OpenBSD

firewalls: back, now, then

Bagatelle - Mauritius, September 10, 2017

Philipp Bühler <pbuehler@sysfive.com>

Trivia:

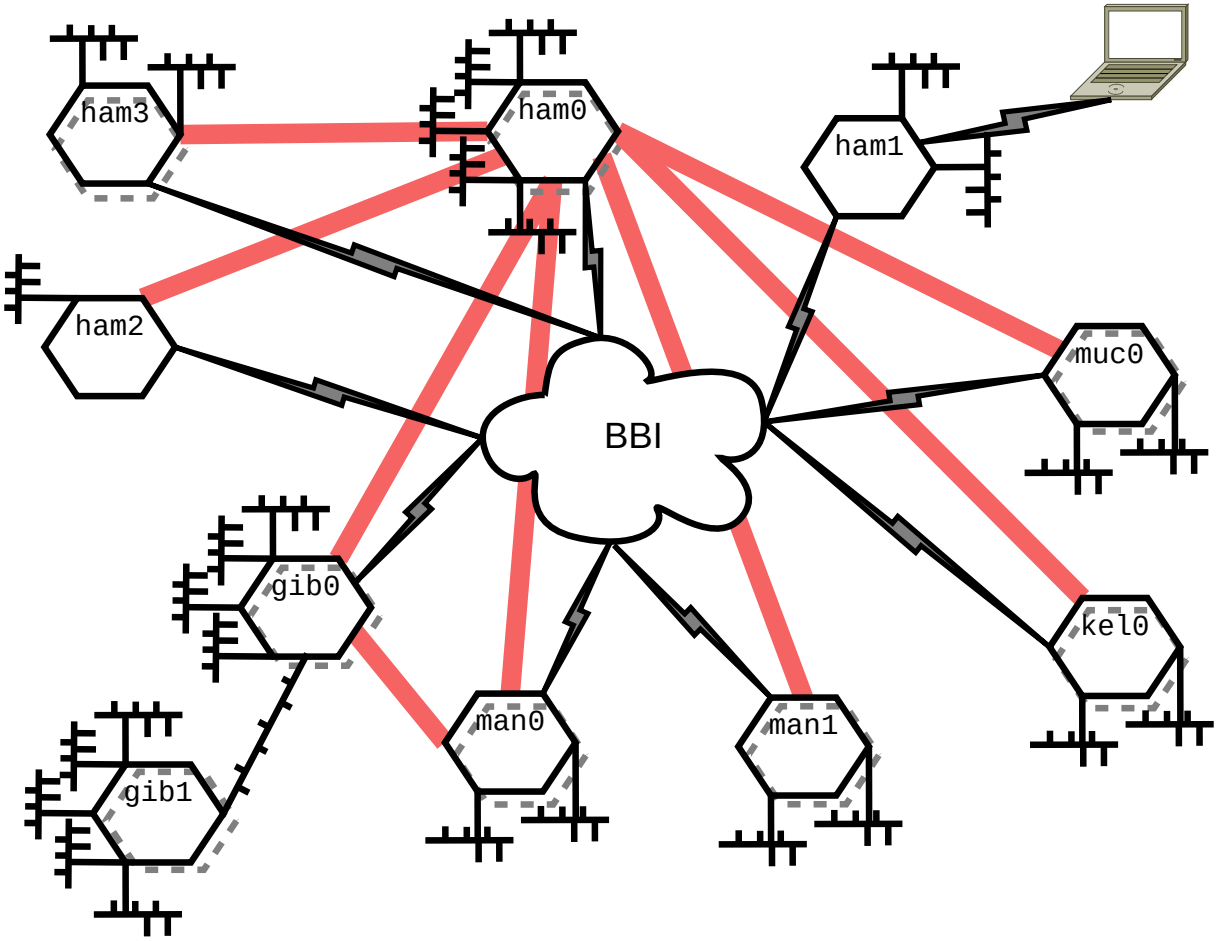
Technical lead at sysfive.com GmbH
Hacking computers since 1983
OpenBSD user since 2.7 (2000)
Developer (pf) 2002-2005

12 people at two locations
Apple][e
on i386, amd64, sparc64, macppc
slacked too much

Contents

- My Company
- History/Roots
- Categories
    - Bridge (L2)
    - packetfilter (L3+L4)
    - "application" (L5-L7)
- Problems to solve
    - Networkstack shortcomings
    - uncontrollable services
    - Address Management
    - Crypto
    - Queueing
- Implementations
    - Layer 2: arp settings, netfilter (br_module), VLAN
    - Layer 3+4: pf, npf, ipfw, ipfilter, netfilter/iptables
    - Applayer: relayd, asorted proxies, "deep packet inspection"

sysfive.com network



my dia(1) skills are sub par, giving up..

On the road to today..

- TCP/IP: 1983 (ARPANET "flag day", Jan 1st), router segregation
- Gatekeeper: 1987 (predecessor of SEAL), rather a service proxy than a "filter"
- first filter: 1989 as a stateless packet filter (DEC -> SEAL, AltaVista FW) - "inspired" by the Morris Worm (likely)
- statefulness: 1990 AT&T - prone to DoS attacks
- pb enters the networks: 1992
- App-Layer: 1994 (FWTK, Gauntlet)
- "NGFW": IDS, IPS, WAF, User-Management

Further reading: http://www.cs.unm.edu/~treport/tr/02-12/firewall.pdf
from 2002 but a treasure trove of other references to also be read to understand more of the history, shortcomings back then (which still might be around...)

frags on the wall / flags out of order

Fragmentation usually happens due to different MTUs while packets travel across the network. Yet, this "feature" can also be used to construct a flow of packets trying to undermine filtering.
Firewalls can "collect" the fragemented packets and reassemble them to a digestable packet flow before sending them to the next host (or router).

Further reading: https://en.wikipedia.org/wiki/IP_fragmentation_attack

Of course any generic network stack bug in implementations run "behind" the firewall might be protected ("ping of death" and others) - but what if the stack is vulnerable on a "generic" level and needs to serve something?

Some attacks try to undermine the state flow of TCP connections to pass packets through (rather DoS by bandwidth) or try to crash the remote state engine (kermit? XXX)

Services out of control

Typically a network administrator has no direct control over which services could possibly started and exposed on the network. To limit the exposure a firewall is usually in place to block access from the outside.
Less often access from the inner network to the public (or other controlled segments) faces restrictions. Rethink this along examples like:

- softether and thelike ("VPN over everything")
- corkscrew and thelike ("ssh tainted as TLS")
- ssh -R (works almost everywhere, very difficult feature to block!)

Which all basically "tunnel" the firewall and potentially allowing access from the outside to inner perimeter which would be usually blocked.

ALWAYS cross-check ("audit") your configuration/policy against what your INTENTION was. Slipping (e.g. "just" wrong NAT) is easily introduced (and might go unnoticed for too long) - but also catastrophic havoc is at YOUR fingertips any minute. Nothing wrong with self-confidence, but review/audit is good practice, esp. when doing large or "exotic" things.

And do not filter ICMP blindly (PMTUD!) - a source of "legit" frags or would block you from ’DF’ packets.

Address Management / Crypto / Queues

- NAT - defacto standard today; not really adding ’security’
- NAT - "loadbalance" outgoing connections (countermeasure to "clever" rate limits)
- NAT - could be the only way to overcome IP prefix collisions (VPN..)
- VRF - rdomains; virtualize routers (firewalls), handle overlapping IP setups
- CARP/VRRP - high available addresses
- VPN - openvpn
- VPN - IPsec
- Queue - restrict usable bandwidth per service/user/...

a word on IPS

- snort, hogwash, suricata, ...
- latent to false positive, actively hindering legit traffic
- pattern management is painful (who audits >30k patterns at yours?)
- can add severe latency
- how about putting it back to IDS -- aside of the hot path (Monitorport)

Layer 2 - ARP/MAC / VLAN

Basics

- static arp entries (arp -s)
- netfilter "bridge module"
- who is doing this anyway?

VLAN

- just 12 bits in the ethernet header
- useful for logical separation
- be cautious about your switch config (and physical access to ports / WiFi)

Layer 3+4 (and above) - Implementations

Layer 3 is typically IPv4 or IPv6 (still IPX, AppleTalk, .. anyone?). Layer 4 carries "protocols" like TCP, UDP, ICMP and their parameters.
The following systems carry different implementations - with OpenBSD's pf(4) spreading out :-).

- OpenBSD: pf
- FreeBSD: pf, ipfw, ipfilter
- NetBSD: npf, pf, ipfilter
- Linux: netfilter (iptables), firewalld, ipchains
- Solaris: ipfilter, pf
- Mac OSX: pf

Implementation Overview

By design, pf(4) wont ever look at anything below or above Layer 3 and 4. For inspection and support of upper layers see relayd(8), ftp-proxy(8) and tftp-proxy(4).

As a contrast, netfilter will try to look "deep" into packets to gather application information it needs to support e.g. ftp. Ever thought that parsing arbitrary, unlimited data in the kernel is a good idea?

pf, npf and ipfilter use syntax based configuration file with a parser ensuring validity. netfilter has only "helpers" and depending on the Linux-Distribution that can differ dramatically:
CentOS6: /etc/sysconfig/iptables,
Debian/Ubuntu: /etc/iptables-save XXX?, Arch, ...
CentOS7: 6 + firewalld

Implementation: OpenBSD pf(4)

- First implementation 2001 with OpenBSD 3.0
- Overcoming license issues with ipfilter
- Full stateful support of IPv4 + IPv6
- /etc/protocols supported: UDP, TCP, ICMP/ICMP6, ESP/AH, many more
- Syntax based configuration file pf.conf(5)
- Rulesets are nestable (anchors) and loaded atomically ("table swap")
- handling fragmentation to pass on "sane" packets
- builtin ruleset optimizer
- packet queueing and detailed statistics builtin
- Interaction with other programs like relayd via /dev/pf socket
- pf states can be synced to other hosts
- authpf for user-authenticated (ssh) rulesets (awesome!)
- NOT YET - almost holding breath - MP capable (unlock of whole network stack including pf any minute)

Further reading. https://www.openbsd.org/faq/pf/index.html

Implementation: NetBSD npf(4)

- MP support
- modularized
- builtin "ALG" (application level gateway) to handle upper layer protocols
- IPv6 yet?

Further reading: http://www.feyrer.de/NetBSD/bx/blosxom.cgi/nb_20100914_0805.html
http://www.netbsd.org/ xxx

Implementation: FreeBSD ipfw(4)

I have no idea... it can do some very funky stuff, but never looked at it.
Is FLC still alive? Been used to create ipfw (and others) rulesets

Implementation: Linux netfilter/iptables

- 3rd (or so) rewrite/reimplementation of a packet filter on steroids
- modularized (mainly for different layer options)
- operating on "chains" (historical it was named ’ipchains’ in first place)
- configuration only by commandline options - every platform(family) having built an eco-system around this for "easy management" (yup).
- addresses all layers with possible information "flow" between the layers -e.g. ftp command channel ’inspection’ to open up needed data ports automatically (keyword: "related")
- ip6tables, iptables6(?), /* no comment */

Word of caution: be very aware of what you’re doing. netfilter can outsmart you in nanoseconds!

Virtualization

- on which level? Stack - VM - ALG/proxy?
- rdomains! there’s a great talk out there ;-)
- the "host" has to have filtering, too - d’uh.
- VM performance (SRVIO or bust..)

What's coming next

- IPv6: everybody (esp. commercial) still sleeping for anything "above basic filtering"
- IPv6: likely that there are still a PLETHORA of bugs in these implementations (frags, NAT64, ..)
- TLSv1.3 and beyond
- But, in the unlikely event ;), that one is designing a new protocol: think about firewalls, do not use: dynamic ports like FTP, JMX, H.323.. horrible, horrible, BAD!
  Do not implement "backchannels"/"callbacks" (read: do not try to open new TCP/IP connections from server side - have very good reasoning inventing something more complicated than HTTP or so on the network level)
- More likely: Writing a new proxy or network daemon for whatever. Be VERY cautious, take a look at GOOD examples on how to minimize attack surface or containing information passing (relayd, bgpd, postfix, qmail, haproxy, nginx). Examples of horror: ISC-BIND (even today), Apache HTTPD, Squid, ipchains/early iptables (really), <long list>
  Throw any analyzer that you can get hold off on it (valgrind, fuzzers, ...) - somebody else WILL do it, be the first.
  Let "the community" have a look, do not hide - esp. not by "obfuscating" code ("magic numbers") and so on.

- Distributed, automated configuration management (several approaches out there, including fails)
- Personally busy enough with TODAY’s problems - no time to dream up future problemsˆWsolutions :-)
- Who knows what - ideas from you?

K-Thx-Bye

- Further questions?
- Thanks again, for having me here.
- No questions?
- Sorry, could not bring GroffTheBSDGoat - maybe next time when I am the keeper again.
- Food + Beer now?