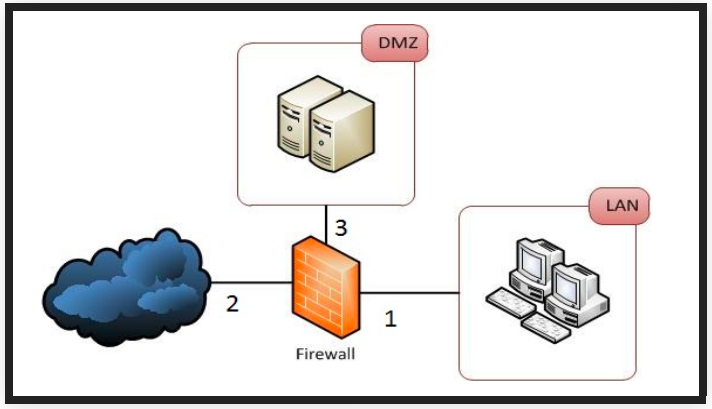
Po co?

- filtrowanie, blokowanie ruchu
- zapobieganie skanowaniu, atakom



https://shinsec.pl/dmz/

ipfwadm

- wersja kernela 2.1 i niżej
- prosty mechanizm filtorwania pakietów:
  - reguły *input*, *output* itp.
  - protokoły TCP, UDP, ICMP
  - strategie *accept*, *deny*, *reject*
- nieelastyczne, mało wygodne przy wielu regułach

ipfwadm

- przykład - pozwalamy jedynie na dostęp do WWW

```
ipfwadm -F -f # flush
ipfwadm -F -p deny
ipfwadm -F -a accept -P tcp -S 172.16.1.0/24 -D 0/0 80
ipfwadm -F -a accept -P tcp -S 0/0 80 -D 172.16.1.0/24

ipfwadm -F -l # list forward rules
```

ipchains

- od wersji kernela 2.2 (1999)
- wiele usprawnień:
  - tworzenie własnych łańcuchów (*chains*)
  - wsparcie dowolnego protokołu IP
  - nowe opcje, np. *replace*

ipchains

- przykład - własny łańcuch:

```
ipchains -P input DENY
ipchains -N tcpin
ipchains -A tcpin -s ! 172.16.0.0/16
ipchains -A tcpin -p tcp -d 172.16.0.0/16 ssh -j ACCEPT
ipchains -A tcpin -p tcp -d 172.16.0.0/16 www -j ACCEPT
ipchains -A input -p tcp -j tcpin
ipchains -A input -p all
```

netfilter i iptables

netfilter

- framework w jądrze Linuxa ($\geq$ 2.4) pozwalający na filtrowanie pakietów, zarządzanie NAT…
- umożliwia rejestrację *callback'ów*, przetwarzających pakiety
- **iptables** - narzędzie do zarządzania *netfilter*

iptables

- od wersji kernela 2.4 (2001)
- wsparcie dla IPv6
- śledzenie stanu protokołu
- nowe łańcuchy wbudowane - PREROUTING i POSTROUTING

netfilter/iptables chains

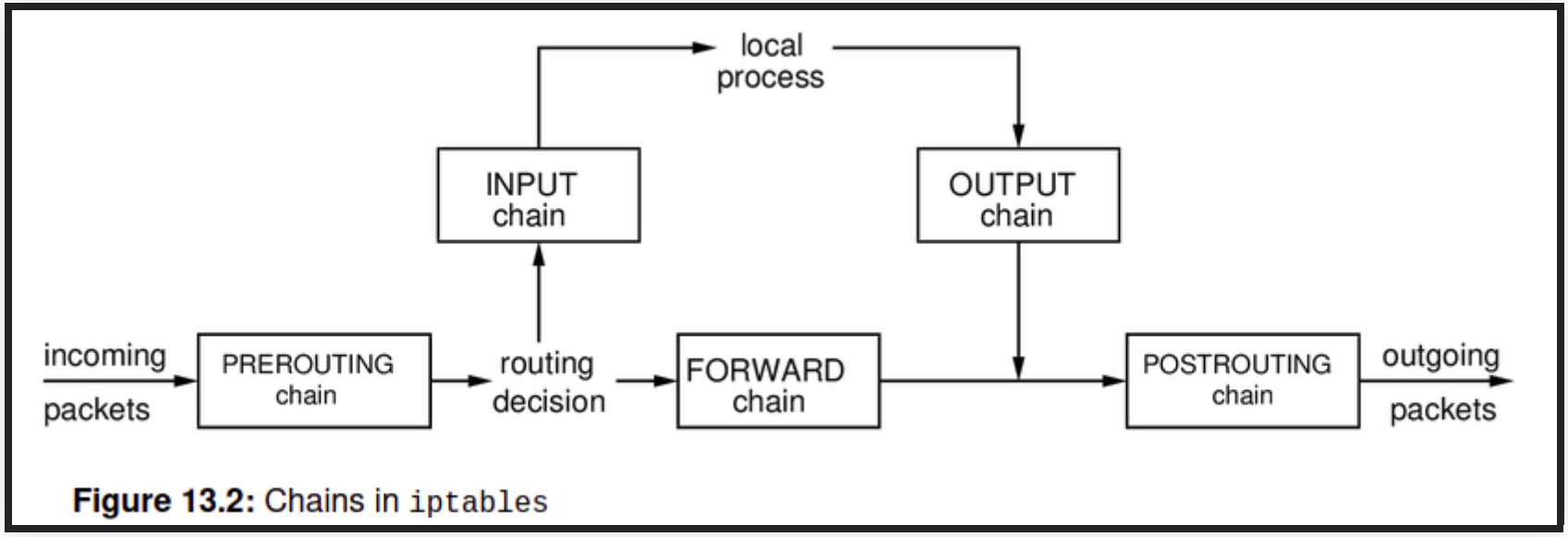


Figure 13.2: Chains in `iptables`

https://sandilands.info/nsl/Firewallswithiptables.html

iptables - tabele

- organizacja reguł ze względu na ich rolę

filter	nat	mangle	raw	security
INPUT	PREROUTING	PREROUTING	PREROUTING	INPUT
FORWARD	POSTROUTING	POSTROUTING	OUTPUT	OUTPUT
OUTPUT	INPUT	INPUT		FORWARD
	OUTPUT	OUTPUT		
		FILTER		

iptables - targets

- ACCEPT
- DROP
- REJECT
- MASQ
- TTL

...

nftables

nftables

- w kernelu od wersji 3.13 (2014)
- brak wbudowanych łańcuchów
- łatwiejsza konfiguracja
- nowa implementacja w oparciu o VM
- zupełnie inny język opisu reguł, dostępność złożonych struktur danych

nftables - bytecode

```
# nft add rule inet filter input tcp dport 22 drop
inet
    [ meta load l4proto => reg 1 ]
    [ cmp eq reg 1 0x00000006 ]
    [ payload load 2b @ transport header + 2 => reg 1 ]
    [ cmp eq reg 1 0x00001600 ]
    [ immediate reg 0 drop ]
```

nftables - składnia

```
% nft add table nat
% nft add chain nat postrouting { \
    type nat hook postrouting priority 100 \
}
% nft add rule nat postrouting ip saddr 192.168.1.0/24  \
oif eth0 snat to 1.2.3.4
```

```
#!/usr/bin/nft -f
flush ruleset
table ip nat {
    chain postrouting {
        type nat hook postrouting priority 100
        ip saddr 192.168.1.0/24 oif eth0 snat to 1.2.3.4
    }
}
```

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

/Performing_Network_Address_Translation_(NAT)

eBPF/XDP firewall

(e)BPF

- (extended) Berkeley Packet filter
- BPF od 1992, eBPF of wersji 3.18 (2014)
- VM w kernelu, pozwalająca na bezpieczne uruchamianie kodu w przestrzeni jądra
- analiza wydajności, filtrowanie pakietów…

man bpf

"eBPF programs can be written in a restricted C th
the clang compiler) into eBPF bytecode. Various fe
from this restricted C, such as loops, global variab
functions, floating-point numbers, and passing stru
arguments. (...) The kernel contains a just-in-time (
translates eBPF bytecode into native machine cod
performance."

XDP

- eXpress Data Path
- framework pozwalający na szybkie przetwarzanie pakietów z użyciem eBPF

"XDP provides bare metal packet processing at the
software stack which makes it ideal for speed with
programmability."

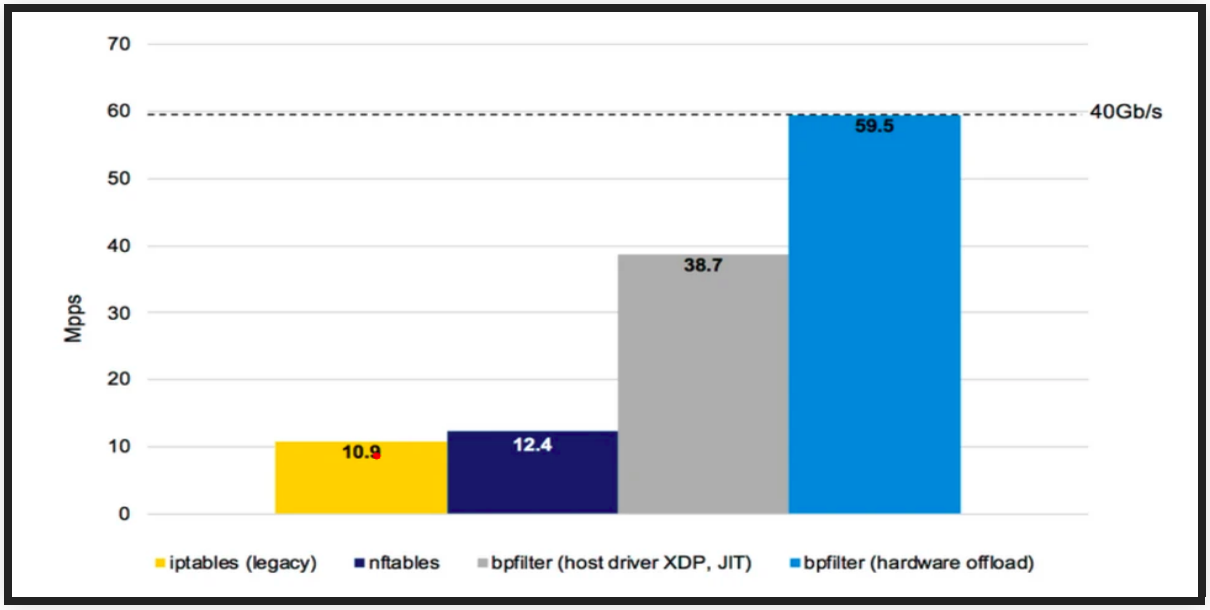
https://www.iovisor.org/technology/xdp

eBPF/XDP - przykład

Przykładowy program dropujący wszystkie pakiety:

```
1  #include<linux/bpf.h>
2  #define SEC(name) __attribute__((section(NAME), used))
3
4  SEC("xdp_drop")
5  int xdp_drop_prog(struct xdp_md *ctx) {
6          return XDP_DROP;
7  }
8
9  char _license[] SEC("license") = "GPL";
```

eBPF/XDP - wydajność



https://www.netronome.com/blog/bpf-ebpf-xdp-and-bpfilter-what-are-these-things-and-what-do-they-mean-enterprise/

IPS/IDS

IPS/IDS vs firewall

goes off you can have it activate locks in the buildi
police, but locking your door in the first place is a b

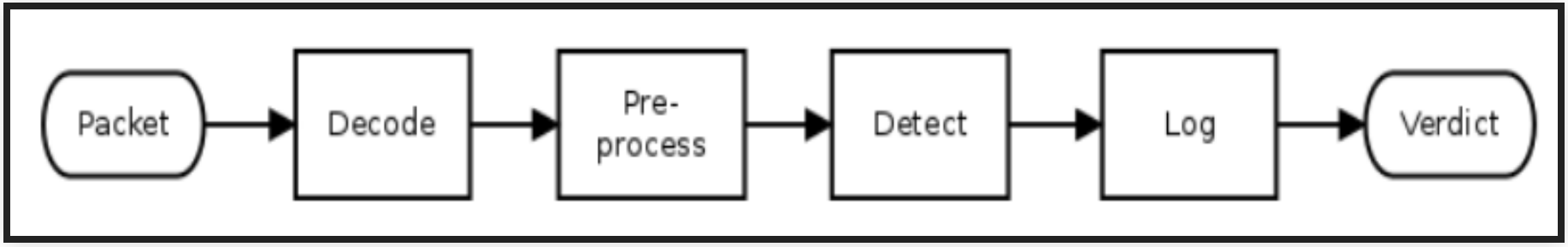
https://seclists.org/snort/2002/q3/1311

snort

snort

- stworzony w 1998, obecnie rozwijany przez *Cisco*
- dawniej C, obecnie C++ (od wersji 3)
- https://github.com/snort3/snort3

snort - przetwarzanie pakietów



https://github.com/snort3/snort3/releases/download/3.1.17.0

/snort_user.pdf

snort - preprocesory

- moduły/pluginy analizujące pakiety po zdekodowaniu, ale przed sprawdzaniem reguł
- defragmentacja pakietów, normalizacja danych przed przekazaniem od silnika reguł

```
preprocessor http_inspect_server: server default \
    http_methods { GET POST PUT SEARCH (...) } \
    chunk_length 500000 \
    server_flow_depth 0 \
    client_flow_depth 0 \
    post_depth 65495 \
    oversize_dir_length 500 \
    max_header_length 750 \
    max_headers 100 \
    max_spaces 200 \
    ...
```

snort - reguły

"Snort rules are divided into two logical sections, th
the rule options.The rule header contains the rule's
source and destination IP addresses and netmasks
destination ports information. The rule option sectio
messages and information on which parts of the pa
inspected to determine if the rule action should be

https://snort.org/documents/snort-users-manual

snort - reguły

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (\
    msg:"SCAN XMAS"; \
    flow:stateless; \
    flags:SRAFPU,12; \
    reference:arachnids,144; \
    classtype:attempted-recon; \
    sid:625; rev:7;)
```

```
alert tcp $EXTERNAL_NET any -> $TELNET_SERVERS 23 ( \
    msg:"TELNET Solaris memory mismanagement exploit attempt'
    flow:to_server,established; \
    content:"|A0 23 A0 10 AE 23 80 10 EE 23 BF EC 82 05 E0 D6
    classtype:shellcode-detect; \
    sid:1430; rev:7;)
```

fail2ban

fail2ban

- lekki, prosty IPS zapobiegający atakom typu brute-force
- analiza logów (*/var/log/auth.log*) w celu wykrycia ataku
- wykorzystuje *iptables* do blokowania adresów IP

fail2ban - konfiguracja

- pliki konfiguracyjne - */etc/fail2ban*
- architektura klient-serwer: *“The Server daemon monitors log file(s) and executes actions when a host is to be banned. The configuration of the Server is done by the Client which handles reading of configuration files”*
- *jail.{conf,local,d/}* - ustawienia dot. blokowania usług (ssh, http…)

fail2ban - przykład

plik */etc/fail2ban/jail.d/ssh.conf*

```
[sshd]
enabled=true
port=22
logpath=/var/log/auth.log
filter=sshd
maxretry=5
bantime=60
ignoreip=127.0.0.1
```

Źródła

- https://tldp.org/LDP/nag2/
- https://infosecwriteups.com/linux-kernel-communication-part-1-netfilter-hooks-15c07a5a5c4e
- https://developers.redhat.com/blog/2016/10/28/what-comes-after-iptables-its-successor-of-course-nftables#nftables_at_a_high_level
- https://www.youtube.com/watch?v=NN4rq4hzFfc&t=1529s
- https://wiki.gentoo.org/wiki/Nftables
- https://docs.cilium.io/en/latest/bpf/
- https://snort.org/documents/snort-users-manual

Dziękuję za uwagę