

# JANET

## IPv6 Hands-on Workshop

### Lab 2: **Services**

UKERNA, Lancaster University  
and University of Southampton, 2006

## Laboratory Overview

- Service provision
  - Web
  - DNS
- Access control lists

These exercises offer an overview of IPv6 in operation from both a network operator's perspective and that of a workgroup administrator. The steps that you will take in this laboratory will be to see the adaptation of key Internet services to support both IPv4 and IPv6.

Places to look when seeking on-line documentation include:

- \* Cisco Software IOS Releases Command References  
<http://www.cisco.com/univercd/cc/td/doc/product/software/>
- Juniper JUNOS Software Documentation  
<http://www.juniper.net/techpubs/software/junos/>

Both of these documents (as well as other resources for this workshop) are linked from  
<http://www.ipv6.org.uk/workshop/>

Note that again you're not necessarily expected to finish everything in this lab - experiment at your own pace.

## Ex1. Services - HTTP

- Apache has been installed on just the linux client, where IPv6 is enabled by default. Under Windows Apache is still not officially capable of IPv6 (requires patching)
- The Listen directive
  - Determines nature of the socket connection
- Visit the IPv6 test page - a waving flag means a connection over IPv6

By default, Apache will Listen on both IPv6 and IPv4 (on Windows, it will be IPv6 only, with no IPv4-mapped addresses). Nothing further is required to make it work. However you can bind to IPv6 only or IPv4 only with the respective listen statements: Listen [::]:80, Listen 0.0.0.0:80.

The configuration has been changed so that Apache is bound to IPv4 only. Open up a browser and visit your local server (homepage in Firefox). Follow the link to test page1. The page should show a Union Jack flag that is NOT waving along with the IPv4 address of your client.

(Use `"/etc/init.d/httpd start"` to start the server if it is not already running).

Modify the configuration within `/etc/httpd/conf/httpd.conf`. Change the listen directive to bind the server to both IPv4 and IPv6. Use `"/etc/init.d/httpd restart"` to restart the server. Revisit the web page - your connection should now be over IPv6, as demonstrated by the waving flag, the client's IPv6 address being shown.

## Ex1a. IPv6 HTTP in Action

- Server-Side Includes (SSIs) can be used to show connection source addresses, and thus whether IPv6 is being used
- Visit the second test page, an .shtml file showing server and client addresses
- Observe the behaviour of other web browsers when accessing this page

In your default instance document root (/var/www/html), there is a second test page. This page is a simple server-side include (SSI) script, test2.shtml with the contents as below.

```
<pre>
SERVER_ADDR  : <!--#echo var="SERVER_ADDR"-->
REMOTE_HOST  : <!--#echo var="REMOTE_HOST"-->
REMOTE_ADDR  : <!--#echo var="REMOTE_ADDR"-->
</pre>
```

Access this page from one of your clients.

What addresses are shown as being used for the connections?

What happens if you use a different browser (e.g. Firefox v. Explorer on Windows, Firefox v. Mozilla on the Linux boxes)?

Try changing the Listen directive again and see what addresses are shown.

## Ex2. Services - DNS

- The Linux node in your cluster is special - it has been allocated as the nameserver for your subnet
- Configure an extra address for this machine
- BIND9 has already been installed, but a few extra lines of configuration are needed to make it IPv6 capable
- Make your other clients in the subnet use this IPv6-speaking NS as their resolver

The DNS server needs to have an additional IPv6 address added. This machine will be the Linux machine. You will need to add the following additional address: 2001:630:81:4X0::53. To do this use:

```
"ip -6 addr add <address> dev eth0"
```

Your next challenge is to add IPv6 and reverse IPv6 zones, and serve IPv6 data for the clients in your team.

## Ex2a. Configuring BIND

- Configure BIND to listen on IPv6 addresses as well as IPv4 (its default)
- Configure named.conf to find the zone file for the delegated domain - the zone file has already been created for you
- Have a look at the structure used and note the similarities between IPv4 and IPv6

By default, Bind will listen on all IPv4 addresses, but will not listen on IPv6. The magic line you want to add within the options section of `/etc/named.conf` is `"listen-on-v6 { any; };"`.

Obviously, "any" can be replaced by specific addresses (for example if you want localhost only). There is an equivalent listen-on directive that takes the same syntax for IPv4 addresses.

**Note: Throughout this exercise, where the BIND configuration examples contain an X, it is in fact your group number.**

BIND is installed in `/var/named`, where `data/` is intended to store zone files for which the server is master. These zones are defined within `/etc/named.conf`. You will need to add IPv6 records to the zone file `"ukwX-6pack-org.db"`, for example:

```
client1      IN      AAAA    2001:0630:0081:0472:0215:c5ff:fe3a:c067
```

You can also add reverse delegations made for the zone `x.4.0.1.8.0.0.3.6.0.1.0.0.2.ip6.arpa`. If you get this far and want to complete the service configuration, you can do so. Uncomment the `"X.4.0.1.8.0.0.3.6.0.1.0.0.2.ip6.arpa"` zone within `named.conf`. Then add the reverse pointers to the zone file `"ukwX-6pack-org-v6ptr.db"`, which has already been created for you. A reverse pointer entry for the following host, `2001:0630:0081:0472:0215:c5ff:fe3a:c067` looks like this:

```
7.6.0.c.a.3.e.f.f.5.c.5.1.2.0.2 PTR    client1
```

Once the DNS server is set up, the client on the other subnet can use it as their resolver.

## Ex2b. DNS IPv6 transport

- Now that your local resolver is listening on IPv6, configure the clients to use it
- Windows does not support IPv6 transport – yet
- Edit `/etc/resolv.conf` on Linux machines to contain the IPv6 name server (remove others)
- Observe results of “dig +trace” to see delegation, comparing the IPv4 (A) and IPv6 (AAAA) outputs
- Use `host` to check the reverse DNS

Windows XP SP2 does not yet support IPv6 DNS transport. It does not have a pretty GUI for configuring IPv6-transport DNS resolvers, however configuration hooks are available when using the `netsh` utility. In the ‘interface ipv6’ context, type “show dns” to list currently configured resolvers. As none have been configured, the list should be empty. As at SP2 DNS resolving over IPv6 is not available - the framework is in place, but the resolvers won’t comply.

`/etc/resolv.conf` on the Linux machines should contain one “nameserver” line listing the IPv6 address of the team’s nameserver, i.e.

```
nameserver 2001:630:81:04X0::53.
```

You can use a packet sniffer to verify the lookups are via IPv6.

The first check is simply to use “host”, “dig” or “nslookup” to confirm that you can still resolve. A good test would be to use dig in +trace mode to look-up a AAAA record for `www.eprints.org`, which is authoritatively served by a dual-stack nameserver. Hopefully at least two hops of your resolution will be over IPv6 transport.

Can you think of what conditions under which they might not be?

## Ex3. Access Controls

- How to filter on the router
- Protect your router configs (e.g. login)
- Router traffic filters c.f. IPv4 access lists
- Experiment with blocking 23/tcp at the edge then pick other protocols individually on nodes, e.g. http on client 1, ssh on 2, etc.

The purpose of this exercise is to demonstrate the similarities between IPv4 and IPv6 access controls from the perspective of JUNOS and node-based packet filtering.

Application-specific ACLs (e.g. Apache or OpenLDAP client controls) are out of scope, but equally as straightforward.

### Ex3a. ACLs on router (login)

- Create a prefix-list as per IPv4
- Create a firewall using the prefix-list
  - Use symbolic filter names
- Bind the filter to the local interface lo0 (applies to all physical interfaces on the router)
- Try blocking access to your router from all other sources bar your workgroup domain

#### CISCO

IOS uses the notion of an 'access-class' to restrict what remote nodes can connect, e.g., using telnet or ssh. The syntax for IPv6 is slightly extended, but essentially the same as IPv4. In the global context, the access-class is defined under the "ipv6 access-list" directive set, with a statement binding that list to the line definition of "ipv6 access-class <name> in".

```
ipv6 access-list locadmin
permit ipv6 2001:630:D0::/52 any
deny ipv6 any any
!
line vty 0 4
access-class 1 in
ipv6 access-class locadmin in
exec-timeout 60 0
password 7 071977404F075924131F0202
login
!
```

#### JUNIPER

JUNOS has the concept of a prefix-list, a collection of network prefixes. These are used when creating match conditions for firewall filters, for example, you would use source-prefix-list if you wanted to filter packets based on their origin. Each filter is comprised of terms, where a term lists one or more match conditions and an action to take if these conditions are met. You can build up complex filters by adding multiple terms. When creating a firewall filter, you specify the family type it belongs to, i.e. inet or inet6, among others. The steps below outline how to configure IPv6 access controls that block out all other hosts from accessing your edge router via ssh.

Define a prefix-list, (it must only contain IPv6 prefixes as it will be used in an IPv6 filter):

From [edit]

```
set policy-options prefix-list local-subnet 2001:630:81:04X0::/60
show policy-options
```

It should look like this:

```
policy-options {
  prefix-list local-subnet {
    2001:630:81:04X0::/60;
  }
}
```

## Ex3a: ACLs (ctd)

- Setting up the ACLs
- Applying the ACL to an interface

Create the IPv6 filter, with match conditions that allow machines, from the prefix used above, to connect to the ssh port of the edge router. The instructions are on the next sheet.

From [edit firewall family inet6 filter **protect-router**]

```
set term permit-ssh from next-header tcp destination-port ssh source-prefix-list local-subnet
set term permit-ssh then accept
```

Block all other connections to the ssh port of the edge router:

From [edit firewall family inet6 filter **protect-router**]

```
set term deny-other-ssh from next-header tcp destination-port ssh
set term deny-other-ssh then count deny-ssh discard
```

Allow all other packets through:

From [edit firewall family inet6 filter **protect-router**]

```
set term else-permit then accept
```

Lastly apply the filter to an interface. The interface lo0 is special, any filters applied to this interface will apply to all the router's physical interfaces. You also have to specify input or output, depending of whether the packets you wish to filter are being received or transmitted.

From [edit]

```
set interfaces lo0 unit 0 family inet6 filter input protect-router
```

## Ex3a: ACLs (ctd)

- Looking at the JUNOS firewall configuration
  - From [edit] run “show firewall”

```
family inet6 {
  filter protect-router {
    term permit-ssh {
      from {
        source-prefix-list {
          local-subnet;
        }
        protocol tcp;
        destination-port ssh;
      }
      then accept;
    }
    term permit-ssh {
      from {
        protocol tcp;
        destination-port ssh;
      }
      then {
        count deny-ssh;
        discard;
      }
    }
    term else-permit {
      then accept;
    }
  }
}
```

## Ex3b. ACLs on the router (routes)

- In the same manner as before, create two additional filters, to:
  - block HTTP access from other groups to your Linux client subnet and then see if they can still access your http server.
- Be sure to apply the filters to the correct physical interfaces

Use the very same syntax to create another filter, this time to block access to your http server from the other groups but not your own and not any external connections.

Once you're satisfied that the filters are all working, and you understand the system, remove the filters from the interfaces by entering the interface definition context and deleting the filters like so:

From [edit]

```
edit interface lo0 unit 0
delete family inet6 filter input protect-router
```

You can check that they are gone by leaving the configuration mode and typing:

```
show configuration interfaces
```

The actual filters themselves, defined within the firewall context, can remain there if you so wish. They will have no effect so long as they are not applied to any interfaces.

## Lab Summary

- IPv6 Apache web server
- IPv6 DNS and BIND 9
- ACLs and firewall filters
  
- Next: Mobile IPv6 theory
- Tomorrow: Routing, including IS-IS and BGP