class: center, middle

Network Management with the OpenBSD and NSH

BSDCan 2024

Presenter: Tom Smyth

Based on PF Tutorial materials developed by Peter Hansteen & Massimiliano Stucchi.

---

whoami (who am I), part Peter:

Peter Hansteen peter@bsdly.net

- Sysadmin, OpenBSD user since before the millenium
- Wrote The Book of PF, now in its third edition
- Blog at bsdly.blogspot.com about (lack of) sanity in IT
- Works at Tietoevry Create
- Yes, I'll do another book any decade now

---

whoami (who am I), part Max:

Massimiliano Stucchi

- Technical Advisor at The Internet Society
  - Here representing myself only
- IPv6 "Enthusiast"
- https://stucchi.ch
- @stucchimax@social.secret-wg.org

---

whoami (who am I), part Tom:

Tom Smyth

- working in IT since 2000
- CTO wireless Connect Ltd. an ISP in Ireland

- Opinions are mine and may be my companies also :)
- PF student, an avid reader of the Book of PF.
- I really Enjoy networking with OpenBSD
- Maintainer of the NSH network Shell for OpenBSD.

Introduce yourself

- A quick introduction about yourself:
  - Your name
  - Your favourite BSD
  - Your experience with networking
  - Your experience with PF
  - Your goal(s)

???

This introduction serves for us to understand the level of the room, and decide on how to better suit the tutorial.

Getting an idea of the level makes it so that we might quickly go through the basics, and focus on something more advanced, rather than spend time on something that everybody knows already.

Agenda

1. Introduction to NSH
2. NSH Basic functionality
3. Excercise: Selecting your prefered Editor
4. Exercise: Modifying Interface configuration
5. Exercise: Modifying Firewall settings
6. Exercise: Hosting Services, redirects
7. Installing NSH on OpenBSD
8. Exercise: Installing NSH on OpenBSD
9. Tips
10. Troubleshooting
11. Exercise:
12. End

---

class: center, middle

NSH Basics

Section 1

Unix Based Routers with a Unified Config.

- Some people are not comfortable with Unix Command line
- Some router clis are not much better
- However some modern router clis are significatnly easier to use.
- Staff Training, how many staff know Unix ?
  - Cli skills level ?
  - Unix skills level ?
- Linux BSD and other systems have inconsistent configuration / command syntax

---

enter NSH ***N***etwork ***SH***ell & its goals

- Shell and interpreter for configuring OpenBSD as a network appliance
- Guide the user in configuration with
  - brief command help with help command or ?
  - double command line completion
  - manual command to provide more detail in an easy to navigate help system
- allow a competent network engineer to harness the full power of OpenBSD without prior Unix Experience.
- keep configuration minimal (hide system default config values)
- unified configuration one configuration file to control all aspects of the router.
- intuitive configuration language similar to that commonly deployed commercial routers / switch
- dont rewrite / translate daemon configuration syntax (wrap around existing config systnax)

NSH ***N***etwork ***SH***ell History

Project started by Chris Cappuccio in 2002

- Developed on a part time basis over the years
- Tom Smyth joined the project as OpenBSD NSH package maintainer in March 2021
- Stefan Sperling joined the project in January 2023

Getting Started with NSH - ***N***etwork ***SH***ell (Interactively)

NSH can be set as a users default shell or started by executing nsh

- NSH has 3 main interactive modes
  - unprivileged mode is entered if a standard user executes nsh
    - allows user to run basic diagnostic commands such as ping tracert, show route, show arp
  - enable privileged mode is entered if the root user executes nsh or if a normal user enters the command 'enable'
    - (read config including sensitive config, but config cannot be modified (safety))
  - privileged config mode is entered from privileged mode by entering the command 'configure'
    - (modify configuration)

```
nsh# nsh
% NSH v1.1
nsh/enable
nsh(p)/configure
nsh(config-p)/exit
nsh(p)/disable
nsh/
```

- NB the different prompts for different NSH modes!

Getting Started with NSH - ***N***etwork ***SH***ell non interactive use

- NSH can be used to load configuration from a file (batch changes/ automation)
- update config - execute a series of NSH commands from a file

```
#nsh -c /home/config-script-to-update-config
```

- Initialise config (startup config)

```
#nsh -i /etc/nshrc
```

Getting Started with NSH - command help

- command ? - display brief command help for "command"

```
nsh(config-p)/pf ?
% Arguments may be abbreviated

   enable       enable pf firewall
   disable      disable pf firewall
   edit         edit, test and stage firewall rules
   check-config test and display staged firewall rules
   reload       test and apply staged firewall rules
nsh(config-p)/
```

- command [tab] [tab] displays a horisontal list of command options for "command"

```
nsh(config-p)/pf
check-config    disable     edit        enable      reload
nsh(config-p)/
```

Getting Started with NSH - Read The Fine Manual

- The manual is accessible within nsh with the manual command

  ```
  manual [search tag]
  ```

- Display the nsh manual page. If a search tag is specified then jump to the first section matching this tag if one or more matching tags exist.
- Alterntively one can access the nsh manual page in other OpenBSD shells with the man command

```
 man nsh
```

Getting Started with NSH - manual [command]

- manual command - opens the nsh manual at the correct page for "command"
- makes use of search tags in mandoc
- user can jump forward to next search tag with [t]
- user can jump back to previous search tag with [shift] [T]
- command [tab] [tab] displays a horisontal list of command options for "command"
- E.g. manual bridge

```
nsh(bridge-bridge101)/manual bridge
```

Getting Started with NSH - manual bridge command output

```
    [no] bridge [bridge-name]
     Modify bridge configuration on the named bridge or layer 2 forwarding
```

```
interfaces such as, bridge(4), veb(4), tpmr(4).  See also OpenBSD manual
pages for bridge(4), veb(4), tpmr(4) and ifconfig(8) (accessible via the
following nsh commands):

      !man bridge
      !man ifconfig
-   e.g. configure bridge settings on bridge1, and display bridge
    configuration help.
E.g show available bridge configuration commands.

      nsh(config-p)/bridge bridge100
      nsh(bridge-bridge100)/?
      % Commands may be abbreviated.
      % Type 'exit' at a prompt to leave bridge configuration mode.
      % Bridge configuration commands are:

        description   Bridge description
        member        Bridge member(s)
        span          Bridge spanning port(s)
```

Getting Started with NSH - manual command - [tab] [tab]

-Display all available search terms or commands in manual

```
nsh(config-p)/manual
ah      ftp-proxy   ldp     protected   span
arp     group       ldpd        quit        ssh
autoconf    help        lladdr      rdomain     switch
bgp     hostname    macaddress  reboot      switchport
bgpctl      hsrp        manual      relay       sync
bgpd        icmp        mbuf        relayd      syncdev
bridge      ifstate     monitor     resolv      tcp
bridgeport  ifstated    mpls        resolv.conf telnet
carp        igmp        multicast   rip     tftp
config      ike     nameserver  ripd        tftp-proxy
configure   iked        ndp     route       tpmr
crontab     ikev2       nppp        route6      traceroute
dhcp        inetd       ntpd        sadb        unsetenv
dhcpd       interface   ospf        sasync      veb
dvmrpd      isakmpd     patch       setenv      vpls
eigrp       kernel      pfsync      shell       vxlan
enable      l2vpn       ping6       smtp        wg
esp     label       pipex       smtpd       wireguard
flow        ldap        powerdown   snmp        write-config
flush       ldapd       privileged  snmpd       <cr>
```

```
nsh(config-p)/manual
```

Getting Started with NSH - manual Command - search tags

- user can jump forward to next search tag with [t]
- user can jump back to previous search tag with [shift] [T]

```
show bridge [bridge-interface | veb-interace | tpmr-interface]

Without specifying an argument, it displays all layer2 forwarding devices
configured on the system, and all members of each layer2 forwarding
device, and any description of the layer2 forwarding device.  Layer 2
forwarding devices supported by this command include bridge(4) standard
bridge, veb(4) virtual ethernet bridge and the tpmr(4) two port mac relay
device.
```

NSH - manual Command search tag continued

```
e.g. Display all layer2 forwarding devices and their member ports

      nsh(p)/show bridge
      % Bridge    Status  Member Interfaces
        bridge1   down
                  Description: -
        bridge100 up      vlan100
                  Description: Tom-Smyths-Bridge
        veb200    up      vlan200
                  Description: Chris-Cappuccios-Bridge
        tpmr102   up      vether1102 vether2102
                  Description: dlg-bridge
      nsh(p)/
e.g. Display the information the tpmr102 layer2 forwarding device
```

Getting Started with NSH - show command

- show commands are read only, they do not alter the state of the system, they are intended to give the user full visibility on selected aspects of the state of the system.
- E.g. show arp - displays Address Resolution Protocol

```
nsh/show arp
Host                                 Ethernet Address   Netif Expire     Flags
```

```
10.0.2.2                             52:54:00:12:35:02    em0 12m37s
10.0.2.15                            08:00:27:bd:cb:77    em0 permanent  l
```

Getting Started with NSH - show route Command

-E.g. show route display the IP route table of the system

```
nsh/show route
Flags: U - up, G - gateway, H - host, L - link layer, R - reject (unreachable),
       D - dynamic, S - static, T - MPLS, c - CLONED, l - LOCAL

% IPv4:
Destination        Gateway            Flags    Refs       Use    Mtu  Interface
0.0.0.0/0          10.0.2.2           UGS         6       881      -   em0
224.0.0.0/4        127.0.0.1          URS         0        53  32768   lo0
10.0.2.0/24        10.0.2.15          U           1         0      -   em0
10.0.2.2           52:54:00:12:35:02  UHLc        1        17      -   em0
10.0.2.15          08:00:27:bd:cb:77  UHL         0        43      -   em0
10.0.2.255         10.0.2.15          UH          0         0      -   em0
127.0.0.0/8        127.0.0.1          UGRS        0         0  32768   lo0
127.0.0.1          127.0.0.1          UH          1         2  32768   lo0
```

Getting Started with NSH - brief diagnostics

- NSH user can set the desired verbosity levels of any command run after setting the verbosity
- NSH displays brief diagnostics by default.

```
nsh/no verbose
% Diagnostic mode disabled
```

```
nsh/show interface em0
% em0
  Interface is up (last change 13:42:23), protocol is up
  Interface type Ethernet (Broadcast), hardware address 08:00:27:bd:cb:77
  Media type autoselect (1000baseT full-duplex), status active
  Internet address 10.0.2.15/24
  rdomain 0, MTU 1500 bytes (hardmtu 16110), Line Rate 1000 Mbps
  40634 packets input, 26668678 bytes, 0 errors, 0 drops
  32334 packets output, 12272854 bytes, 0 errors, 0 unsupported
  656 input, 379 output (average bytes/packet)
  0 collisions
```

Getting Started with NSH - verbose diagnostics

- NSH user can use the verbose command to increase the level of detai displayed by subsequent nsh commands.

```
nsh/verbose
% Diagnostic mode enabled

nsh/show interface em0
% em0
  Interface is up (last change 13:42:15), protocol is up
  Interface type Ethernet (Broadcast), hardware address 08:00:27:bd:cb:77
  Media type autoselect (1000baseT full-duplex), status active
  Internet address 10.0.2.15/24
  rdomain 0, MTU 1500 bytes (hardmtu 16110), Line Rate 1000 Mbps
  40632 packets input, 26668498 bytes, 0 errors, 0 drops
  32332 packets output, 12272674 bytes, 0 errors, 0 unsupported
  656 input, 379 output (average bytes/packet)
  0 collisions
  Flags:
    <UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>
  Hardware features:
    <CSUM_TCPv4,CSUM_UDPv4,VLAN_MTU,VLAN_HWTAGGING>
  Supported media types on em0:
    media 10baseT
    media 10baseT, mediaopt full-duplex
    media 100baseTX
    media 100baseTX, mediaopt full-duplex
    media 1000baseT, mediaopt full-duplex
    media 1000baseT
    media autoselect
```

Getting Started with NSH - show monitor

- show monitor is the implementation of route monitor in OpenBSD which displays any changes to the RIB Routing Information Base on the system

```
nsh/show monitor
% Entering monitor mode ... press ENTER or ^C to leave ...
% Message of size 192 on Tue May 23 12:33:35 2023
% RTM_ADD: Add Route: len 192, table 0, pid: 92724, seq 1, errno 0, flags:<UP,GATEWAY,DONE,S
% locks:  inits:
% sockaddrs: <DST,GATEWAY,NETMASK,IFP,IFA>
 100.64.0.0 127.0.0.1 255.192.0.0 lo0 127.0.0.1
% Message of size 192 on Tue May 23 12:33:48 2023
% RTM_DELETE: Delete Route: len 192, table 0, pid: 83139, seq 1, errno 0, flags:<GATEWAY,DON
% locks:  inits:
% sockaddrs: <DST,GATEWAY,NETMASK,IFP,IFA>
 100.64.0.0 127.0.0.1 255.192.0.0 lo0 127.0.0.1
^C% select: Interrupted system call
```

- E.g. above shows that an admin was adding and then removing a static route to 100.64.0.0/10 pointing to the loopack

---

Getting Started with NSH - config contexts - global context

- global configuration context contains configuration items that modify the sytem configuration such as:
  - hostname
  - enabling daemons such as
    - sshd
    - snmpd
    - relayd

```
nsh(config-p)/show run
!
hostname nsh
```

Getting Started with NSH - config contexts - interface context

- interface / bridge configuration context -allows the user query and change what is setup on the
- interface or bridge
- allows for unique per interface configuration
- similar behavior to other router / switch operatingg systems that are widely deployed.

```
nsh(config-p)/interface vio0
nsh(interface-vio0)/show active-config
interface vio0
 group egress
 autoconf4
!
```

Getting Started with NSH - show active-config

- One of the Design goals of NSH to have all config in one location

- Large configurations can be challenging when a user wants to just check and modify a small part of the config
- The show active-config command allows the user to display configuration on the currently active bridge or interface
  - before configuration changes are made
  - after configuration changes are entered
  - validate current context and configuration

```
nsh(config-p)/interface em0
nsh(interface-em0)/show active-config
interface em0
 group egress
 autoconf4
!
```

- The show active-config command only displays the active configuration in the currently selected interface or bridge
- This saves alot of scrolling on large configuratons!

Getting Started with NSH - show active-config

- show active configuration works in bridge context as well

```
nsh(config-p)/interface bridge101
nsh(bridge-bridge101)/show active-config
bridge bridge101
 description new bridge for nshtutorial demo
 group bridge
 shutdown
!
```

Getting Started with NSH - show ip

- Display a list of configured IP addresses
  - on what interfaces they are configured
  - on what rdomain are they are configured
  - how the IP address was configured

```
nsh(config-p)/show ip
Address    Interface  RDomain  Type
10.0.2.15  em0              0  dhcp
127.0.0.1  lo0              0  static
::1        lo0              0  static
fe80:4::1  lo0              0  link-local
nsh(config-p)/
```

Getting Started with NSH - show autoconf

- Displays a list dynamic / autoconfigured IP addresses,
  - what interfaces they are bound to
  - what other configuration was imported
    - default gateway
    - dns servers
  - and where the configuration was pulled from

```
nsh(config-p)/show autoconf
em0 [Bound]
    inet 10.0.2.15 netmask 255.255.255.0
    default gateway 10.0.2.2
    nameservers 192.168.67.221
    lease 23 hours
    dhcp server 10.0.2.2
nsh(config-p)/
```

Getting Started with NSH - Firewall configuration - pf command

- Firewall can be configured in NSH with pf command

```
nsh(config-p)/pf ?
% Arguments may be abbreviated

   enable       enable pf firewall
   disable      disable pf firewall
   edit         edit, test and stage firewall rules
   check-config test and display staged firewall rules
   reload       test and apply staged firewall rules
nsh(config-p)/
```

Getting Started with NSH - Firewall configuration - pf edit

- pf edit command will edit the firewall with your preferred editor

```
nsh(config-p)/pf edit
/var/run/pf.conf.0 is empty. Load an example config? [Y/n]
```

- If there was no firewall rules previously edited in NSH you will be asked, do you want to load an example configuration
- example config files are generally copied from /etc/examples

- it is recommended to load an example to get you started, and edit to suit your needs.

Getting Started with NSH - Firewall configuration - pf edit

```
       $OpenBSD: pf.conf,v 1.4 2018/07/10 19:28:35 henning Exp $
#
# See pf.conf(5) for syntax and examples.
# Remember to set net.inet.ip.forwarding=1 and/or net.inet6.ip6.forwarding=1
# in /etc/sysctl.conf if packets are to be forwarded between interfaces.

# increase default state limit from 100'000 states on busy systems
#set limit states 500000

set skip on lo

# filter rules and anchor for ftp-proxy(8)
#anchor "ftp-proxy/*"
#pass in quick inet proto tcp to port ftp divert-to 127.0.0.1 port 8021

pass            # establish keep-state
```

- Default pf rules as loaded by NSH
- Editor combined with pfctl is used to minimise code base of NSH
- Has an advantage of allowing users to edit config of multiple interdependent daemons and config before activating them

---

Getting Started with NSH - Firewall configuration - pf edit

- General configuration for pf
- Useful for debugging, applying default timeout values, etc.

```
#       $OpenBSD: pf.conf,v 1.4 2018/07/10 19:28:35 henning Exp $
#
# See pf.conf(5) for syntax and examples.
# Remember to set net.inet.ip.forwarding=1 and/or net.inet6.ip6.forwarding=1
# in /etc/sysctl.conf if packets are to be forwarded between interfaces.
INSERT BAD SYNTAX Error
# increase default state limit from 100'000 states on busy systems
#set limit states 500000
```

- NSH tests the config when saving the configuration on exiting the editor

```
/var/run/pf.conf.0:6: syntax error
nsh(config-p)/
```

Getting Started with NSH - Firewall configuration - pf check-config -error

- you can run a check of the staged pf with the command
- pf check-config

```
nsh(config-p)/pf check-config
Loaded 714 passive OS fingerprints
/var/run/pf.conf.0:6: syntax error
set skip on { lo }
nsh(config-p)/
```

- This is the equivalent of the pfctl -nvv command
- Checks the staged config (not the active config)

Getting Started with NSH - Firewall configuration - pf check-config -error

- Shows config until the first error is encountered

```
ksh# pfctl -nvvf /etc/pf.conf
Loaded 714 passive OS fingerprints
/var/run/pf.conf.0:6: syntax error
set skip on { lo }
ksh#
```

- Where was the error in the config ?

Getting Started with NSH - Firewall configuration - pf check-config -success

- pf check-config
- If syntax check passes it will display the list of rules in order.

```
nsh(config-p)/pf check-config
Loaded 714 passive OS fingerprints
set skip on { lo }
@0 block return all
@1 pass all flags S/SA
@2 block return in on ! lo0 proto tcp from any to any port 6000:6010
```

---

class: center, middle

Exercise 1

Protecting your host

Excercise 1 - Let's start

- Lab environment:
  - Open your favourite browser, then
  - Go to labs.pftutorial.net
- check that pf is indeed loaded and running (*hint*: pfctl)
- Try accessing other lab hosts

Exercise 1 - net config

- Configure the external interface on gateway
- *vi /etc/hostname.vio0*

```
inet 10.255.255.XX/24
!route add 0/0 10.255.255.254
inet6 fd18:b5d:cafe::XX/64
!route add -inet6 2000::/3 fd18:b5d:cafe::a
!route add -inet6 fd00::/8 fd18:b5d:cafe::a
```

- and then *vi /etc/resolv.conf*

```
nameserver 10.255.255.254
nameserver fd18:b5d:cafe::a
```

followed by

```
sh /etc/netstart
```

Exercise 1 - on gateway

- Start with a block ruleset

```
block
pass quick inet6 proto tcp from fd18::/16 to port ssh
    pass quick inet6 proto icmp6 from fd18::/16
```

- Allow traffic to be generated from your host, and allow ICMPv6

```
 pass from self
```

and then, reload *pf.conf*

```
pfctl -vnf /etc/pf.conf
pfctl -f /etc/pf.conf
```

- **NB:** Reload pf this way after every statement in the exercises

---

Exercise 1 - Tests

- From your gateway ping a host
- First IPv6

```
# ping6 fd18:b5d:cafe::a
PING fd18:b5d:cafe::a (fd18:b5d:cafe::a): 56 data bytes
64 bytes from fd18:b5d:cafe::a: icmp_seq=0 hlim=64 time=0.548 ms
64 bytes from fd18:b5d:cafe::a: icmp_seq=1 hlim=64 time=0.492 ms
64 bytes from fd18:b5d:cafe::a: icmp_seq=2 hlim=64 time=0.494 ms
```

- Then IPv4

```
# ping stucchi.ch
PING stucchi.ch (37.59.51.141): 56 data bytes
64 bytes from 37.59.51.141: icmp_seq=0 ttl=56 time=6.264 ms
64 bytes from 37.59.51.141: icmp_seq=1 ttl=56 time=6.273 ms
64 bytes from 37.59.51.141: icmp_seq=2 ttl=56 time=6.117 ms
```

Exercise 1 - Wrap up

- Does ping work?
- Do other commands work?
  - working from total block, proceed to make restricted workstation
    - name resolution
    - *http* and *https*
- Access public web sites, other Internet resources.
- What would it take to access the other lab hosts?

---

class: center, middle

Questions ?

???

Let's ask if there are any questions before continuing. Make sure we have everyone onboard.

---

class: center, middle

NSH Interaction with interfaces

Section 2

NSH compared with Openbsd

- mg /etc/hostname.em0
- sh /etc/netstart em0
- is equivalent to the following command

```
nsh(interface-em0)/?
% Commands may be abbreviated.
% Type 'exit' at a prompt to leave interface configuration mode.
% Interface configuration commands are:

  inet              IPv4/IPv6 addresses
  ip                Alias for "inet" command
  autoconf4         IPv4 Autoconfigurable address (DHCP)
  description       Interface description
  group             Interface group
  rdomain           Interface routing domain
  rtlabel           Interface route labels
  priority          Data packet priority
  llpriority        Link Level packet priority
  mtu               Maximum Transmission Unit
  metric            Routing metric
  link              Link level options
  arp               Address Resolution Protocol
  staticarp         Always use static ARP to find other hosts
...
```

Introducing NAT

- **N**etwork **A**ddress **T**ranslation (RFC1631 onwards)
- -> 'Hide' several hosts behind 1 or more public addresses, using RFC1918 addresses
- -> can be used by ISPs for conserving scarce IP addresses in large networks (CG-NAT) 100.64.0.0/10
- Modern PF has *nat-to* on 'pass' and 'match' rules:

```
match out on $extif inet nat-to ($extif)
```

- *Neat trick*: egress is the interface group that has a default route, you can filter on it

```
match out on egress inet nat-to (egress)
```

- In modern networks we **should** (also) have IPv6 (inet6)

???

NAT, the stopgap measure that's old enough to drink, more than 22 years old. NAT was created as a temporary measure that hasn't been replaced by now. Not even IPv6.

We haven't discusses "egress" yet, so this is the right time to introduce it.

Spend some time discussing it, along with the rest of the NAT specifications.

Unfortunately, there's also NAT for IPv6, called NAT66

A (filtering) Gateway

"I decide which packets pass"

Enable forwarding:

- Temporarily set from command line with sysctl:

```
# sysctl net.inet.ip.forwarding=1
# sysctl net.inet6.ip6.forwarding=1
```

- Make permanent in /etc/sysctl.conf

```
net.inet.ip.forwarding=1
net.inet6.ip6.forwarding=1
```

The minimal gateway

- Do you *NAT* for IPv4? Of course you do.
- Do you run IPv6? Of course you do.

```
ext_if=bge0
int_if=bge1
match out on egress inet nat-to ($ext_if)
block all
pass proto tcp from { self, $int_if:network }
```

- The "pass" rule, withouth *inet* or *inet6* applies to both

Keep in mind: This is a point of policy enforcement

A Point of policy enforcement

- Now some policy, and macros

```
  ext_if=bge0
  int_if=bge1

  client_out = "{ ftp-data, ftp, ssh, domain, pop3, auth, nntp, http, \
      https, 2628, 5999, 8000, 8080 }
  udp_services = "{ domain, ntp }"

  match out on egress inet nat-to ($ext_if)
* block
* pass quick proto { tcp, udp } to port $udp_services keep state
*
* pass proto tcp from $int_if:network to port $client_out
*
* pass proto tcp to self port ssh
```

- What services do your clients consume?

???

Log to on the system we have for showing out, and then show the rules there and how they expand to different parts.

Letting dhcpd(8) direct access

OpenBSD dhcpd(8) can interact with your ruleset via tables:

/etc/rc.conf.local

```
dhcpd_flags="-L leased_ip_table -A abandoned_ip_table -C changed_ip_table bge1"
```

```
  ext_if=bge0
  int_if=bge1
* table &lt;abandoned_ip_table&gt; persist counters
* table &lt;changed_ip_table&gt; persist counters
* table &lt;leased_ip_table&gt; persist counters

  client_out = "{ ftp-data, ftp, ssh, domain, pop3, auth, nntp, http, \
                  https, 2628, 5999, 8000, 8080 }"
  udp_services = "{ domain, ntp }"

  match out on egress inet nat-to ($ext_if)
  block
  pass quick proto { tcp, udp } to port $udp_services keep state
* pass proto tcp from &lt;leased_ip_table&gt; to port $client_out
  pass proto tcp to self port ssh
```

=> only pass traffic from hosts with active leases from *me*

???

Maybe move this slide to a later section.

Redirects (and divert-to)

Modern PF has two classes of redirect

- **rdr-to** on match and pass rules - rewrite destination address while filtering (locally or even to other hosts)

  ```
  pass in on egress to port www rdr-to $webserver
  ```

- **divert-to** on match and pass rules - divert() socket for local use

  ```
  pass in on egress to port smtp divert-to 127.0.0.1 port spamd
  ```

FTP Proxy

- If your users need to access FTP services, ftp-proxy is what you need
- FTP does not easily pass through a block firewall, some help is needed

```
$ doas rcctl enable ftpproxy6
```

- or for IPv4

```
$ doas rcctl enable ftpproxy
```

- and then add an anchor and divert rules to your config

```
anchor "ftp-proxy/*"
  ...
pass in quick inet proto  tcp to port ftp divert-to 127.0.0.1 port 8021
pass in quick inet6 proto tcp to port ftp divert-to ::1 port 8021
pass out proto tcp from $proxy to port ftp
```

There is even a reverse mode (**-R**) for when you host FTP servers, see man ftp-proxy

---

class: center, middle

Exercise 2

Protecting your network

Exercise 2 - Goals

- Your network grows, you become a gateway
- Extend the configuration to enable the network to access the internet

---

background-image: url(images/exercise2.png)

Exercise 2 - Your network

Exercise 2

- Turn on ip forwarding (sysctl)

  ```
  # sysctl net.inet.ip.forwarding=1
  # sysctl net.inet6.ip6.forwarding=1
  ```

- Set up NAT

  ```
  match out on egress inet nat-to (egress)
  ```

Also, pass traffic from that local net

Exercise 2 - preparation

- Configure the hosts with the following IPv6 addresses
  - *Gateway (vio1):* fd18:b5d:XX::a/64
  - *Host1:* fd18:b5d:XX::80/64
  - *Host2:* fd18:b5d:XX::25/64 <br/ >
- On Host1 and Host2, set fd18:b5d:XX::a as the default IPv6 gateway
- and also the following IPv4 addresses
  - *Gateway (vio1):* 192.168.XX.1/24
    - *Host1:* 192.168.XX.2/24
    - *Host2:* 192.168.XX.3/24
- On Host1 and Host2 set 192.168.XX.1 as the default IPv4 gateway

---

Exercise 2 - check your results

- From client 1, ping a host on the internet
- First IPv6

```
# ping6 stucchi.ch
PING stucchi.ch (2001:41d0:8:6ed8::80): 56 data bytes
64 bytes from 2001:41d0:8:6ed8::80: icmp_seq=1 hlim=56 time=7.414 ms
64 bytes from 2001:41d0:8:6ed8::80: icmp_seq=2 hlim=56 time=6.333 ms
64 bytes from 2001:41d0:8:6ed8::80: icmp_seq=3 hlim=56 time=6.441 ms
```

- Then IPv4

```
# ping stucchi.ch
PING stucchi.ch (37.59.51.141): 56 data bytes
64 bytes from 37.59.51.141: icmp_seq=0 ttl=56 time=6.264 ms
64 bytes from 37.59.51.141: icmp_seq=1 ttl=56 time=6.273 ms
64 bytes from 37.59.51.141: icmp_seq=2 ttl=56 time=6.117 ms
```

Exercise 2b: FTP

Try fetching *ftp://ftp.ripe.net/pub/stats/ripencc/delegated-ripencc-extended-latest*

```
# wget ftp://ftp.ripe.net/pub/stats/ripencc/delegated-ripencc-extended-latest
```

Check your result

If it didn't work, configure FTP-proxy and try again.

---

class: center, middle

Installing NSH

Section 4

Installing NSH Package and build

- NSH is available as a package
- Recommend installing the package as it sets up scripts nicely

```
pkg_add nsh
```

- NSH is available on github
- NSH will be available on got
- Build the latest master snapshot

```
git clone https://github.com/yellowman/nsh
cd nsh
make
su
make install
```

Installing NSH - Integrating NSH as main admin interface*

- Caution advised on existing production systems!!!
- Scripts are available to import most settings into running config file

```
git clone https://github.com/yellowman/nsh
cd nsh
cd scripts
cd shell
```

-If you want to have NSH take over completely

- and move config files into /var/run/

```
chmod 700 extensive-nsh-openbsd-integration.sh
su
./extensive-nsh-openbsd-integration.sh
```

Installing NSH setting NSH as the default shell for a user

- Login as the user
- locate nsh
- at the command prompt run chsh -s /usr/local/bin/nsh

```
chsh -s /usr/local/bin/nsh
```

class: center, middle

Exercise 3

Offering services

Excercise 3 - Goals

- You're now offering services
- **Host 1** will provide **http** service
- **Host 2** will provide **smtp** service
- We need to setup:
  - The services
  - Redirects
  - Firewall rules

---

background-image: url(images/exercise3.png) class: right

Exercise 3 - Network

???

This is not exactly the network we have, but could be thought as such.

Host1 and Host2 are in what could be considered our DMZ.

Exercise 3 - on Host1

- We need to configure and start httpd

```
 # cp /etc/examples/httpd.conf /etc/httpd.conf

 < comment out the HTTPS part >

 # rcctl enable httpd
```

```
# rcctl start httpd
httpd(ok)
```

Exercise 3 - on Host2

- Change the config to listen on all interfaces:
  - Change the appropriate line in

/etc/mail/smtpd.conf

```
listen on all
```

- Then start the daemon

```
# rcctl enable smtpd
# rcctl start smtpd
smtpd(ok)
```

- (It might take a while)

Exercise 3 - on gateway

/etc/pf.conf

```
webserver_v4 = "$IP_addr_of_host1"
webserver_v6 = "fd18:b5d:XX::80"
webports = "{ http, https }"
emailserver_v4 = "$IP_addr_of_host2"
emailserver_v6 = "fd18:b5d:XX::25"
email = "{ smtp, pop3, imap, imap3, imaps, pop3s }"

match in on egress inet proto tcp to egress port $webports rdr-to $webserver_v4
match in on egress inet proto tcp to egress port $email rdr-to $emailserver_v4

pass inet proto tcp to $webserver_v4 port $webports
pass inet proto tcp to $emailserver_v4 port $email
pass log inet proto tcp from $emailserver_v4 to port smtp

pass inet6 proto tcp to $webserver_v6 port $webports
pass inet6 proto tcp to $emailserver_v6 port $email
pass log inet6 proto tcp from $emailserver_v6 to port smtp
```

- **NB:** No redirects are needed for IPv6

Exercise 3 - checks

- Try connecting to the HTTP and SMTP port of your friends/neighbours:
- From Gateway:

```
telnet -6 fd18:b5d:XX::80 80
telnet -4 10.255.255.XX 80
```

- and

```
telnet -6 fd18:b5d:XX::25 25
telnet -4 10.255.255.XX 25
```

Tips

- Decide your network topology
  - DMZ (?)
  - Multi-customer (?)
  - Multi-customer, Multi-DMZ(?)
- Segment off your subnets
  - IPv4 (Do you NAT)?
  - IPv6
  - Do you do NAT64?
- Per subnet (customer)
  - Which services do you expose?
  - Write the rules
  - pamper^H^H^H^Hproxying

---

class: center, middle

Tips

Section 5

Choosing your ISP, a quick guide

- Are they national or regional IX members?
- Do they have geographical redundancy ?
  - or do you need to arrange that for yourself ?

- Do they actually understand your questions about peering, routing, multiple paths?
  - (avoid consumer oriented SOHO-only shops)
- Do they *suck*?

Getting transit

- Find well peered transit providers
  - Can improve quality and shorten AS paths
  - No capacity problems
- Find your top traffic destinations:
  - Can improve quality
  - Peer with them or find closer upstream
  - Traffic profile from flow collectors can be useful

Common mistakes

- No diversity
  - All reached over same cable
  - All connect to the same transit
  - All have poor onward transit and peering arrangements
- Signing up with too many transit providers
  - Lots of small circuits
  - These cost more per Mbps than larger ones

Basic OpenBGPd configuration, operation and interaction with PF

- **B**order **G**ateway **P**rotocol
  - Manage and exchange route information with BGP peers
- Once you have the ASn registered, do the basic config.
- In your *pf.conf*:
  - enable BGP to pass between your routers and your peers' -- **TCP and UDP 179**
- **Neat trick**: Define tables in your pf.conf
  - bgpd maintains them via **pftable** attributes on bgpd.conf objects

---

Use cases for OSPF, BGP or ECMP

- **OSPF: O**pen **S**hortest **P**ath **F**irst
  - is a IGP **I**interior **G**ateway **P**rotocol
  - Each router maintains link state information for links and networks within your AS
  - Calculates routing cost
  - Use ospf6d for IPv6
  - Use ospfd for IPv4
  - Need to *pass proto ospf* between routers.
- **BGP:** announces and receives routes
  - can be both an IGP or EGP **E**xterior **G**ateway **P**rotocol
  - highly scalable (Internet scale)
  - can be used for signaling and sending additional information with route announcements
  - Use bgpd
  - need to *pass proto tcp port 179* between routers

Use cases for OSPF, BGP or ECMP (cont)

- **ECMP: E**qual **C**ost **M**ulti-**P**ath
  - target reachable via more than one route
  - load distribution or redundancy over multiple links
  - **Tip** Use ifstated to handle link downtime.

BCP38, MANRS and Internet peering

"**BCP38**" -- Discussed also in another effort

Mutually **A**greed **N**orms for **R**outing **S**ecurity (MANRS)

- Define four concrete actions network operators should implement
- Coordination
  - Keep your contacts updated
- Validation
  - Route objects, RPKI, BGPSec
- Anti-spoofing
- uRPF
- Filtering on external Interfaces facing external suppliers
  - Drop inbound Traffic with a src IP claiming to be from your networks / private networks.
  - Drop outbound Traffic with a src IP address that is not in your Public IP network range.

- Build a visible community of security-minded operators
- Valuable resource: The Routing Manifesto

Introducing VXLAN in your network

vxlan - the **V**irtual e**X**tensible **L**ocal **A**rea **N**etwork tunnel interface

- Pushes layer 2 network (Ethernet frames) over layer 3 (IP) tunnels
  - 24-bit *vnetid* (vs max 4k VLANs)
- Has *no* built in security
- Intended for '*trusted*' (Datacenter, inter-hypervisor) environments
  - Otherwise, consider transport over IPSEC.
- Default transport over **UDP 4789** (aka **vxlan**)
- make sure that traffic passes between endpoints

Introducing VXLAN in your network

```
 # ifconfig vxlan0 tunnel 192.168.100.101 192.168.200.201 vnetid 17
 # ifconfig vxlan0 10.11.12.100/24
```

```
 # ifconfig vxlan0 tunnel 192.168.200.201 192.168.100.101  vnetid 17
 # ifconfig vxlan0 10.11.12.101/24
```

```
table &lt;vxendpoints&gt; { 192.168.200.201 192.168.200.204 }
pass from &lt;vxendpoints&gt; to port vxlan
```

Buy Reyk a beer.

Readable and maintainable toolsets

- **Macros**
  - descriptive names, keep uniform
- **Tables**
  - descriptive names
  - consider daemon/scripting interface
- **Interface groups**
  - you know egress already
  - make your own and filter on them
- **Anchors**

  - group rules by common criteria
  - tagging
  - interface or group

- Service names vs port numbers

- **Comments** - yes, you **will** forget why this was a good idea

---

Useful 3rd party packages (ports) for OpenBSD

OpenBSD base operating system can be supplimented by the following packages and features:

- pftop - a curses-based utility for real-time display of active states and rules for pf. It is a cross between top and pfctl -sr and pfctl -ss.
  - pftop can be installed with the following command
    pkg_add pftop
  - nsh **n**etwork **sh**ell
  - nsh can be installed with the following command
    pkg_add nsh

Now let's add wireless

- Wireless used to be hard, (WPA in particular), now it's 'just another interface'
- 802.11* support in OpenBSD has a,b,g,n, ac only in some drivers (bwfm(4))
- Not all drivers support hostap
  - check man pages before buying kit for access point use
- Optionally setup with commercial APs for radio part
  - do DHCP, filtering, authentication and so forth from OpenBSD

---

class: center, middle

Questions ?

???

Let's ask if there are any questions before continuing. Make sure we have everyone onboard.

---

class: center, middle

Troubleshooting

Section 6

"It's all your fault. Until you track down and fix the root cause."

Troubleshooting 101: ICMP(v6)

- ICMP: **I**nternet **C**ontrol **M**essage **P**rotocol
- The *ping of death* scare is almost over, let's enable ping:

```
icmp_types = "{ echoreq, unreach }"

pass inet proto icmp all icmp-type $icmp_types keep state

pass inet proto icmp from $localnet icmp-type $icmp_types
pass inet proto icmp to $ext_if icmp-type $icmp_types
pass inet6 proto icmp6 from $localnet icmp6-type $icmp6_types
pass inet6 proto icmp6 to $ext_if icmp6-type $icmp6_types
```

- **echoreq**: lets ping do its thing
- **unreach**: lets you do *path MTU discovery* (PMTUD)

Troubleshooting 101: Statistics

- Statistics can be had with **pfctl -s info**

For statistics (bytes/packets passed per rule) attach *labels* per rule

```
pass log proto { tcp, udp } to $emailserver port smtp label "mail-in"
pass log proto { tcp, udp } from $emailserver to port smtp label "mail-out"
```

```
$ doas pfctl -vs rules
pass inet proto tcp from any to 192.0.2.225 port = smtp flags S/SA keep state label "mail-i
[ Evaluations: 1664158 Packets: 1601986 Bytes: 763762591 States: 0 ]
[ Inserted: uid 0 pid 24490 ]
pass inet proto tcp from 192.0.2.225 to any port = smtp flags S/SA keep state label "mail-o
[ Evaluations: 2814933 Packets: 2711211 Bytes: 492510664 States: 0 ]
[ Inserted: uid 0 pid 24490 ]
```

Troubleshooting 101: Statistics

- If you need to pass the data to a script
- Or a database

- A graphing engine

```
$ doas pfctl -zvsl
mail-in 1664158 1601986 763762591 887895 682427415 714091 81335176
mail-out 2814933 2711211 492510664 1407278 239776267 1303933 252734397
```

Troubleshooting 101: log to pflog

Rules with the **log** keyword log packet data to the pflog device(s)

```
# log blocked packets
block log(all)

# logs initial packet of matching connections:
pass log proto tcp to port ssh

# logs all matching packets:
pass log(all) proto tcp to port ssh log(all)

# logs matches on this and all succeeding rules
pass log(matches) proto tcp to port ssh

# logs all packets matches on this and all succeeding rules
pass log(all, matches) proto tcp to port ssh
```

```
match log(all, matches) # log *everything*
```

Troubleshooting 101: tcpdump, read from pflog

- tcpdump is your friend
- Let it loose on the pflog device:

```
$ doas tcpdump -n -e -ttt -i pflog0
tcpdump: WARNING: snaplen raised from 116 to 160
tcpdump: listening on pflog0, link-type PFLOG
May 29 21:06:27.165561 rule def/(match) pass in on bge1: 192.168.103.126.15526 >
213.187.179.198.22: . ack 2951513182 win 16332 (DF) [tos 0x10]
May 29 21:06:27.166934 rule 16/(match) pass in on bge0: 158.36.191.135.22 >
213.187.179.198.59516: . ack 1734404306 win 64800 [tos 0x8]
May 29 21:06:27.166939 rule 2/(match) match in on bge0: 158.36.191.135.22 >
213.187.179.198.59516: . ack 1 win 64800 [tos 0x8]
May 29 21:06:27.168340 rule def/(match) pass out on bge1: 213.187.179.198.22 >
192.168.103.126.15526: P 69:153(84) ack 0 win 17520 [tos 0x10]
May 29 21:06:27.169150 rule def/(match) pass out on bge1: 213.187.179.198.22 >
```

```
192.168.103.126.15526: P 153:333(180) ack 0 win 17520 [tos 0x10]
May 29 21:06:27.169265 rule def/(match) pass out on bge1: 213.187.179.198.22 >
```

- **NB** rule number, matches your *loaded* rule set

Troubleshooting 101: Hitting and avoiding limits

- On busy systems, you may need to raise limits from default values
- Check with:

```
$ doas pfctl -s info
```

- versus the output of **pfctl -s memory** and **pfctl -s timeouts**
- You may need to bump up from defaults:

```
# increase state limit from 10'000 states on busy systems
set limit states 100000
# increase no of source nodes
set limit src-nodes 100000
```

Troubleshooting 101: netflow aka pflow (IPFIX)

- Records TCP/IP *flow* metadata
  - srcIP
  - dstIP
  - (srcPort, dstPort)
  - startTime
  - endTime
  - Packets
  - Bytes
- OpenBSD has the pflow(4) virtual network interface
  - which generates the datagrams from the state table
- Useful for network monitoring, DDoS protection, etc.

Troubleshooting 101: netflow setup

- Set up a *sensor*:

```
$ cat /etc/hostname.pflow0
flowsrc 192.168.103.1 flowdst 192.168.103.252:9995
pflowproto 10
```

- Then configure your *collector* at the **flowdst** IP address for analysis and network overlordship.
- Lots of collector options available in ports: nfsen, flow-tools, pmacct, FastNetMon and others.
- More info:
  - Michael W. Lucas: Network Flow Analysis
  - and Peter N. M. Hansteen: Yes, You Too Can Be An Evil Network Overlord - On The Cheap With OpenBSD, pflow And nfsen.

Flow Anlyser example Fastnetmon

- Example of a typcial flow anlayser software fastnetmon:
  - User can view FastNetMon statistics via the CLI client fastnetmon_client

```
# fastnetmon_client
FastNetMon 1.1.7 master git- Try Advanced edition: https://fastnetmon.com
IPs ordered by: packets
Incoming traffic       1505664 pps  15397 mbps     85 flows
37.203.[redacted]        59184 pps    485 mbps      0 flows
37.203.[redacted]        45040 pps    504 mbps      0 flows
37.203.[redacted]        26924 pps    270 mbps      0 flows
185.55.[redacted]        24211 pps    240 mbps      0 flows
5.134.[redacted]         23872 pps    290 mbps      0 flows
45.11.[redacted]         23634 pps    250 mbps      0 flows
185.55.[redacted]        22451 pps    255 mbps      0 flows
45.11.[redacted]         20943 pps    254 mbps      0 flows
185.55.[redacted]        20298 pps    246 mbps      0 flows
5.134.[redacted]         20188 pps    236 mbps      0 flows
```

- With FastNetMon one can implement mitigations based on tresholds
  - Packets per second pps
  - Bandwidth per second Mbps

---

class: center, middle

Exercise 4

Queueing

Exercise 4 - Goals

- With the configs from exercise 3, now add:
- A set of queues, and
- Statements to add rules to the queues

Exercise 4 - on Gateway

- Configure the queues

/etc/pf.conf

```
queue rootq on $ext_if bandwidth 20M
    queue main parent rootq bandwidth 20479K min 1M max 20479K qlimit 100
        queue default parent main bandwidth 9600K min 6000K max 18M default
        queue http parent main bandwidth 9600K min 6000K max 18M
        queue smtp parent main bandwidth 9600K min 6000K max 18M
    queue spamd parent rootq bandwidth 1K min 0K max 1K qlimit 300
```

Exercise 4 - on Gateway

- and then apply them to the match statements

/etc/pf.conf

```
match in on egress inet proto tcp to egress port $webports rdr-to $webserver_v4 \
      queue http
match in on egress inet proto tcp to egress port $email rdr-to $emailserver_v4 \
      queue smtp

pass inet6 proto tcp to $webserver_v6 port $webports set queue http
pass inet6 proto tcp to $emailserver_v6 port $email set queue smtp
pass log inet6 proto tcp from $emailserver_v6 to port smtp set queue smtp
```

Exercise 4 - Check

- Check the queues have been effectively created

```
# systat queues
```

- or, alternatively

```
# pfctl -vsq
```

---

class: center, middle

Questions ?

Last chance...

or nshtutorial@ogmaconnect.com

for tutorial specific questions about NSH

or nsh@lists.deschutesdigital.com for general questions for the community and community based help. Let's ask if there are any questions before continuing. Make sure we have everyone onboard.

Web accessible resources

OpenBSD website and documentation

[http://www.openbsd.org/%5D(http://www.openbsd.org/) The official OpenBSD website – to donate: [http://www.openbsd.org/donations.html%5D(http://www.openbsd.org/donations.html) and please do donate, corporates may prefer [https://www.openbsdfoundation.org/%5D(https://www.openbsdfoundation.org/) - a Canadian non-profit

The PF User Guide on the OpenBSD web site

OpenBSD online man pages

Note: You can convert the man page of pf.conf to PDF for reading in your favourite reader with the command:

man -T pdf pf.conf > pf.conf.pdf

Resources

Books / e-Books

Michael W Lucas: Absolute OpenBSD, 2nd ed.

Peter N. M. Hansteen: The Book of PF, 3rd ed.

Elizabeth D. Zwicky, Simon Cooper, D. Brent Chapman Building Internet Firewalls, 2nd ed.

Blogs

[http://undeadly.org/%5D(http://undeadly.org/) - The OpenBSD Journal news site

[http://bsdly.blogspot.com/%5D(http://bsdly.blogspot.com/) - Peter's rants^H^H^H^H^Hblog posts

[http://www.tedunangst.com/flak/%5D(http://www.tedunangst.com/flak/) tedu@ on developments

---

background-image: url(images/end.png)

-

???

Notes for this slide

---

background-image: url(images/end2.png)

-

???

Notes for this other slide
