from client independent of dest IP
    - ▷ make reply packets to the client look like as sent by server
    - ▷ possibly to implement same transparency towards server
    - ▷ no need to tell clients about proxies anymore!
  - SOCKS
    - ▷ application protocol independent proxy
    - ▷ one proxy for all application protocols
    - ▷ uses separate protocol between client and proxy
    - ▷ needs explicit support from client application
    - ▷ integrated username/password authentication

# Comparison

---

## Packet Filter

- pro
  - ▷ total control on lowest per-packet level
  - ▷ very high performance
  - ▷ possible to implement failover / load balancing
  - ▷ NAT as extension solves address space problem
- contra
  - ▷ configuration requires sophisticated knowledge
  - ▷ problems when no state / window tracking used
  - ▷ support for complex protocols (H.323, SIP) difficult to implement

## Proxy

- pro
  - ▷ no knowledge about layer3/4 protocol needed
  - ▷ configuration very easy
  - ▷ address space automatically separated
  - ▷ integrates easily with other applications like IDS
  - ▷ easy implementation, just normal application programs
- contra
  - ▷ separate proxies needed for almost every protocol
  - ▷ bad performance
  - ▷ uses lots of resources (e.g. sockets) on gateway
  - ▷ horribly breaks end-to-end
  - ▷ needs explicit configuration of client apps if not transparent proxy

# Comparison

---

- **Transparent Proxy**
  - uses ideas/methods of packet filtering (NAT) to achieve protocol transparency
  - horrible violation of layering
  
- **Stateful Packet Filter**
  - use ideas of proxies (tracking of higher layer state) to achieve better security and easier configuration
  - horrible violation of layering

# Conclusion

- Conclusion
  - proxies work for small installations where number of used protocols is small and administrative staff not very experienced
  - packet filters without state tracking are difficult to configure correctly
  - packet filters with state tracking are good solution for most usage scenarios: powerful but yet easy to configure correctly
  - for highest security, best of both worlds can be combined
    - ▷ imagine a stateful bridging packet filter in front of a proxy :)

# Thanks

---

- Thanks to
  - the BBS people, Z-Netz, FIDO, ...
    - ▷ for heavily increasing my computer usage in 1992
  - KNF
    - ▷ for bringing me in touch with the internet as early as 1995
    - ▷ for providing a playground for technical people
    - ▷ for telling me about the existence of Linux!
  - Alan Cox, Alexey Kuznetsov, David Miller, Andi Kleen
    - ▷ for implementing (one of?) the world's best TCP/IP stacks
  - Paul 'Rusty' Russell
    - ▷ for starting the netfilter/iptables project
    - ▷ for trusting me to maintain it today
  - Linux User Group Nuernberg (ALIGN, LUG-N)
    - ▷ for helping me with my initial Linux problems