




Welcome to

# Building a Firewall

Show and tell

Henrik Kramselund he/him han/ham hlk@zencurity.com @kramse

Slides are available as PDF, kramse@Github   
building-a-firewall.tex in the repo security-courses

Slides are available as PDF kramshoej@Github

Not affiliated with the OpenBSD project, but a long time and very happy user

# Goal



We are building a new firewall, yay!

For this purpose I brought a lot of hardware! Really a lot!

- OpenBSD as a CPE in a network with ordinary internet traffic
- Arist router/switch in the datacenter
- Juniper EX enterprise switch
- We will discuss how to build a network with available devices
- Talk about BGP, PF and service daemons
- Mostly Juniper configuration and OpenBSD configs
- BGP to PF Tables, firewalling/NAT based on BGP updates
- OpenBSD niceness, why choosing OpenBSD made a lot of things easier
- Keywords: OpenBSD, BGP, routing, IEEE 802.1q, VLAN, IEEE802.1p, CoS/QoS, VoIP, firewalling

# Thank you OpenBSD



Donating to OpenBSD also supports OpenSSH development

OpenBSD has a complete *OpenBSD PF - User's Guide* available at:

<https://www.openbsd.org/faq/pf/index.html>

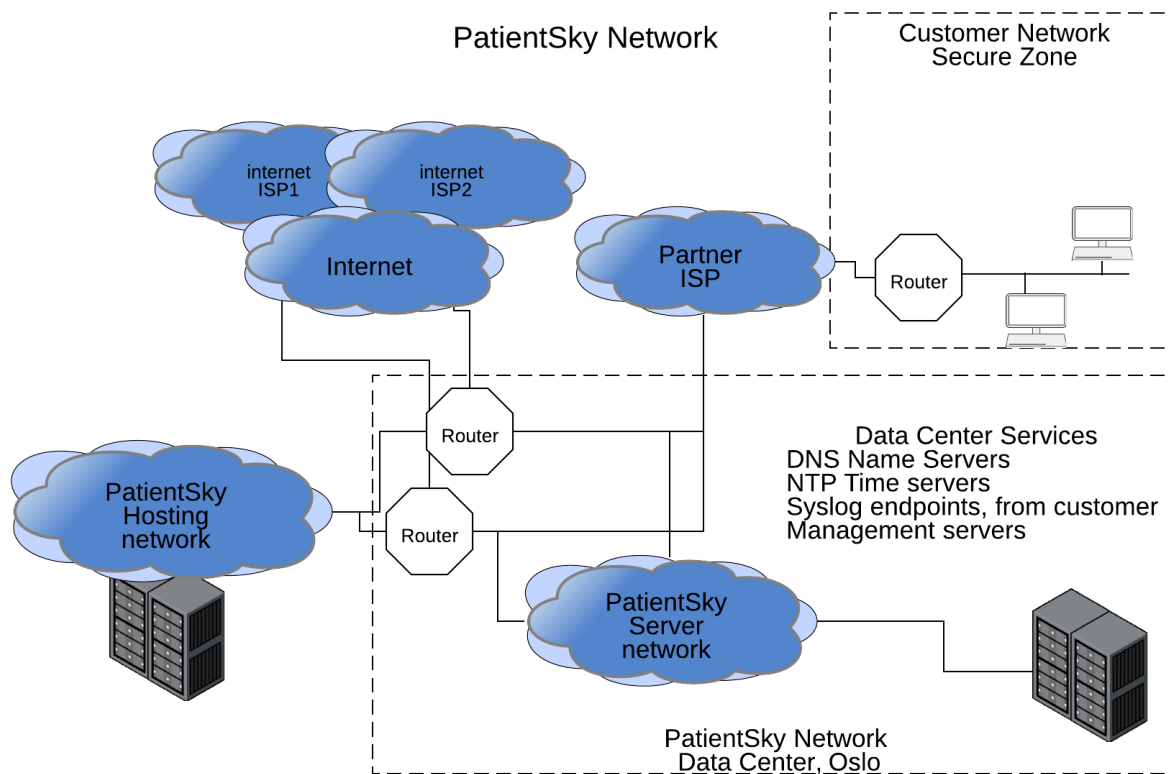
This includes a generic firewall example (4 pages if printed):

<https://www.openbsd.org/faq/pf/example1.html>

I will use some examples from a network I built with some interesting bits.

The same processes and features are available in commercial tools too

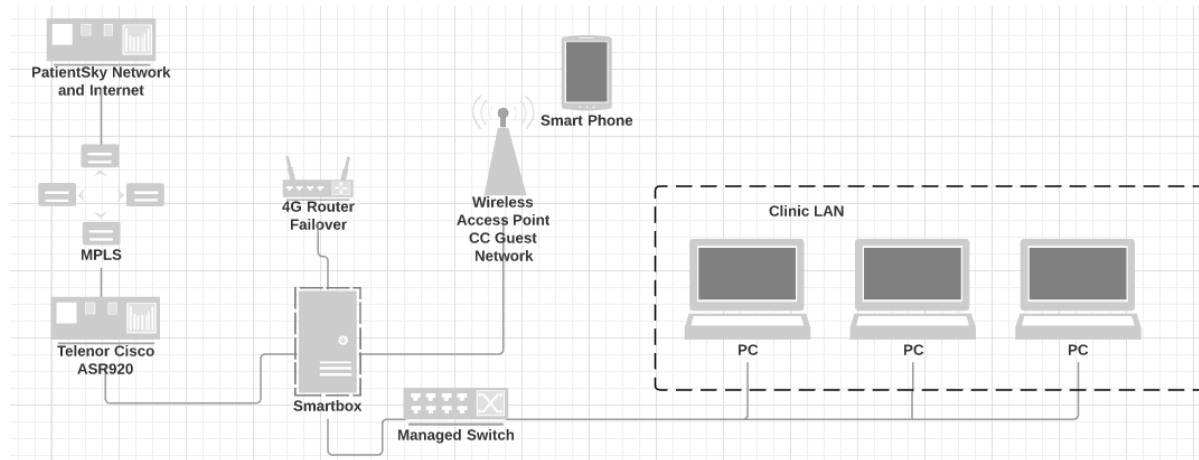
# Overview



# OpenBSD CPE: BGP, PF and service daemons

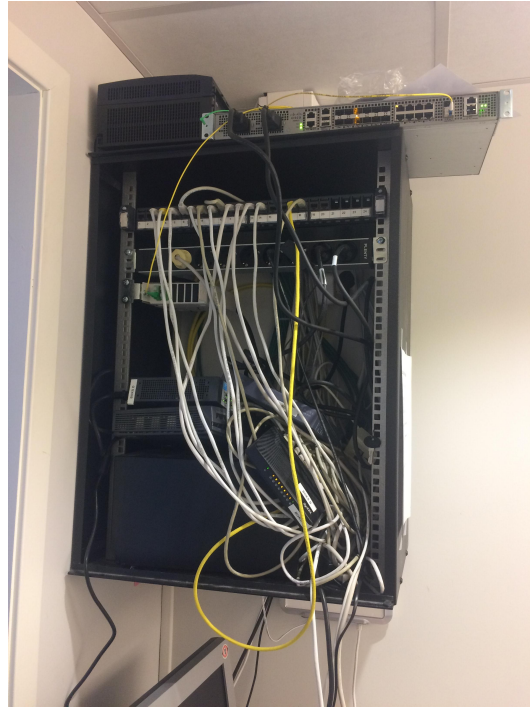


## Example network



- OpenBSD operating system
- Solid hardware + free operating system = reliable service
- ISP CPE, yes some ISPs use ASR920 as CPE for 8Mbit SHDSL too☺

## Existing infrastructure, in some places



Pretty nice heh, not always this bad, but usually not good

# Building a Network with a Firewall



Vi vil nu gennemgå netværksdesign med udgangspunkt i vores setup

Vores setup indeholder:

- Routers
- Firewall
- Wireless
- DMZ
- DHCPD, BGPD, OSPFD, ...
- Name service

Den kunne udvides med flere andre teknologier vi har til rådighed:

- VLAN inkl VLAN trunking/distribution, WPA Enterprise

Husk følgende slides er min mening, og fra et specifikt netværk

Typisk udvikler det sig over tid

# Principle of Economy of Mechanism



**Definition 14-4** The *principle of economy of mechanism* states that security mechanisms should be as simple as possible.

Simple — > fewer complications — > fewer security errors

Use WPA passphrase instead of MAC address based authentication



## Principle of Fail-Safe defaults



**Definition 14-3** The *principle of fail-safe defaults* states that, unless a subject is given explicit access to an object, it should be denied access to that object.

Default access *none*

In firewalls default-deny - that which is not allowed is prohibited

Newer devices today can come with no administrative users, while older devices often came with default admin/admin users

Real world example, OpenSSH config files that come with `PermitRootLogin no`

# Important processes and components



- OpenBSD kernel - does routing, thank you
- OpenBSD kernel - multiple routing tables, allow drop-in replacement in networks
- OpenNTP - time keeping
- OpenBGP - BGP to get NHN prefixes
- relayd - provides failover, change default route
- OpenSSHD - secure remote access
- DHCPD - dhcp service to LAN
- OpenBSD PF - awesome firewall software connecting it all nicely
- OpenBSD PF queue - allow detailed control of bandwidth
- OpenBSD PF prio into VLAN header - QoS/CoS of VoIP traffic
- NLnet Labs Unbound DNS server <https://www.nlnetlabs.nl/projects/unbound/about/>

The configurations shown in the presentation are examples from a real network

# Relayd failover to 4G router



```
# cat /etc/relayd.conf
primary = "185.xxx.xxx.x"
secondary = "192.168.8.1"
interval 10
table <gateways> { $primary ip ttl 1 priority 10, $secondary ip ttl 1 priority 50 }
router "uplinks" {
    route 0.0.0.0/0
    forward to <gateways> check icmp
}
```

Easy to understand, easy to implement

# OpenBSD queue pf.conf - from 20/20Mbit customer



```
# Queue to fix TCP originating from our host, if we send more than
# bandwidth the shaping done by ISP cause huge backoffs
queue root on em3 bandwidth 20M max 20M
```

```
# Currently used for VoIP and PatientSky Hosting
# Note: VoIP Max 25% of bandwidth, excess dropped by provider!
queue high parent root bandwidth 5M max 5M
queue normal parent root bandwidth 20M max 20M default
queue low parent root bandwidth 15M max 15M
```

```
# Download queues on inside interface to LAN
# by limiting this, we end up receiving less than max from outside
queue dn_parent on em2 bandwidth 20M max 20M
queue dn_high parent dn_parent bandwidth 20M max 20M
queue dn_default parent dn_parent bandwidth 15M max 15M default
```

```
# Wifi
queue guest_parent on em0 bandwidth 15M max 15M
queue guest_default parent guest_parent bandwidth 15M max 15M default
```

You can only limit what you send, download queues remove need for specific queue in data center for EACH customer!

# OpenBSD use queueing in rules



```
table <HOSTED_NETWORKS> const { 185.60.160.0/22 }

# Rules start here
block all

# Normal would be 3 and patientsky higher priority
pass out set queue normal set prio 3
pass out to <HOSTED_NETWORKS> set queue high set prio 5

# High prio on all traffic originating from us and our <HOSTED_NETWORKS> address space
pass in on egress from <HOSTED_NETWORKS> to (egress:0) set queue dn_high set prio 5
```

- When you limit outgoing - to the LAN, results is because of TCP it limits what you receive :-)
- Not really doing queueing inside LAN, some switches not controlled, customer responsibility
- If internal LAN with gigabit switches cannot handle VoIP, expect other problems

# OpenBGP download prefixes to PF Tables



```
...  
# neighbors and peers  
psnet="50033"  
group "nhn" {  
# peering to get NHN network  
    remote-as $psnet  
    neighbor 185.161.xxx.xx  
}  
  
# Our local network  
network 172.22.xxx.0/27  
  
# Filtering  
allow from any  
allow from AS 50033  
match from group "nhn" community $psnet:56828 set pftable "NHN"
```

Two functions, announce our local NHN prefix, internal LAN IP

and getting a table of almost 10.000 prefixes

# OpenBSD multiple routing domains are cool



with BGP running we can use the prefixes in rules, here no-NAT rule:

```
# towards end of pf.conf
# Routing Domain 1 used for LAN
anchor "inside" on rdomain 1 {
    # Allow administrative access when on-site
    pass in quick on em2 inet proto tcp from any to em2 port 34
    # Internal LAN must be allowed out
    pass in on em2
    # Guest network, no access to internal LAN or NHN
    # Prio 0 in ISP is Best Effort
    pass in on em0 to { !(em2:network) !<NHN> } set queue low set prio 0
    # Make sure our Hosted networks have priority and NHN traffic is sent through unharmed
    pass out quick to <HOSTED_NETWORKS> nat-to (egress:0) rtable 0 set queue high set prio 5
    pass out quick to <NHN> rtable 0 set queue normal set prio 3
    pass out to !<INSIDE_NETWORKS> nat-to (egress:0) rtable 0 set queue normal set prio 3
}
```

# OpenBSD priority



```
pass out quick to <HOSTED_NETWORKS> nat-to (egress:0) rtable 0 set queue high set prio 5
```

```
pass in proto tcp to port 25 set prio 2
```

```
pass in proto tcp to port 22 set prio (2, 5)
```

Prio is copied directly into IEEE 802.1q header, making it easy to use IEEE 802.1p

If the packet is transmitted on a `vlan(4)` interface, the queueing priority will also be written as the priority code point in the 802.1Q VLAN header. If two priorities are given, packets which have a TOS of lowdelay and TCP ACKs with no data payload will be assigned to the second one.

Hint: OpenSSH `sshd_config` using IPQoS can achieve the same



# Junos MX config, show configuration class-of-service



```
rewrite-rules {
  ieee-802.1 ISP {
    forwarding-class assured-forwarding {
      loss-priority low code-point 101;
      loss-priority high code-point 101;
      loss-priority medium-high code-point 101;
      loss-priority medium-low code-point 101;
    }
    forwarding-class expedited-forwarding {
      loss-priority low code-point 011;
      loss-priority high code-point 011;
      loss-priority medium-high code-point 011;
      loss-priority medium-low code-point 011;
    }
  }
}
```

Pro tip: this requires traffic to already be classified into these classes.

We solved it by sending VLAN traffic with prio from OpenBSD in data center to MX

## Junos outgoing, show configuration class-of-service



```
interfaces {
  ae0 {
    ...
    unit 1008 {
      classifiers {
        ieee-802.1 default;
      }
      rewrite-rules {
        ieee-802.1 ISP vlan-tag outer-and-inner;
      }
    }
  }
}
```

Note: We use double vlan-tags outer 1008 inner 100 in data center.  
This ends up with VLAN 100 on ALL hosts/sites

Result: We have the same simple config on all hosts

# OpenBSD niceness



Why choosing OpenBSD made a lot of things easier

- Free to install routers, firewalls, where we need them, no license
- Secure and stable, less worries, stable network yay!
- Nifty tricks with OpenBGP makes for a very elegant PF config
- PF integrated with IEEE 802.1p - on VLAN interfaces
- PF has a very readable format with syntactic sugar and dynamic constructs like `(em2:network)` the network on interface em2
- OpenBSD has stable release schedule, every 6 months

TL;DR Full control with easy transparent configs