Nokia for Business

White Paper:

IPv6 How-To

May 2005

NOKIA
CONNECTING PEOPLE

Setup (Figure 1)

Line Color	Meaning
Blue	IPv4
Green	IPv6
Yellow	6-over-4 tunnels
Red	Physical connection with IPv4 and IPv6

Table 1: Network Line Colors and Meanings

Host(s)	Description
Whiterabbit	Running Red Hat Enterprise Linux 3 with Check Point Provider-1 and has one ethernet card.
Cheshire	Running Microsoft Windows 2000 Server with Service Pack 4 and has two ethernet cards.
Doormouse	Running Red Hat Enterprise Linux 3 and has one ethernet card.
Madhatter	Nokia IP330 security appliance running Nokia IPSO 3.8.1 B028 and Check Point NGFP4 R55p with an IPv6 license (standalone installation).
Redqueen Old110	Nokia IP110 security appliances running Nokia IPSO 3.8.1 B028.
Tweedledee Tweedledum	Nokia IP120 security appliances running Nokia IPSO 3.8 B031 and Check Point NGFP4 R55p.
NSAS	Nokia IP440 security appliance running Nokia IPSO 3.7 B044 and Nokia Secure Access Systems 3.1.0.

Table 2: Host Descriptions

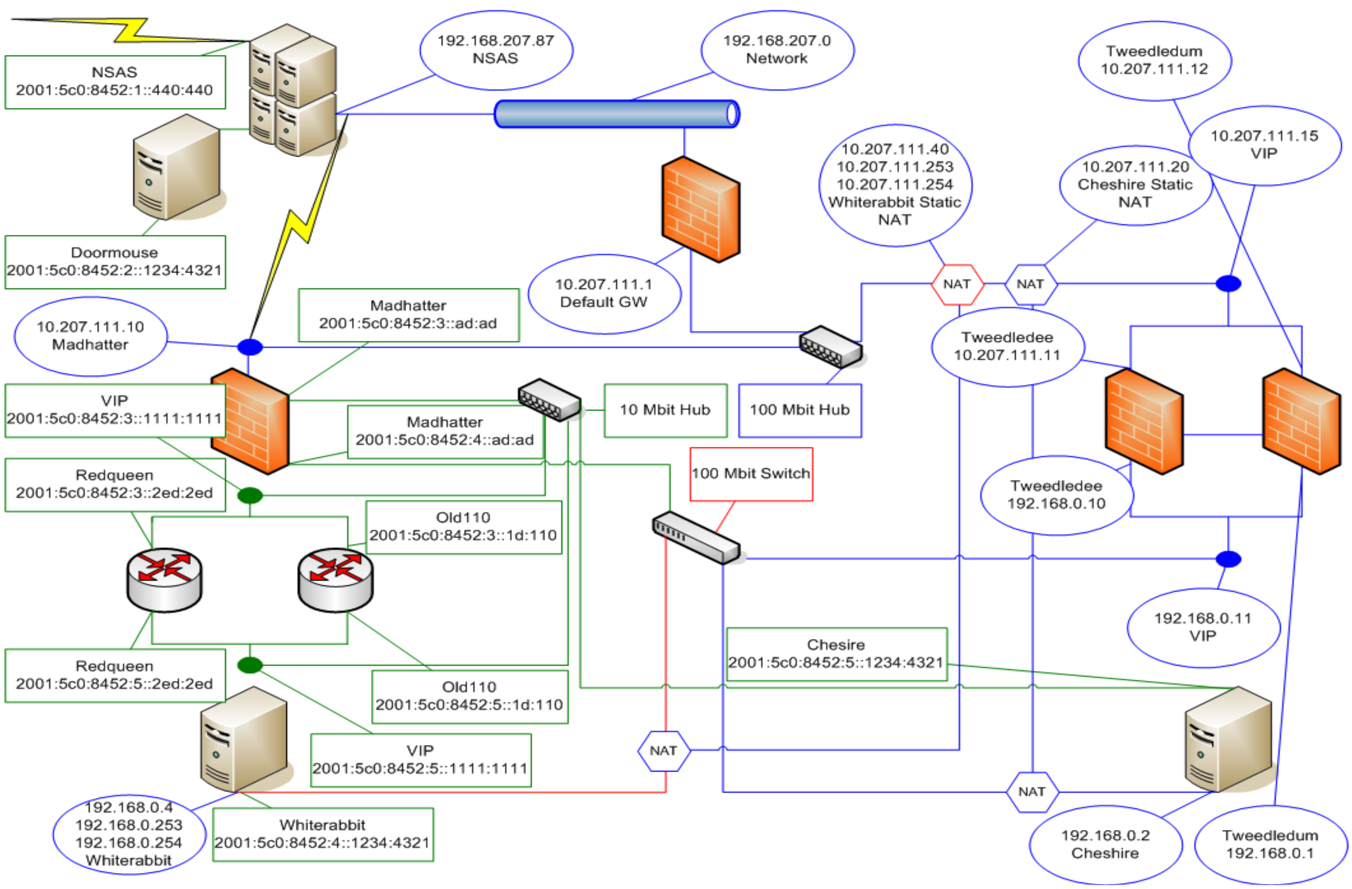


Figure 1: Setup

NOKIA
CONNECTING PEOPLE

Abstract

This document is intended to assist people with implementation of IPv6 with Nokia IP Appliances, Check Point FireWall-1, Linux and Windows. The ultimate goal of this is to get the reader started with IPv6, setting up a network and examining the relations between all the components.

Theory

This section covers some of the basic theory of IPv6, the differences between v4 and v6, as well as some interesting facts.

Why IPv6

We're running out of IP addresses in the world - that's a fact. IPv4 *was* intended to be able to give everyone a live IP address. This, of course, is not the case.

We started running out of live addresses to give everyone so we introduced Network Address Translation (NAT) to let people hide multiple private IP addresses behind one live IP address. Although most people would consider this a viable solution, it's really more of a stop-gap than a full blown solution. Our "everyone has a live IP" model is now broken.

Bob Hinden and Steven Deering created RFC 2460 which is the base model for IPv6 and a lot has been expanded in the time from when they created the RFC to IPv6 implementation today.

Let's start with making correlations between IPv4 and IPv6 so the migration won't be as hard.

Examples	IPv4	IPv6
IP Address (example)	10.20.30.40	fec0:c0ff:ee::01
Subnet Masks	/8 ~ /32	/3 ~ /128
who-has	ARP	Neighbor Solicitation
Address Ranges (total)	4,228,250,625	3.40E+038

Table 3: Examples of IPv4 vs. IPv6

The most noticeable difference is the large amount of addresses available to IPv6 over IPv4. Again, those are just a few examples to whet your appetite for knowledge.

Just like it's popular predecessor, IPv4, IPv6 still uses IP addresses, subnet masks and other similar utilities as well as routing protocols (which we'll take a look at later). Let's start with IP addresses as they will be most prevalent in our work.

IP Addresses

With IPv6, we move from 32-bit addressing to 128-bit addressing. As such, we need a new way to define our IP addresses as decimal just doesn't cut it. So, we use hexadecimal instead. Hexadecimal addresses start at *0* and move to through to the letter *f*. Here's our chart:

Decimal	Hexadecimal
0	0
...	...
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Table 4: Decimal and Hexadecimal Equivalents

That's great, but how about the actual structuring of the address? Here's an expanded example from RFC 2732:

FEDC:BA98:7654:3210:FEDC:BA98:7654:3210

This is a fully expanded IPv6 IP address. So, why do I keep using the word *expanded*? Check this out:

If you have an IP address of, say:

FEC0:C0FF:EE01:0000:0000:0000:0000:0001

you can omit the groups of zeros to read:

FEC0:C0FF:EE01::1

Notice the double colon in place of the groups of zeros from the previous IP address. There is one limitation to this trick which is that you can only do this **once** per IP address. At no one point in any given IPv6 IP address will you see two sets of double colon's. Also, preceding zeros can be omitted as long as nothing else precedes them within the same block.

IP Prefixes

Now, just like IPv4, we can't just start picking out IP addresses to use wherever we want because of conflicts, subnets and other such obstacles. To be able to choose our IP addresses, we need to examine the prefixes available to us and select the one(s) which will correspond to our needs. What exactly does that mean? Well, for instance, you wouldn't use 10/8 on a routable network as that network is intended for Hide-NAT use only

Prefixes are the first part of the IPv6 address and tells us a lot about the IP address in question. These are similar to the first octet of an IPv4 address. For instance, if your IPv4 address starts with 224.x.y.z, you know it's a multicast IP address. Or if it starts with 127.x.y.z, you know it's a loopback. Let's check out some of the prefixes we will come across in our journey. **This is not a complete list of all prefixes!**

Prefix	Description	IPv4 version (or similar)
fe80 ~ febf	Link-Local Address. Packet will never leave the router.	None.
fec0 ~ feff	Site-Local Address. Private Range IP addresses.	10.0.0.0/8 192.168.0.0/16
2001	Global-Unicast (Live IP Address)	1.2.3.4
ffxy (Where *xy* is a number)	Multicast.	224.0.0.0/8
3ffe	6bone address.	None.
::ffff:*w.x.y.z*	IPv4 Compatible Address.	Native.

Table 5: Sample of some IPv6 Prefixes

Now that's interesting. We're starting to understand how IP addresses are formed and we have these cool things called prefixes which tell us what kind of IP address we're dealing with.

IP Subnets

Before going on to the next section, one quick note about subnets. In the IPv6 world, we use the slash notation for our subnet designators. You could use the dotted notation (there's nothing wrong with that) if you wanted to; however, if you're not a masochist, I would avoid it as your hand will likely cramp before you reach a broadcast address. Here are some examples of prefix subnets:

A live, assigned IP Address

2001:5c0:8452::/48 –

A Site-Local (Hide NAT) Address

fec0::/16 –

/111 could be dotted to

ffff:ffff:ffff:ffff:ffff:ffff:fffe:0

Implementation

Now that we have a basic understanding of what IPv6 looks like, let's work on getting your network setup. First, sit down and plan your network. It doesn't have to be fancy, but you should know what you want to accomplish with your setup.

Now comes the not so exciting part...

Setup

1. Install your operating systems, hook up the cables, install your patches and ensure basic IPv4 connectivity. I am going to place two limitations on you at this point:

   Do **not** install Check Point FireWall-1 on your Nokia Appliance just yet. We'll get to that later.

   If you are using Windows XP, do **not** install SP2

   Okay, so now you've got Whiterabbit and Cheshire setup with Tweedledee and Tweedledum setup in a VRRP pair. (Note : You should not use Provider-1 for IPv6 and Check Point).

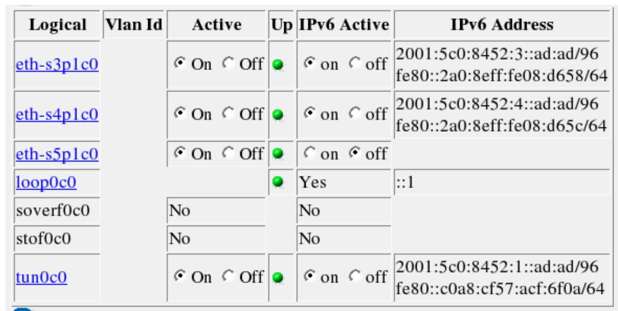
2. Install your management station on one of these two computers
3. Create your Check Point rulebase as you would like them. Again, for this environment, you have to statically NAT your management station so that you can successfully push policy to Madhatter.
4. Test your IPv4 connection:
    a. Can you browse the Internet?
    b. Is Check Point working properly for logging?
5. Give it the once-over and make sure. Go ahead; I'll wait.

Oh, you're back already; That was fast. Now, let's do some IPv6'ing which is why you're here, right? Madhatter, in my case, is running Nokia IPSO 3.8.1b028 with Check Point NGFP4 R55p. As a limitation, Check Point does not support control connections over IPv6 which is why we need the IPv4 address on the external and also why we statically NAT'd our management station.

Configuring Nokia IPSO

First thing's first...

1. Open your favourite web-browser and connect to Madhatter
2. Logging in as Admin. Go into the IPv6 configuration area and Logical Interfaces.
3. Turn on, and Activate the interfaces you want to use for IPv6.
4. Open their respective Logical Interface and assign them their proper IPv6 addresses. If you are setting this up for an "internal only" network, you should use Site-Local addresses (fec0~feff) – If your ISP supports IPv6 or if you are otherwise setting up a "live" network, you should use your assigned Global Unicast IP address.



Logical	Vlan Id	Active	Up	IPv6 Active	IPv6 Address
eth-s3p1c0		(•) On ( ) Off	●	(•) on ( ) off	2001:5c0:8452:3::ad:ad/96 fe80::2a0:8eff:fe08:d658/64
eth-s4p1c0		(•) On ( ) Off	●	(•) on ( ) off	2001:5c0:8452:4::ad:ad/96 fe80::2a0:8eff:fe08:d65c/64
eth-s5p1c0		(•) On ( ) Off	●	( ) on (•) off	
loop0c0			●	Yes	::1
soverf0c0		No		No	
stof0c0		No		No	
tun0c0		(•) On ( ) Off	●	(•) on ( ) off	2001:5c0:8452:1::ad:ad/96 fe80::c0a8:cf57:acf:6f0a/64

Figure 2: IPv6 Configuration and Logical Interfaces

This is what you should see once you have configured your interfaces, IP addresses and subnet masks on your Nokia appliance. For the time being, just ignore the tun0c0 interface because it's special and will have it's own section later on in the document.

Next, we need to ensure that everyone can talk to one another and, for that, we need Neighbor Discovery.

1. Go back to the main IPv6 configuration page and select this option.
2. Under Global Neighbor Discovery Settings:
    a. Queue Limit: 1
    b. Unicast Retry Limit: 3
    c. Multicast Retry Limit: 4
    d. Duplicate Address Detection Retry Limit: 3

Figure 3: Neighbor Discovery

This ensures that when the interface comes online, it will send out a total of three retry detections for Unicast addressing as well as Multicast and DAD. Let's take a closer look at DAD, shall we?

Duplicate Address Detection, or DAD, is IPv6's way of checking for duplicate Link-Local addresses. As we know, Link-Local addressing is also known as Stateless Auto configuration, kinda' like DHCP but not, and for this to work, we need to ensure that there are no other Link-Local addresses the same as ours. If there are, we need to be manually configured.

Once we have this setup, we should look at routing.

1. Go into the Static Routes area for IPv6.
2. Add any Static Routes and (if possible at this time) a default gateway for your Nokia appliance.

If you want to use a dynamic routing protocol, we can set that up now as well. I am using RIP for my IPv6 hosts and routers to learn and update their routing tables. RIP is an older protocol, a little slower but very easy to configure and understand. Here's a quick rundown on how it works: Every route has a metric from 1 to 16 where 16 is a dead route, timed out or otherwise not used. Every hop a RIP packet takes will add 1 to the effective metric to propagate routes up to 16 where it stops being used.

Turn RIPng on for the interfaces you have configured. From here, all we have to do is add a metric for the route propagation as the default (for some reason) is 0 which means that these routes will never be used. Add a 1 for the metrics, Apply and Save your changes. Done and done. RIPng will be examined in more detail later on through packet captures.

Great... Now Madhatter is setup but how are we supposed to configure the Redqueen and Old_110? If you are installing from the Boot Manager, you will need an IPv4 FTP server for access to the Nokia IPSO.tgz file.

1. Once complete, reboot the machine, give it a hostname, however, when asked how you would like to configure it, select VT-100 browser using Lynx.
2. Wait until you have the Login: prompt for Nokia IPSO and login as Admin.
3. Once in, run the command lynx to start the text-only browser.
4. Use the arrow keys to navigate, space to go to the next page and [ENTER] to toggle switches and radio buttons. Make your way to the IPv6 configuration area and configure an interface with Neighbor Discovery.
5. Once this is complete, we can now do the rest through Voyager by connecting from Whiterabbit (configured below) with Mozilla Firefox.

Configuring Linux

Log into your Linux machine as root. If you don't have IPv6 statically built into the kernel, you will need to load the module.

1. At the prompt, type *insmod ipv6*.
2. Once done, run *ifconfig -a* to get the logical listings for your ethernet devices (Mine only has eth0 as a real network device).
3. The command *ifconfig eth0 add 2001:5c0:8452:4::1234:4321/96* will add the IP address 2001:5c0:8452:4::1234:4321 with a subnet length of /96 to the eth0 device.
4. Once this has been entered, run *ifconfig eth0* and you should now see your IPv6 information listed here.

If RIPng doesn't propagate to your Linux machine automatically, or if you want to add any Static Routes (or a default gateway), you can use the following: *route -A inet6 add default gw 2001:5c0:8452:4::ad:ad* which will add a default gateway pointing to the directly connected interface of Madhatter.

Configuring Windows

On your Windows machine, double-check to ensure that the only network adapter with IPv6 checked in is the one you are going to use for IPv6. For instance, I have two Ethernet adapters in my Windows 2000 machine; One is strictly for IPv4 and the other is for IPv6. Next, make sure you have the IPv6 developer pack for Windows.

1. Go into your Network Control Panel and rename them appropriately (I called mine IPv4 and IPv6, go figure...).
2. Then, go into the one called IPv4, and uncheck the IPv6 box.
3. Open up a command prompt and type *ipv6 if* which will give you a listing of your IPv6 interfaces.
4. Look for the logical number for your Local Area Connection corresponding to your IPv6 NIC (For this example, mine is 5).
5. Run *ipv6 adu 5/2001:5c0:8452:5::1234:4321* and hit Enter. This tells Windows that you want to configure an IPv6 address on interface 5 and the IP address appended to the end of it.

If RIPng doesn't propagate to your Windows machine automatically, or if you want to add any Static Routes (or a default gateway), you can use the following: *ipv6 rtu 4/2001:5c0:8452:5::1111:1111* which will add a default gateway pointing to the directly connected Virtual interface of the VRRP pair of Redqueen and Old_110.

Traffic Captures

Now that we have our network setup, let's take a closer look at IPv6 packets and what they have inside of them.

Starting off nice and easy, we'll examine ICMP Echo Requests and Echo Replies before moving on.

```
⊟ Frame 5 (118 bytes on wire, 118 bytes captured)
    Arrival Time: Apr  8, 2005 12:28:53.603900000
    Time delta from previous packet: 0.189157000 seconds
    Time since reference or first frame: 0.192588000 seconds
    Frame Number: 5
    Packet Length: 118 bytes
    Capture Length: 118 bytes
⊟ Ethernet II, Src: 00:a0:8e:08:d6:58, Dst: 00:a0:8e:20:17:ce
    Destination: 00:a0:8e:20:17:ce (NokiaInt_20:17:ce)
    Source: 00:a0:8e:08:d6:58 (NokiaInt_08:d6:58)
    Type: IPv6 (0x86dd)
⊟ Internet Protocol Version 6
    Version: 6
    Traffic class: 0x00
    Flowlabel: 0x00000
    Payload length: 64
    Next header: ICMPv6 (0x3a)
    Hop limit: 63
    Source address: 2001:5c0:8452:4::1234:4321
    Destination address: 2001:5c0:8452:5::1234:4321
⊟ Internet Control Message Protocol v6
    Type: 128 (Echo request)
    Code: 0
    Checksum: 0xf33a (correct)
    ID: 0x320f
    Sequence: 0x0003
    Data (56 bytes)
```

Figure 4: ICMP Echo Request

Here we have ICMP(128) which is an Echo Request sent from Whiterabbit to Cheshire. The main two areas we want to examine are Layers three and four. Layer three shows us our Source and Destination IP addresses, the Hop Limit for the packet and the IP version we are using. Inside Layer four we can see the ICMPv6 code for the request.

```
⊟ Frame 5 (118 bytes on wire, 118 bytes captured)
    Arrival Time: Apr  8, 2005 12:28:53.603900000
    Time delta from previous packet: 0.189157000 seconds
    Time since reference or first frame: 0.192588000 seconds
    Frame Number: 5
    Packet Length: 118 bytes
    Capture Length: 118 bytes
⊟ Ethernet II, Src: 00:a0:8e:08:d6:58, Dst: 00:a0:8e:20:17:ce
    Destination: 00:a0:8e:20:17:ce (NokiaInt_20:17:ce)
    Source: 00:a0:8e:08:d6:58 (NokiaInt_08:d6:58)
    Type: IPv6 (0x86dd)
⊟ Internet Protocol Version 6
    Version: 6
    Traffic class: 0x00
    Flowlabel: 0x00000
    Payload length: 64
    Next header: ICMPv6 (0x3a)
    Hop limit: 63
    Source address: 2001:5c0:8452:4::1234:4321
    Destination address: 2001:5c0:8452:5::1234:4321
⊟ Internet Control Message Protocol v6
    Type: 128 (Echo request)
    Code: 0
    Checksum: 0xf33a (correct)
    ID: 0x320f
    Sequence: 0x0003
    Data (56 bytes)
```

Figure 5: ICMP Echo Reply

As should be expected, here we see the ICMP Echo Reply. The Source and Destination addresses have reversed (as they should) and the ICMPv6 code changed to 129. This may not be the most exciting but it helps to see how things are working on a low-level.

```
⊟ Frame 5 (118 bytes on wire, 118 bytes captured)
    Arrival Time: Apr  8, 2005 12:28:53.603900000
    Time delta from previous packet: 0.189157000 seconds
    Time since reference or first frame: 0.192588000 seconds
    Frame Number: 5
    Packet Length: 118 bytes
    Capture Length: 118 bytes
⊟ Ethernet II, Src: 00:a0:8e:08:d6:58, Dst: 00:a0:8e:20:17:ce
    Destination: 00:a0:8e:20:17:ce (NokiaInt_20:17:ce)
    Source: 00:a0:8e:08:d6:58 (NokiaInt_08:d6:58)
    Type: IPv6 (0x86dd)
⊟ Internet Protocol Version 6
    Version: 6
    Traffic class: 0x00
    Flowlabel: 0x00000
    Payload length: 64
    Next header: ICMPv6 (0x3a)
    Hop limit: 63
    Source address: 2001:5c0:8452:4::1234:4321
    Destination address: 2001:5c0:8452:5::1234:4321
⊟ Internet Control Message Protocol v6
    Type: 128 (Echo request)
    Code: 0
    Checksum: 0xf33a (correct)
    ID: 0x320f
    Sequence: 0x0003
    Data (56 bytes)
```

Figure 6: Neighbor Solicitation

This packet, which is also an ICMPv6 packet, is from Cheshire's Link-Local address (fe80::260:f8ff:fe01:c0) soliciting for a neighbour. This traffic (and subsequently Neighbor Discovery) is handled at the

Link-Local level instead of the Site-Local, Global Unicast or others because neighbors are just that: Neighbors on the same network. In the event we were using Link-Local addresses only for an ad-hoc network, we want to ensure that we can find out who's beside us. If we're using any addresses above Link-Local, any Neighbor Solicitation/Discovery will just be dealt with at the Link-Local level.

```
⊟ Frame 16 (86 bytes on wire, 86 bytes captured)
    Arrival Time: Apr  8, 2005 12:28:55.156801000
    Time delta from previous packet: 0.000603000 seconds
    Time since reference or first frame: 1.745489000 seconds
    Frame Number: 16
    Packet Length: 86 bytes
    Capture Length: 86 bytes
⊟ Ethernet II, Src: 00:a0:8e:20:21:69, Dst: 00:60:f8:01:00:c0
    Destination: 00:60:f8:01:00:c0 (LoranInt_01:00:c0)
    Source: 00:a0:8e:20:21:69 (NokiaInt_20:21:69)
    Type: IPv6 (0x86dd)
⊟ Internet Protocol Version 6
    Version: 6
    Traffic class: 0x0f
    Flowlabel: 0x00000
    Payload length: 32
    Next header: ICMPv6 (0x3a)
    Hop limit: 255
    Source address: fe80::2a0:8eff:fe20:2169
    Destination address: fe80::260:f8ff:fe01:c0
⊟ Internet Control Message Protocol v6
    Type: 136 (Neighbor advertisement)
    Code: 0
    Checksum: 0xe711 (correct)
  ⊟ Flags: 0xe0000000
      1... .... .... .... .... .... .... .... = Router
      .1.. .... .... .... .... .... .... .... = Solicited
      ..1. .... .... .... .... .... .... .... = Override
    Target: fe80::2a0:8eff:fe20:17ff
  ⊟ ICMPv6 options
      Type: 2 (Target link-layer address)
      Length: 8 bytes (1)
      Link-layer address: 00:00:5e:00:02:03
```

Figure 7: Neighbor Advertisement

This is ICMP/136 and now we can see some of the flags that are set. This is a Solicited Router which will force the receiver to update any cached Link-Layer addresses by setting the Override flag.

Remember how I said that we would cover RIPng when we got to packet captures? Well, guess what's on the next page? RIPng... In fact, a really big picture of RIPng.

```
⊞ Frame 1 (186 bytes on wire, 186 bytes captured)
⊞ Ethernet II, Src: 00:a0:8e:20:21:67, Dst: 33:33:00:00:00:09
⊟ Internet Protocol Version 6
    Version: 6
    Traffic class: 0xc0
    Flowlabel: 0x00000
    Payload length: 132
    Next header: UDP (0x11)
    Hop limit: 255
    Source address: fe80::2a0:8eff:fe20:2167
    Destination address: ff02::9
⊟ User Datagram Protocol, Src Port: 521 (521), Dst Port: 521 (521)
    Source port: 521 (521)
    Destination port: 521 (521)
    Length: 132
    Checksum: 0x1798 (correct)
⊟ RIPng
    Command: Response (2)
    Version: 1
  ⊟ IP Address: 2001:5c0:8452:1::/96, Metric: 4
      IP Address: 2001:5c0:8452:1::
      Tag: 0x0000
      Prefix length: 96
      Metric: 4
  ⊟ IP Address: 2001:5c0:8452:4::/96, Metric: 4
      IP Address: 2001:5c0:8452:4::
      Tag: 0x0000
      Prefix length: 96
      Metric: 4
  ⊟ IP Address: 2001:5c0:8452:2::/96, Metric: 6
      IP Address: 2001:5c0:8452:2::
      Tag: 0x0000
      Prefix length: 96
      Metric: 6
  ⊟ IP Address: 2001:5c0:8452:5::/96, Metric: 2
      IP Address: 2001:5c0:8452:5::
      Tag: 0x0000
      Prefix length: 96
      Metric: 2
  ⊟ IP Address: 2001:5c0:8452:3::/96, Metric: 2
      IP Address: 2001:5c0:8452:3::
      Tag: 0x0000
      Prefix length: 96
      Metric: 2
  ⊟ IP Address: ::/0, Metric: 2
      IP Address: ::
      Tag: 0x0000
      Prefix length: 0
      Metric: 2
```

Figure 8: UDP/521 - RIPng

In the packet above, we can see UDP/521 being sent to a weird looking address of ff02::9. If we reference the little chart I made about prefixes in the first part of the document, we know that this is a multicast address. The address we are seeing is all-routers (ff02::) and the host-bit identifier of 9 indicates RIPng routers. Each network listed has a metric assigned to it. When we setup RIPng, we started with a metric of 1 and, for every hop, we add another number.

Check Point FireWall-1

I won't go through the motions of step-by-step'ing you through the installation of Check Point FireWall-1. I will, however, point out that you need to ensure you have an IPv6 license so that you can create and modify IPv6 objects. I'll let you do all of that now.

Once installed and licensed properly, let's start by creating objects for all of our networks and IP addresses. Use the image below to reference my lab setup...

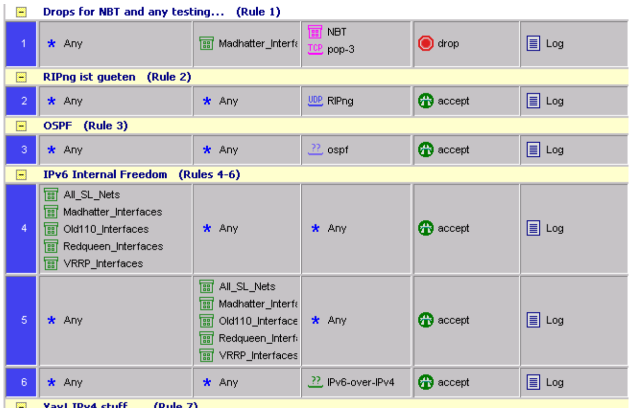


Figure 9: Check Point firewall rules

Since this is in a controlled lab environment, my security is rather lax on this specific firewall. The first thing I did was create network objects for all of my IPv6 subnets being used. You can see what it looks like below...

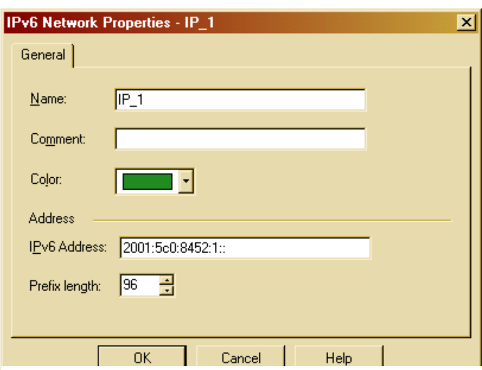


Figure 10: IPv6 Network Properties

This is just like setting an IPv4 network in the sense that you do not set the host-bit identifier and we have a subnet (Prefix length in this case). Next on the list is creating host objects. The **important** thing to remember is that we have two IP addresses (at a minimum) for each host: Link-Local and (other).

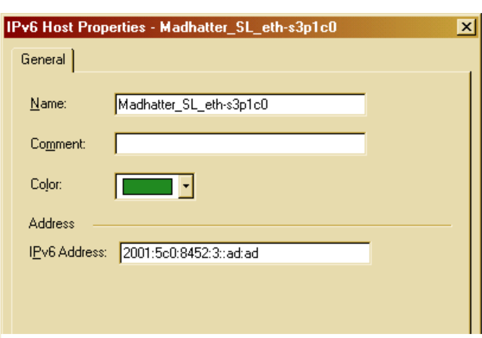


Figure 11: IPv6 Host (Interface) Properties

Here you can see my Global Unicast address for eth-s3p1c0 on Madhatter. Although it's not pictured here, I have also created a Link-Local address object for all of my interfaces and grouped them together.

These objects can be used in the rulebase just like any other objects you would normally place in a rulebase. There are some limitations with Check Point and IPv6:

- Currently, your topology cannot have IPv6 in it. Therefore, IPv6 and Anti-Spoofing don't work together.
- NAT-PT is not supported. Honestly, with all the IPv6 addresses out there, who's going to need it?
- Rules must only have IPv6 **or** IPv4 only in the rules. For instance, you can't mix an object this is IPv4 with another that is IPv6 in the same rule. Why? I don't know.

SmartView Tracker

You can also use SmartView Tracker to see any logs for IPv6. Take a look at the following screenshot:

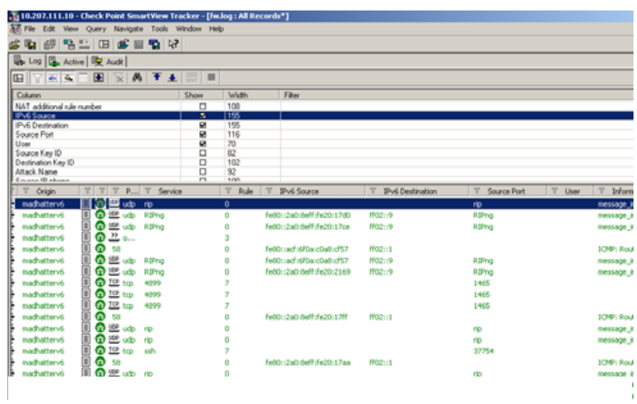


Figure 12: SmartView Tracker

In the top frame, you have to select the IPv6 Source and Destination as it is not enabled by default. After that, it's just like looking at IPv4 logs but with IPv6 traffic. As a side note, IP/58 going to the multicast of ff02::1 are Router Solicitation and Discovery packets being used for Multicast Listener Discovery (MLD) so that hosts wishing to receive multicast are able to do so.

State Table Information

In all honesty, I haven't got this all figured out but here is what I **do** know. ^_^

I ran a program to generate a SYN/SYN-ACK/ACK connection:

From: [WhiteRabbit]

To [Tupac-Amaru {tun0c1:GU}]

Once this connection was established, I ran "fw6 tab -t ipv6_conversion_table -u" on [Madhatter] to examine the traffic. Please note that LL=Link-Local, GU=Global Unicast, (MH)=Madhatter, (TA)=Tupac-Amaru, (WR)=WhiteRabbit, (C)=Connected directly, (O1)=Old_110 and (RQ)=RedQueen. You can safely ignore any and all (O1) and (RQ) entries as they are just propagated from RIPng on my other networks.

```
[root@madhatterv6 log]# fw6 tab -t ipv6_conversion_table -u
localhost:
-------- ipv6_conversion_table --------
dynamic, id 8119, attributes: keep, sync, expires 1, limit 50000, hashsize 32768
, free function 971800f4 0
<000080fe, 00000000, ff8ea002, 672120fe; 00000001; 3136/3600>      # LL of eth-s1p1c0 (RQ)
<17230573> -> <c0050120, 01005284, 00000000, ad00ad00> (00000000)  # GU of tun0c0 (MH)
<7f000001> -> <00000000, 00000000, 00000000, 00000000> (00000000)  # No idea // RIPng (?)
<1e130a55> -> <000080fe, 00000000, 57cfa8c0, 0a6fcf0a> (00000000)  # LL of tun0c0 (MH)
<1bada925> -> <000080fe, 00000000, 0a6fcf0a, 57cfa8c0> (00000000)  # LL of tun0c0 (TA)
<000002ff, 00000000, 00000000, 09000000; 00000008; 1996/3600>      # RIPng (ff02::9)
<14d3b435> -> <000080fe, 00000000, ff8ea002, ce1720fe> (00000000)  # LL of eth-s1p1c0 (O1)
<14d3b435> -> <000080fe, 00000000, ff8ea002, ce1720fe> (00000000)  # LL of eth-s1p1c0 (O1)
<c0050120, 04005284, 00000000, 21433412; 00000002; 3590/3600>      # GU of eth0 (WR) // SRC IP
<12b55ad7> -> <000080fe, 00000000, ff8ea002, 692120fe> (00000000)  # LL of eth-s3p1c0 (RQ)
<1609d791> -> <c0050120, 01005284, 00000000, 40044004> (00000000)  # GU of tun0c0 (TA) // DST IP
<c0050120, 01005284, 00000000, ad00ad00; 7fffffff>                 # GU of tun0c0 (MH)
<c0050120, 03005284, 00000000, ad00ad00; 7fffffff>                 # GU of eth-s3p1c0 (MH)
<c0050120, 01005284, 00000000, 40044004; 00000002; 3590/3600>      # GU of tun0c0 (TA) // DST IP
<000080fe, 00000000, ff8ea002, 5cd608fe; 7fffffff>                 # LL of eth-s4p1c0 (MH)
<00000000, 00000000, 00000000, 00000000; 7fffffff>                 # No idea // RIPng (?)
<10000001> -> <000080fe, 00000000, ff8ea002, 58d608fe> (00000000)  # LL of eth-s3p1c0 (MH)
<10010dc7> -> <c0050120, 03005284, 00000000, ad00ad00> (00000000)  # GU of eth-s3p1c0 (MH)
<000080fe, 00000000, 57cfa8c0, 0a6fcf0a; 7fffffff>                 # LL of tun0c0 (MH)
<1c5983f7> -> <000080fe, 00000000, ff8ea002, 5cd608fe> (00000000)  # LL of eth-s4p1c0 (MH)
<000080fe, 00000000, ff8ea002, 692120fe; 00000001; 3136/3600>      # LL of eth-s3p1c0 (RQ)
<000080fe, 00000000, 0a6fcf0a, 57cfa8c0; 00000001; 3244/3600>      # LL of tun0c0 (TA)
<000080fe, 00000000, ff8ea002, 58d608fe; 7fffffff>                 # LL of eth-s3p1c0 (MH)
<000002ff, 00000000, 00000000, 09000000; 00000008; 1996/3600>      # RIPng (ff02::9)
<c0050120, 04005284, 00000000, ad00ad00; 7fffffff>                 # GU of eth-s4p1c0 (MH)
<135937c5> -> <c0050120, 04005284, 00000000, ad00ad00> (00000000)  # GU of eth-s4p1c0 (MH)
<16055df1> -> <c0050120, 04005284, 00000000, 21433412> (00000000)  # GU of eth0 (WR) //SRC IP
<1ab86027> -> <000080fe, 00000000, ff8ea002, 672120fe> (00000000)  # LL of eth-s1p1c0 (RQ)
```

Table 6: Check Point state table information

NOKIA
CONNECTING PEOPLE

As you can see, there are still some pieces of information that I have not solved yet. :) For instance, I cannot find out where the Source and Destination Port information is contained. (I used SPort 4096(dec)/1000(hex) and DPort 8443(dec)/20fb(hex) for this test).

Notice how Check Point mangles the IP addresses and moves pieces around? I haven't been able to figure that out or get a definitive answer about it. Oh well...

Before moving on, let's look at the service that I've created in my rulebase called "IPv6-over-IPv4" which, is just like it's namesake... Sending IPv6 traffic across IPv4 networks. For this, we use IP/41 as our service which we need to create manually.

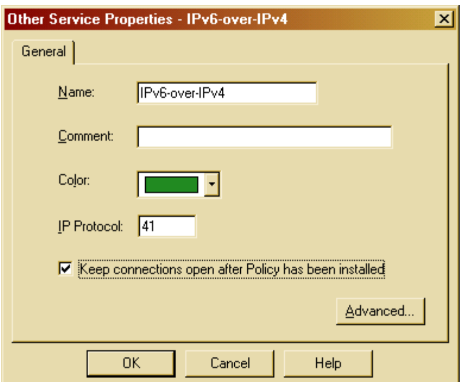


Figure 13: New IP Service – IP/41

You may be asking yourself when and why you would need to use this. Well, if you're using a tunnel broker on your residential connection (this will be covered later on) or if you want to create point-to-point tunnels with IPv6 over IPv4. I think it's time for a segue...

Point-To-Point Tunnels

So, you've got IPv6 going back and forth from workstation to workstation through your internal network. Good job. Now, how about something a little trickier.

Do you remember at the start of this paper I said I had an IP440 as well? Guess what it's purpose is? Point-to-Point Tunnel. (Not too hard to guess as this is the title of the section, eh?) However, to get to the Nokia IP440, I have to cross two IPv4-only subnets which may seem like quite the daunting task.

For this, I am using Madhatter to establish the tunnel to the IP440 machine. The IP440 has a total of four interfaces listed below:

- eth-s1p1c0 – 192.168.207.87
- eth-s1p2c0 – Live IP address (Not listed for security reasons)
- eth-s1p3c0 – 2001:5c0:8452:2::440:440
- eth-s1p4c0 – 172.16.16.50

To get to the IP440 from my subnet, I have to go from 10.207.111.0/24 to 192.168.207.0/24 and none of the intermediate devices support IPv6. So, as mentioned, we are going to setup a direct IPv6 PtP tunnel to get this to work.

IPv6 supports the transfer of packets across IPv4 in tunnels and clouds; The former is what we will be examining here. When IPv6 is encapsulated within IPv4, it uses IP/41 to accomplish this. A (very) basic figure of encapsulation is here:

Packet to then tunnel entry point

[IPv6 Header | Payload....]

Packet leaving the tunnel

[IPv4 Header | IPv6 Header | Payload....]

So, let's start getting our IP440 ready for the tunnel. If you haven't already set up the IP440 for it's designated purpose, you should do that now. Also, add the IPv4 interfaces to it and ensure that routing works okay. Once all of this has been verified, pick an interface to host the IPv6-only network behind it (I used eth-s1p3c0), configure the interface and the subsequent network behind it. Ensure routing works here as well (Basically, just go through all the steps we went through during the earlier parts of this document).

Now, log into Voyager on the IP440, go into the IPv6 Configuration section and select *IPv6 in IPv4 Tunnels*.

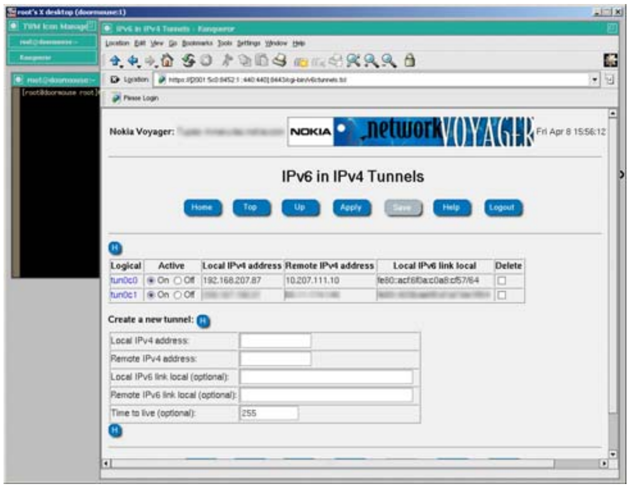


Figure 14: IPv6in IPv4 Tunnels – Nokia IPSO Configuration

Interesting Side Note: This is a screenshot of Doormouse's X Server through VNC over IPv6 only! :)

You can see here that most of the information you will be required to enter is straightforward: Enter the Local IPv4 address and the Remote IPv4 address. Done. The next step was to go to Madhatter and configure the same thing but with the IP addresses reversed (obviously). Last, but certainly not least, is routing. I enabled RIPng on the tun0c0 interfaces of each Nokia appliance and all the routing was propagated through the tunnel.

If you are going to configure a tunnel with a Linux machine, observe the following considerations...

1. The command "ip tunnel add tun440 mode sit remote 192.168.207.87 local 192.168.0.4 ttl 255" will create a Simple Interface Transition device with the two IPv4 addresses listed.
2. Next, "ip link set tun440 up" will turn our device on.
3. Assign a special Link-Local address to the device. Here's the breakdown of it.

   ip addr add fe80::<hex_of>:<remote_IP>:<hex_of>:<local_IP>/64 dev <device>

   which, for this example, would be seen as

   ip addr add fe80::c0a8:cf57:c0a8:cf57:4/64 dev tun440

VRRPv3 for IPv6

Virtual Router Redundancy Protocol (VRRP) is a protocol that is used for high availability on networks. For instance, if router 'A' goes down, router 'B' (who is standing by) will take over the job. They use Virtual IP addresses (VIP) for clients to use and use Priorities for failing over. Here's a quick run-down if you've never used VRRP before...

- Router 'A' has a Priority of 100 with a Delta of 10.
- Router 'B' has a Priority of 95 and a Delta of 10.
- Router 'A' continually sends it's Priority level to a multicast address.
- If an interface goes down, the Delta is subtracted from the Priority for a new Priority (100 – 10 = 90).
- Router 'B' notices that it has the higher priority now and takes over (95 > 90).

It's really quite simple when everything works properly which is why I'm writing this section now.

VRRPv3 is used for IPv6 networks and functions in the same way as it's IPv4 counterpart with a few extra features which we'll get to soon.

First, reference my network diagram and take note of the two routers called "Redqueen" and "old110". Notice how they each have their own IP addresses but converge to a single address on each side. Our clients will use this as their default gateway instead of the physical address. Let's start setting it up, shall we?

For this to work, you will have to ensure you are using (at least) Nokia IPSO 3.8.1 or higher. Log into Redqueen and go to the IPv6 configuration area. We will need to create a Virtual Router ID (VRID) for each interface. On eth-s1p1c0, let's create VRID "1" and on eth-s3p1c0, let's call it VRID "2". Nice and simple. Apply your changes and take a look at the following screenshot:

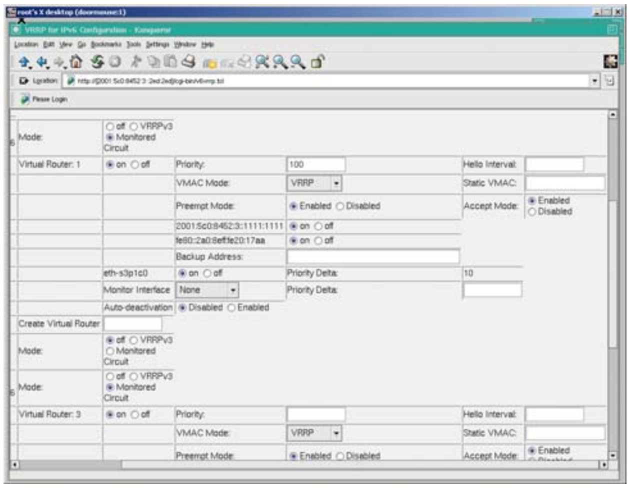


Figure 15: VRRPv3 – Nokia IPSO Configuration

FYI: The only interface you can (really) see is eth-s1p1c0 and only the name is cut off from the screenshot. Everything else of importance is still viewable.

Wow! There's a whole lot of stuff in there, eh? If you've used VRRP before, most of it should look familiar however there are two new options from IPv4: **Preempt Mode** and **Accept Mode**, both of which we'll cover. In the mean time, let's get VRRP up and running.

In the Priority area, we are going to list the default priority for the interface. If you only have two routers in the VRRP cluster, setting the Primary to 100 and the Backup to 95 is usually a safe bet.

The Hello Interval is how often VRRP packets are sent out to the multicast address. The time used to be in seconds but is now listed in **centiseconds** so 100 equals 1 full second.

The VMAC mode option is how you would like the Virtual MAC address handled by the VRRP cluster. The main reason people change this from VRRP to Static is for compatibility with some switches and routers. If you don't have any issues, you should leave it at VRRP.

Preempt Mode and Accept Mode will be discussed in detail later on in this section. For now, set them both to “Enabled”.

Now you should be prompted for a “Backup Address”. (*I don't know why they just don't call it the “VIP Address” but that's just semantics. -e.d.*) This address is going to be your VIP for the cluster. First, you **must** backup your Link-Local addresses. Enter in a Virtual IP address for you Link-Local addresses now. We'll get to Site-Local (or Global Unicast if you're live) in a moment.

Next, you will be prompted to monitor an interface. This is what makes VRRP work like a charm. This interface (that we're working on) will monitor any other interfaces you specify and, if they go down, a full fail-over occurs. Otherwise, just one interface will fail-over and you may end up with asymmetric routing. Select the drop-down menu and select eth-s3p1c0 to monitor.

Press Apply and you will notice that a Priority Delta box has appeared next to the Monitor Interface selection. In here put in the number 10. This is the number that will be subtracted from the default Priority for the new, fail-over Priority.

Also, we can add another “Backup Address” on this interface. Since we already have our Link-Local addresses in there, let's add our Site-Local (or Global Unicast if you're live) into the box. Apply and Save your changes.

If all has gone well, you should now have VRRPv3 setup on your routers and, in the event one goes down, the other will take over. Tell your clients to use the Virtual IP address (Backup Addresses) for their default gateways.

Well, I did promise to talk about those two new features: Preempt Mode and Accept Mode, so here we go...

Accept Mode

Accept Mode, which is disabled by default, determines whether or not the cluster will allow direct connections to the VIP address. For most people, having this set to “Enabled” is going to be their best option as they will most likely be using this as a gateway router for clients to connect to another network. This is like the IPv4 checkbox in Nokia IPSO VRRP that says “Accept connections to the Virtual IP address”.

Preempt Mode

Preempt Mode, however, is completely new to the protocol. Preempt Mode is Enabled by default which, again, is a good thing for most people. Let's say that Router 'B' is in Master State with a Priority of 95 and Router 'A' comes online with a Priority of 100; With Preempt Mode Enabled, Router 'A' will take over as Master status and Router 'B' will demote itself to Backup State. Why would you **not** want this? Let's say you need to do some work on Router 'A' (upgrading/configuring/whatevering) but don't want to take it out of production in the event that Router 'B' fails. You lower the Priority on Router 'A' so now it's in Backup State; Then, you turn Preempt Mode off on both of them and, lastly, you re-prioritize Router 'A' back to 100 and do your work. Now, Router 'B' (with a lower Priority) is routing all your traffic and Router 'A' (with a higher Priority) will take over **only** if Router 'B' fails.

Another good reason to disable Preempt Mode is if your VRRP kicks in **before** your firewall software. Router 'A' goes down (for whatever reason) and Router 'B' takes over as it should. When Router 'A' boots back up, VRRP fires up during the OS loading stage which will take over as Master right away however your **firewall** software hasn't loaded yet leaving your network **unprotected** and probably **unroutable** for a period of time. Once this happens, your phone starts ringing off the hook with users not being able to go anywhere and that's never fun.

The multicast address for VRRP (IPv4) is 224.0.0.18 and for IPv6 it's ff02::12 so make sure your security rules allow for communication to and from these hosts or else you'll end up with two routers in Master state.

Real-Life Example

By now, I'm sure you're thinking to yourself: This is great but how do I incorporate this in a production environment? Let's get to it and we'll have you IPv6'ing live on the 'Net in no time flat.

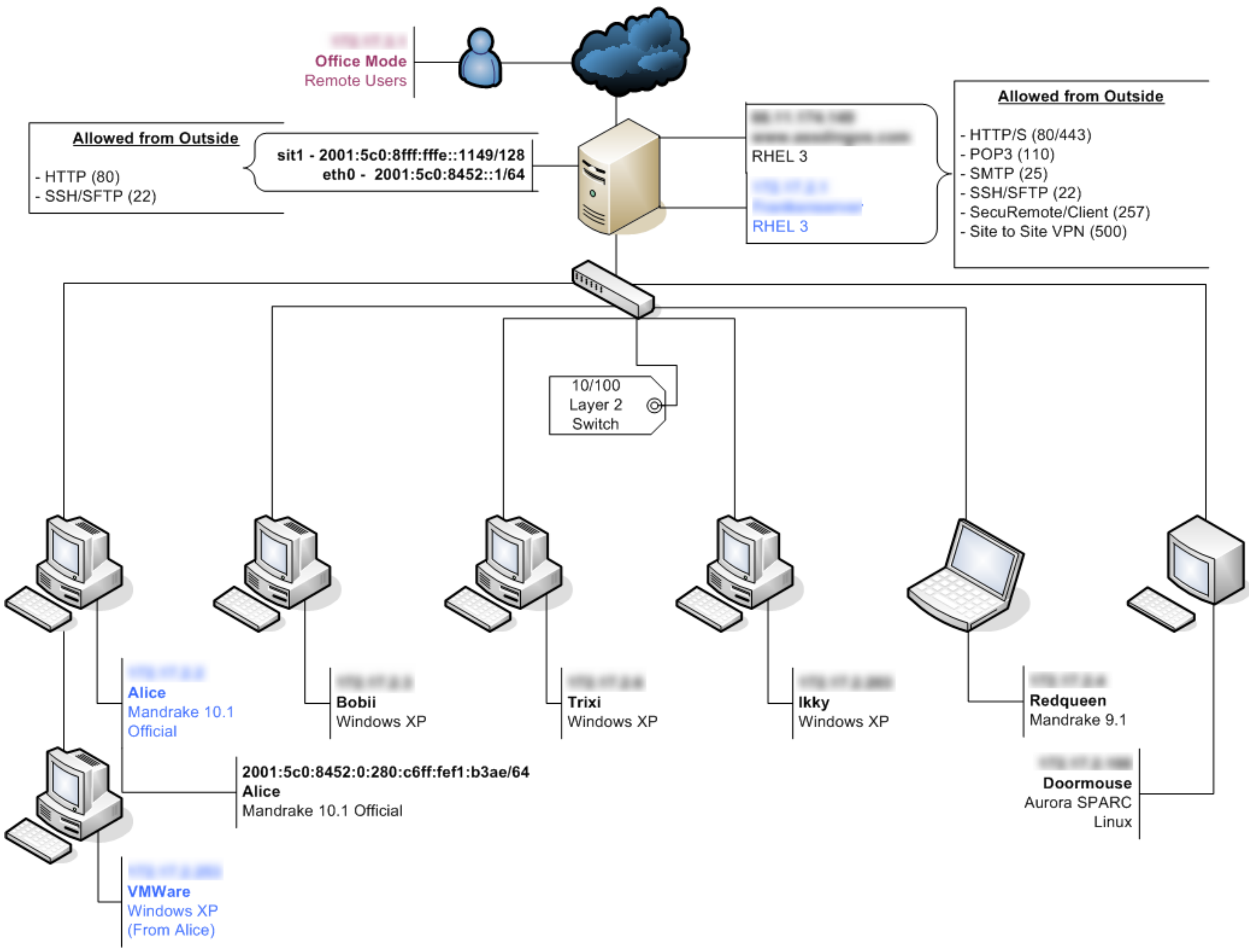


Figure 16: Example of my Home Network

I hope you understand why I've blurred out the IPv4 addresses on the inside and outside. Sure, it won't take much to figure out what they are, but I like the sense of security with it. :)

As you can see, my main desktop PC (Alice) is running Mandrake Linux 10.1 Official (2.6.8-1-custom). My server (who we'll call Frank) is running Red Hat Enterprise Linux (2.4.21-4.EL) and Check Point NGFP4 R55.

When setting up your IPv6 at home, you will need to find out some information from your Internet Service Provider (ISP). Most of them do not offer native IPv6 support so we're going to have to use a Tunnel Broker. A Tunnel Broker is a site who will offer

an IPv6-in-IPv4 Point-to-Point tunnel with you. There are quite a few to choose from but I use Hexago (www.hexago.com) for a few reasons: They're stable, they have a nice client to use on Linux and other operating systems and, finally, they're Canadian. Wooo! No matter who you choose to tunnel with, you will probably end up getting a /64 subnet assigned to you as well as a /127 Point-to-Point tunnel address. If you go with Hexago, sign-up for a free account to get a network assigned to you; If you login anonymously, you will only get a host IP address.

Get it all setup and install your client (if your broker provided you one) and finally run the client. You should now have an sit0 interface on your server/router. This, if you remember from above, is what we used on Whiterabbit when we setup our tunnel to the IP440. This is the same idea but with Global-Unicast addresses. First, try and ping6 the other end of your tunnel. If you can do that, then try to ping6 an actual IPv6 website. You may have to use the -I flag to specify an interface if your routing hasn't been applied yet.

If you've used the Hexago client, you will have had to specify an interface for which your /64 network will be based off. I chose eth0 which is my internal interface so that I could have my client computers obtain an IPv6 address from Frank. This is done with radv(d) on Linux – When your IPv6 network module loads, it will query for any RA servers to offer an IP address and, since Frank has an entire /64 network, he gets the UID from the card and generates Alice an IP address. Cool, huh?

Once you have this all setup on your client PC, try and ping6 the remote tunnel end to see if it's all working. Check your routing, security rules and tunnel broker client configuration if anything isn't working. If it is, head over to http://www.ipv6.bieringer.de and you should see a dancing penguin (Yes, a dancing penguin). Can you see it? If so, you've successfully accessed an IPv6-**only** site! That's right, this page cannot be accessed over IPv4. Wanna' see other animals dance? Head over to

http://[2001:200:0:8002:203:47ff:fea5:3085]/ and look at the dancing turtle. Look at it go... Wheee!

Basic Troubleshooting

So, something's gone wrong or maybe not even working in the first place. Where to start? First and foremost, *tcpdump* is your best friend. *tcpdump* has a sister named *Ethereal* who is, actually, a bit prettier than her brother but they both get the job done.

If you're using *tcpdump*, one of the little tricks you pick up is using *grep* in conjunction with it. For instance, if you just run *tcpdump -i eth0* and hit [ENTER], you'll probably end-up with a tonne of traffic that you don't care about at the moment. Let's say you're seeing a lot of VRRP advertisements and some v4 arp who-has that are just filling the screen up. Try: *tcpdump -i eth0* | *grep -v VRRP* | *grep -v arp* which will exclude those regex (REgular EXpressions) after -v.

Check for any firewall rules (or, if you didn't listen to me and are using a router, any ACL's) which may be restricting access to the packets you are trying to send back and forth.

Cables – Do you know how much troubleshooting can be solved by ***accurately*** checking the cables? Check the cables. Make a loopback connecter as well to test ports. They're easy as pie to build (although I can't cook to save my life) and they will tell you when a port has gone bad – (lo + RJ-45 Port) – LED = Dead Port.

TCP stacks – Can you ping6 ::1 at all? If you get an error about "Cannot Assign Requested Address" then you need to insmod ipv6.

Take a break. Decompress and play some games for a bit. Unwind and go for a walk. Do whatever it is that clears your mind.

Last but not least, K.I.S.S. - Keep It Simple, Stupid. Although I'm not trying to call anyone stupid, keeping it simple will save you from many headaches later on especially with the Pointy-Haired Boss (*Thanks Dilbert – e.d.*).

About the Author

My handle is Gr@ve_Rose, not the most *31337* handle to have, but I like it. I've been using computers since I was about six years old on my Dad's Commodore 64 where I started hacking. I used to play games and change the BASIC code to go to the last level after the opening scene. ^_^ After moving to Ottawa, we got our first x86 computer; It was a 386/33 with 8 whole megs of RAM and a thirty meg hard disk. It cost us roughly two thousand dollars. We eventually started getting new computers and I kept playing with them and learning more about them. After high school, I started working with Digital Equipment Company (DEC) who are infamous with their Alpha chips. I worked in the MIS department and that's where my love of Unix started. I quickly ran out and bought a copy of Red Hat 4.2 from EB and installed it on an old 486. Thus, the journey started...

I currently work with Nokia security appliances and Check Point FireWall-1 which, if you ask me, is a lot of fun. It helps keep me on my toes, always learning more and gives me legitimate reasons to try and crack networks (Only my own and others with permission). I've been published thrice in *2600 – The Hacker Quarterly* and am continuing to submit articles to help benefit the hacker community. Please see http://www.catb.org/~esr/jargon/html/H/hacker.html and http://www.catb.org/~esr/jargon/html/H/hacker-ethic.html for what I mean.

I hope that this document has helped start you on your path of IPv6. Learn what you can, share it with others and continue to learn; The process never stops.

About Nokia
Nokia is the world leader in mobile communications, driving the growth and sustainability of the broader mobility industry. Nokia is dedicated to enhancing people's lives and productivity by providing easy-to-use and secure products like mobile phones, and solutions for imaging, games, media, mobile network operators and businesses. Nokia is a broadly held company with listings on five major exchanges.

For more information, please visit *http://www.nokia.com/forbusiness*

Americas
Nokia
313 Fairchild Drive, Mountain View, CA 94043
Tel: 1 877 997 9199
Email: mobile.business.americas@nokia.com

Europe, Middle East and Africa
Nokia
Nokia House, Summit Avenue
Southwood, Hampshire, GU14 0NG, UK
Tel UK: +44 161 601 8908
Tel France: +33 170 708 166
Email: mobile.business.emea@nokia.com

Asia Pacific
Nokia
438B Alexandra Road
#07-00 Alexandra Technopark, Singapore 119968
Tel: +65 6588 3364
Email: mobile.business.apac@nokia.com

www.nokia.com

NOKIA
CONNECTING PEOPLE