IPv6 Security Practice

János Mohácsi

NIIF/HUNGARNET

Copy …Rights

- *This slide set is the ownership of the 6DISS project via its partners*
- *The Powerpoint version of this material may be reused and modified only with written authorization*
- *Using part of this material must mention 6DISS courtesy*
- *PDF files are available from www.6diss.org*

Contributions

- Main authors
  - János Mohácsi, NIIF/HUNGARNET - Hungary

Configuring IPv6 Firewalls with Windows XP SP2

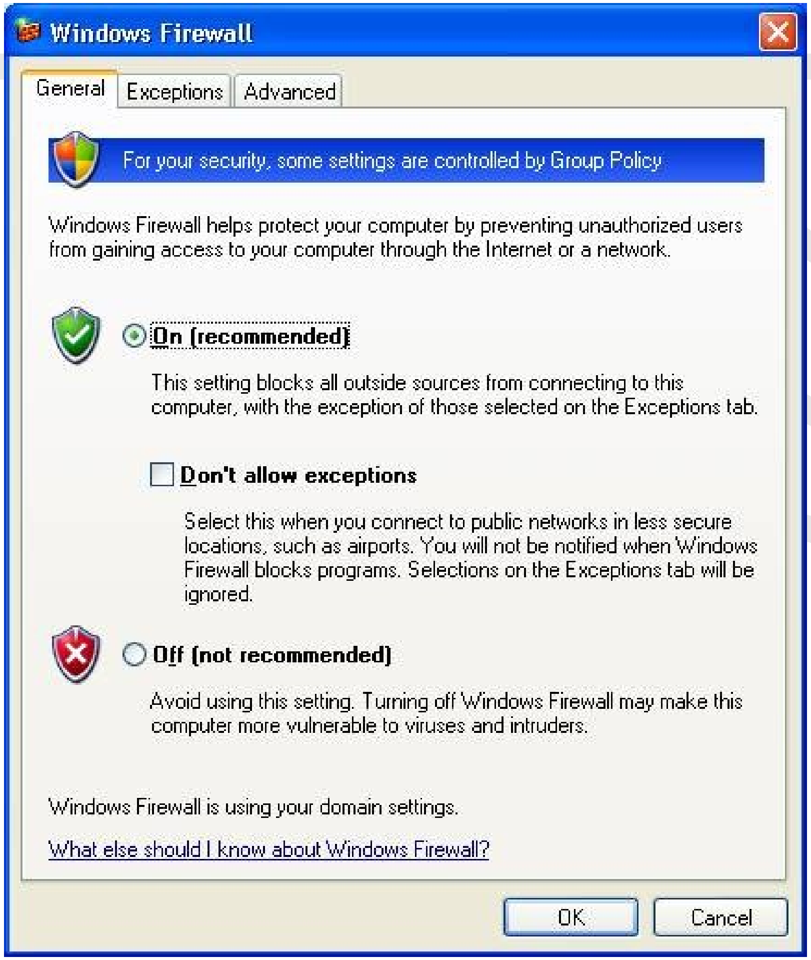
Windows XP Firewall configuration

- Windows XP ICF – same rules for IPv4 and IPv6
  - Show configuration:
    - `netsh firewall show config`
  - Set configuration
    - `set icmpsetting [ type = ] 2-5|8-9|11-13|17|ALL [ [ mode = ] ENABLE|DISABLE [ profile = ] CURRENT|DOMAIN|STANDARD|ALL [ interface = ] name ]`
    - `set opmode [ mode = ] ENABLE|DISABLE [ [ exceptions = ] ENABLE|DISABLE [ profile = ] CURRENT|DOMAIN|STANDARD|ALL [ interface = ] name ]`
    - `add portopening [ protocol = ] TCP|UDP|ALL [ port = ] 1-65535 [ name = ] name [ [ mode = ] ENABLE|DISABLE [ scope = ] ALL|SUBNET|CUSTOM [ addresses = ] addresses [ profile = ] CURRENT|DOMAIN|STANDARD|ALL [ interface = ] name ]`
    - `set logging [filelocation=<location>] [filesize=integer] [droppedpackets=enable|disable] [successfulconnections=enable|disable]`

Windows XP Firewall configuration /2

- After SP2
  - in the firewall you can configure Path MTU discovery support
  - per process configuration possible
  - Can be deployed by group configuration

Windows SP2 ICF



For your security, some settings are controlled by Group Policy

Windows Firewall helps protect your computer by preventing unauthorized users from gaining access to your computer through the Internet or a network.

● On (recommended)

This setting blocks all outside sources from connecting to this computer, with the exception of those selected on the Exceptions tab.

□ Don't allow exceptions

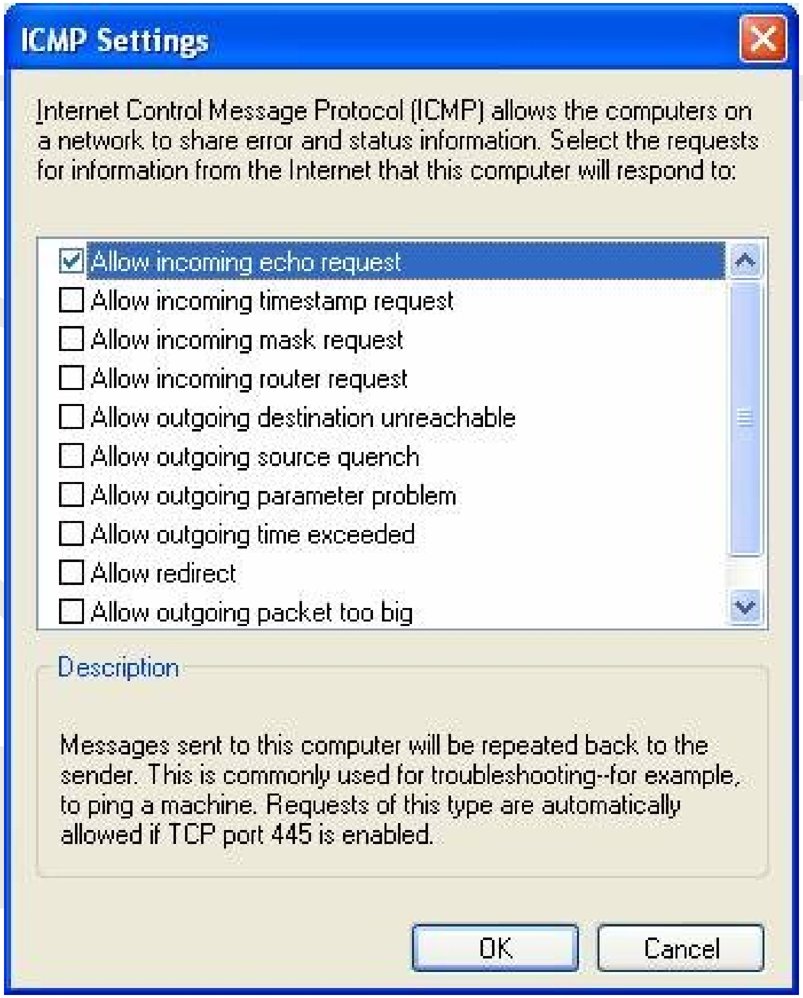
Select this when you connect to public networks in less secure locations, such as airports. You will not be notified when Windows Firewall blocks programs. Selections on the Exceptions tab will be ignored.

○ Off (not recommended)

Avoid using this setting. Turning off Windows Firewall may make this computer more vulnerable to viruses and intruders.

Windows Firewall is using your domain settings.

What else should I know about Windows Firewall?



Internet Control Message Protocol (ICMP) allows the computers on a network to share error and status information. Select the requests for information from the Internet that this computer will respond to:

- ☑ Allow incoming echo request
- □ Allow incoming timestamp request
- □ Allow incoming mask request
- □ Allow incoming router request
- □ Allow outgoing destination unreachable
- □ Allow outgoing source quench
- □ Allow outgoing parameter problem
- □ Allow outgoing time exceeded
- □ Allow redirect
- □ Allow outgoing packet too big

Description

Messages sent to this computer will be repeated back to the sender. This is commonly used for troubleshooting--for example, to ping a machine. Requests of this type are automatically allowed if TCP port 445 is enabled.

Further information

- Further information: http://www.microsoft.com/technet/community/columns/cableguy/cg0204.mspx

Exercises

1. Test ping from neighbouring PC
2. Enable Firewall on Windows XP SP2
3. Test ping from the neighbouring PC
4. Enable ICMP echo + Path MTU discovery to work

Configuring IPv6 Firewalls with pf

János Mohácsi

NIIF/HUNGARNET

What is pf?

- OpenBSD included IPFilter in the default install since 3.0. Included in FreeBSD since 5.3 (5.x as port) and in NetBSD since 2.0.
- Ideas from ipf which is available for Linux, Solaris, HP-UX, IRIX additionally to the operating systems above.
- Principles
  - working on IP packet level (vs. application level proxies or ethernet level bridges)
  - intercepting each IP packet that passes through the kernel (in and out on each interface), passing or blocking it
  - stateless inspection based on fields of each packet
  - stateful filtering keeping track of connections, additional information makes filtering more powerful (sequence number checks) and easier (replies, random client ports)
  - filtering for local host or network (multihomed host, IP forwarding or bridging)

pf filter rules

- linear linked list, evaluated top to bottom for each packet (unlike netfilter's chains tree)
- rules contain parameters that match/mismatch a packet rules pass or block a packet
- last matching rule wins (except for 'quick', which aborts rule evaluation)
- rules can create state, further state matching packets are passed without rule set evaluation

common basic rule syntax

```
pass|block in|out on <int> [<af>] [proto
  <protocol>] from <src_ip> port
  <src_port> to <dst_ip> port <dst_port>
```

<int>: a network interface

<af>: address family: inet or inet6

<protocol: protocol: icmp, tcp, udp, icmp6 etc.

<src_ip>: an IP address (or range)

<src_port>: a TCP or UDP port number

<dst_ip>: an IP address (or range)

<dst_port>: a TCP or UDP port number

| denotes OR, [] denotes optional part

common basic rule syntax/2

```
pass|block in|out on all
```

- apply a rule to all interfaces, or to all sources, and to all destinations, and to all ports the powerful all keyword should be employed:

Small examples

1. block out on tun0 all port 25
2. pass in on fxp0 from 192.168.1.0/24 to 192.168.2.51 port 22
3. pass out on fxp0 from 192.168.2.51 to 192.168.1.0/24 port 22

Common basic rule syntax /3

- Quick
  - Using the quick keyword in a rule commands PF to apply the rule immediately. Further rule processing is abandoned.
- Log
  - The all important logging feature is now in use via the log keyword. Logging is accomplished emitting bpf like log via virtual network inteface pflog0 that can be collected and filtered by by the pflogd daemon. Logging can be applied to any rule.
    - link layer header used for pf related information (rule, action)
    - Easy to process with tcpdump
    - Important: when using log and state keywords in the same rule: only the packet that establishes the state is logged.

State table

- pf can "keep state" or perform "stateful inspection". Stateful inspection refers to PF's ability to track the state, or progress, of a network connection. By storing information about each connection in a state table, PF is able to determine if a packet passing through the firewall belongs to an already established connection. If it does, it is passed through the firewall without going through any ruleset evaluation.
  - TCP (sequence number checks on each packet), ICMP error messages match referred to packet (simplifies rules without breaking PMTU discoveries etc.)
  - UDP, ICMP queries/replies, other protocols: pseudo-connections with timeouts
  - adjustable timeouts (aggressive, normal, high-latency, conservative)
  - binary search tree (AVL, now Red-Black), O(log n) even in worst-case
  - key is two address/port pairs

Ruleset IPv4 1(nothing in, DNS, all TCP, ping out)

```
EXT = “bge0"
LAN = “bge1"
LANip4 = "192.168.1.1“
EXTip4 = “192.168.2.1
LANnet4 = "192.168.1.0/24“
Lo4 = “127.0.0.1”
# expire state connections early
set optimization aggressive
block in log all
# allow DNS requests to go out
pass out on $EXT inet proto udp from {$EXTip4, $Lo4, $LANnet4} to any port=domain keep state
# all TCP request allowed out
pass out on $EXT inet proto tcp from {EXTip4, $Lo4, $LANnet4} to any keep state
# all ping request allowed out
pass out on $EXT inet proto icmp all icmp-type 8 code 0 keep state
# DNS request inside
pass in on $LAN inet proto from $LANnet4 to any port domain
# TCP request inside
pass in on $LAN inet proto tcp from $LANnet4 to any
# ICMP request inside
pass in on $LAN inet proto icmp all icmp-type 8 code 0
```

Antispoofing

- antispoof, parser generates blocking rules appropriate for the specified interfaces

```
antispoof for lo0
   block in on !lo0 inet from 127.0.0.1/8 to any
   block in on !lo0 inet6 from ::1 to any
antispoof for bge1 #inet/inet6
   block in on !bge1 inet from 192.168.1.1/24 to any
   block in inet from 192.168.1.1 to any
   block in on !bge1 inet6 from 2001:db8:1:2::/64 to
     any
   block in on !bge1 inet6 from
     fe80::209:6bff:fe8c:845b to any
   block in inet6 from 2001:db8:1:2::1 to any
```

IPv6 specific rules -reminder

- Neighbor solicitation/neighbor advertisement is REQUIRED – icmp-type 135/136 (ipv6-icmp-type neighbrsol/neighbradv)
- For stateless address autoconfiguration router advertisement and router solicitation REQUIRED– icmp type 133/134 (ipv6-icmp-type routersol/routeradv)
- Path MTU discovery – automatic if you use keep-state
  - Same applies for Destination unreachable, Time exceeded and IPv6 Parameter problem messages
  - Otherwise you have to build your own rules

Supported ICMPv6 types

unreach	1	Destination unreachable
toobig	2	Packet too big
timex	3	Time Exceeded
paramprob	4	Parameter problem
echoreq	128	Echo Request
echorep	129	Echo Reply
groupqry	130	ICMPv6 Membership query
listqry	130	MLD listener query
grouprep	131	ICMPv6 membership report
listenrep	131	MLD listener report
groupterm	132	ICMPv6 membership termination
listendone	132	MLD listener done
routersol	133	ND router solicitation
routeradv	134	ND router advertisement
neighbrsol	135	ND neighbor solicitation
neighbradv	136	ND neighbor advertisement

Supported ICMPv6 types /2

redir	137	ND redirection
routerrenum	138	ICMPv6 router renumbering
wrureq	139	Who are you request
wrurep	140	Who are you reply
fqdnreq	139	ICMPv6 Fully Qualified Domain Name Query
fqdnrep	140	ICMPv6 Fully Qualified Domain Name Reply
nireq	139	Neighbor Information Query
nirep	140	Neighbor Information Reply
mtraceresp	200	MLD Multicast trace response
mtrace	201	MLD Multicast trace

Ruleset IPv6 1(nothing in, DNS, all TCP, ping out)

```
EXT = “bge0"
LAN = “bge1"
LANip6 = “2001:db8:1:1::1“
EXTip6 = “2001:db8:1:2::1”
LANnet6 = “2001:db8:1:1::1/64“
Lo6 = “::1”
# expire state connections early
set optimization aggressive
block in log all
# allow DNS requests to go out
pass out on $EXT inet6 proto udp from {$EXTip6, $Lo6, $LANnet6} to any port=domain
   keep state
# all TCP request allowed out
pass out on $EXT inet6 proto tcp from {EXTip6, $Lo6, $LANnet6} to any keep state
# all ping request allowed out
pass out on $EXT inet6 proto icmp6 all icmp6-type echoreq keep state
# ND solicitation out
pass out on $EXT inet6 proto icmp6 all icmp6-type {neighbradv, neighbrsol}
# ND advertisement in
pass in on $EXT inet6 proto icmp6 all icmp6-type {neighbradv, neighbrsol}
```

Ruleset IPv6 1(continue – with router advertisement from FW)

```
#router advertisement out
pass out on $LAN inet6 proto icmp6 all icmp6-type routersadv
# router solicitation in
pass in on $LAN inet6 proto icmp6 all icmp6-type routerrsol
# DNS request inside
pass in on $LAN inet6 proto from $LANnet6 to any port domain
# TCP request inside
pass in on $LAN inet6 proto tcp from $LANnet6 to any
# ICMP request inside
pass in on $LAN inet6 proto icmp6 all icmp6-type echoreq
```

Ruleset IPv6 2 (allow access to internal mail & www server – additional rules)

```
#internal server address
LANSRV6="2001:db8:1:2::2"
LANSRV4="192.168.1.2"
#allow incoming connection to SMTP server
pass in on $EXT inet6 proto tcp from any to $LANSRV6 port=25 keep-state
pass in on $EXT inet proto tcp from any to $LANSRV4 port=25 keep-state
#all reply from SMTP server (does not really necessary)
pass in on $LAN inet6 proto tcp from $LANSRV6 port=25 to any keep-state
pass in on $LAN inet proto tcp from $LANSRV4 port=25 to any keep-state
#allow incoming connection to WWW server
pass in on $EXT inet6 proto tcp from any to $LANSRV6 port=www keep-state
pass in on $EXT inet proto tcp from any to $LANSRV4 port=www keep-state
#all reply from SMTP server (does not really necessary)
pass in on $LAN inet6 proto tcp from $LANSRV6 port=www to any keep-state
pass in on $LAN inet proto tcp from $LANSRV4 port=www to any keep-state
```

Problems with IPv6

- No fragment normalisation – not possible! – fragmentation only at the end-host
- no real support for extension headers – check existence of extension header possible without chain processing
- IPv6 fragments are blocked unconditionally
- No IPv6 support in ftp-proxy
- No support to filter inside tunnel – except if tunnel terminated at firewall

Tunnel filtering example - simplest

```
pass out on $EXT inet proto ipv6 from
  $EXT to $TUNNELREMOTE4 keep state
pass in  on $EXT inet proto ipv6 from
  $TUNNELREMOTE4 to $ext_if  keep
  state
pass out on gif0 inet6 all keep state
pass in  on gif0 inet6 all keep state
```

More restrictive IPv6 rules

- Allow Rtsol/rtadv on a more specific address

```
pass out on $LAN inet6 proto ipv6-icmp from fe80::/16
  to ff02::2 icmp6-type routersol code 0
pass in  on $LAN inet6 proto ipv6-icmp from fe80::/16
  to ff02::1 icmp6-type routeradv code 0
pass in  on $LAN inet6 proto ipv6-icmp from fe80::/16
  to fe80::/16 icmp6-type routeradv code 0
```

- Allow NDsol/Ndadv on more specific address

```
pass out on $if inet6 proto ipv6-icmp from { ::
  fe80::/16 } to ff02::/16 icmp6-type grouprep code 0
pass out on $if inet6 proto ipv6-icmp from ($if) to
  any icmp6-type neighbrsol code 0
pass in  on $if inet6 proto ipv6-icmp from any to
   ($if) icmp6-type neighbradv code 0
```

Further Information

- Further information
  - The OpenBSD pf FAQ:
    - http://www.openbsd.org/faq/pf/index.html
  - Tons of information:
    - http://www.benzedrine.cz/pf.html
    - mailing list with archive

Demonstration

1. Small set of pf rules for personal firewall:

```
#block everything
block in log all
block out log all
#allow everything for loopback
pass in quick on lo0 all
pass out quick on lo0 all
#allow all outgoing packets
pass out quick proto tcp from $ext_if to any keep state
pass out quick proto udp from $ext_if to any keep state
pass out quick inet proto icmp from $ext_if to any keep state
pass out quick proto ipv6-icmp from any to any keep state
pass in quick proto ipv6-icmp from any to any
pass in quick proto tcp from any to any port = 22
```

Demonstration/2

1. Disable ssh access
2. Disable ICMPv6 packets

Cisco IOS
IPv6 Access Control Lists

Patrick Grossetete

Cisco IOS IPv6 Product Manager

Pgrosset@cisco.com

Cisco IOS IPv6 Standard Access Control Lists

- Cisco IOS IPv6 access-lists are used to filter traffic and restrict access to the router. IPv6 prefix-lists are used to filter routing protocol updates.
- IPv6 Standard ACL (Permit/Deny)
  - IPv6 source/destination addresses
  - IPv6 prefix-lists
  - On Inbound and Outbound interfaces
- Minimum Cisco IOS releases
  - Cisco IOS 12.2(2)T or 12.3(1)M
  - Cisco IOS 12.0(21)ST1 and Cisco 12.0(22)S on Cisco 12000 series only
  - Cisco 12.2(14)S

Cisco IOS IPv6 Extended ACL

- Adds support for IPv6 option header and upper layer filtering
- Only named access-lists are supported for IPv6
- IPv6 and IPv4 ACL functionality
  - Implicit deny any any as final rule in each ACL.
  - A reference to an empty ACL will permit any any.
  - ACLs are NEVER applied to self-originated traffic.
- Minimum Cisco IOS releases
  - Cisco IOS 12.2(13)T or 12.3(1)M
  - Cisco 12.0(23)S on Cisco 12000 series only, 12.0(25)S adds hardware assisted ACL on Engine 3
  - Cisco 12.2(14)S

Cisco IOS IPv6 Extended ACL overview

- CLI mirrors IPv4 extended ACL CLI
- Implicit permit rules, enable neighbor discovery
- ULP, DSCP, flow-label,… matches
- Logging
- Time-based
- Reflexive
- CEFv6 and dCEFv6 ACL feature support
- Extended ACL can apply even if option headers are in a packet

Cisco IOS IPv6 ACL Implicit Rules

- Implicit permit rules, enable neighbor discovery
  - The following implicit rules exist at the end of each IPv6 ACL to allow ICMPv6 neighbor discovery:

    permit icmp any any nd-na

    permit icmp any any nd-ns

    deny ipv6 any any
  - Be careful, when you add "deny ipv6 any any log" at the end

Cisco IOS IPv6 Extended ACL Match

- TCP/UDP/SCTP and ports (eq, lt, gt, neq, range)
- ICMPv6 code and type
- Fragments
- Routing Header
- Undetermined transport
  - The first unknown NH can be matched against (numerically rather than by name).
  - Since an unknown NH cannot be traversed, the ULP cannot be determined.

Cisco IOS IPv6 Extended ACL

- Logging

  -
    ```
    (conf-ipv6-acl)# permit tcp any any log-input
    (conf-ipv6-acl)# permit ipv6 any any log
    ```
- Time based

  -
    ```
    (conf)# time-range bar
    (conf-trange)# periodic daily 10:00 to 13:00
    (conf-trange)# ipv6 access-list tin
    (conf-ipv6-acl)# deny tcp any any eq www time-range bar
    (conf-ipv6-acl)# permit ipv6 any any
    ```

Cisco IOS IPv6 ACL Reflexive

- Reflect
  - A reflexive ACL is created dynamically, when traffic matches a permit entry containing the reflect keyword.
- Evaluate
  - Apply the packet against a reflexive ACL.
  - The implicit deny any any rule does not apply at the end of a reflexive ACL; matching continues after the evaluate in this case.

Cisco IOS IPv6 ACL CLI (1)

- Entering address-family sub-mode
  - [no] ipv6 access-list <name>
  - Add or delete an ACL.
- IPv6 address-family sub-mode
  - [no] permit | deny ipv6 | <protocol> any | host <src> | src/len [sport] any | host <dest> | dest/len [dport] [reflect <name> [timeout <secs>]] [fragments] [routing] [dscp <val>] [flow-label <val>][time-range <name>] [log | log-input] [sequence <num>]
  - Permit or deny rule defining the acl entry. Individual entries can be inserted or removed by specifying the sequence number.
  - Protocol is one of TCP, UDP, SCTP, ICMPv6 or NH value.

Cisco IOS IPv6 ACL CLI (2)

  - [no] evaluate
  - Evaluate the dynamically created acl via the permit reflect keyword.
  - [no] remark
  - User description of an ACL.
- Leaving the sub-mode
  - exit
- Showing the IPv6 ACL configuration
  - # show ipv6 access-list [name]
show access-list [name]
- Clearing the IPv6 ACL match count
  - # clear ipv6 access-list [name]
clear access-list [name]

Cisco IOS IPv6 ACL CLI (3)

- Applying an ACL to an interface
  - (config-int)# ipv6 traffic-filter <acl_name> in | out
- Restricting access to the router
  - (config-access-class)# ipv6 access-class <acl_name> in | out
- Applying an ACL to filter debug traffic
  - (Router)# debug ipv6 packet [access-list <acl_name>] [detail]

Further Information

- Further information:
  - http://www.cisco.com/go/ipv6

ACL Exercises

Exercises: Access Controls

- How to filter, both on router and clients
- First step is to protect router configs (e.g. vty)
- Second router traffic filters c.f. IPv4 access lists
- Finally per-node filtering
- Strategy: Deny 23/tcp at the edge (i.e. workgroup router), then pick other protocols individually on nodes, e.g. http on client 1, ssh on 2, etc.

Exercise 1: ACLs on line vty

- Define an access-class as per IPv4
  - Use symbolic names rather than class indexes of a particular range
- Bind that access-class to a line definition
- Test

Exercise 2: ACLs on forwarding

- ACLs can also be bound to interfaces to impact on the traffic that can be routed through a router.
- Use ACLs to achieve three filters:
  - block SSH connections coming in from two of the other teams, but not all
  - block HTTP access anyone outside of the workshop
  - block all outbound SMTP traffic from a certain ip range

IPSec with OSPFv3 on Cisco routers

Configuring IPSec on OSPF for IPv6

- OSPF for IPv6 configured – define security policy:
  - SPI
  - Key
- In Cisco IOS you can configure OSPFv3 security policy per:
  - area
  - interface

OSPv3 IPSec Authentication configuration

- Defining IPSec Authentication on an Interface

```
interface type number
  ipv6 ospf authentication ipsec
  spi spi md5 [key-encryption-type]
  key | null
```

- Defining Authentication in an OSPF Area

```
ipv6 router ospf process-id
   area area-id authentication ipsec spi
     spi md5 [key-encryption-type] key
```

- Debug

```
show crypto ipsec sa ipv6
```

Configuration steps in MQC

- Define Class Map
  - Separate traffic into classes based on access lists (ACLs), DSCP/ToS, MPLS EXP, protocol, etc. or combinations of those criteria

```
class-map [match-any | match-all] class-name
```

- Define Policy Map (Service Policy)
  - Associate a class map with one or more QoS policies, e.g. bandwidth allocation, queue management, (re)-marking

```
policy-map policy-map-name
```

IPv6 QoS configuration on Cisco routers

Configuration steps in MQC

- Apply a Service Policy to an interface
  - Associate a policy map to an physical or logical interface at input or output.

```
service-policy {input | output} policy-map-name
```

Configuration examples

```
class-map match-any
  ip_premium_out
  match ip dscp 46
  match ip dscp 47
  match ip dscp 40
  match mpls experimental 5

class-map match-any lbe_out
  match ip dscp 8
  match mpls experimental 1
```

IP Premium classification *class-map*

LBE classification *class-map*

Configuration examples

```
policy-map QoS_out
  class ip_premium_out
     priority
  class lbe_out
     bandwidth percent 1
  class class-default
     exit
  exit
```

QoS policy definition *policy-map*

```
interface POS 0/1
  service policy output QoS_out
```

Apply *service policy* to an interface