IPv6 Hardening Guide for Windows Servers

How to Securely Configure Windows Servers to Prevent IPv6-related Attacks

Version:	1.0
Date:	22/12/2014
Classification:	Public
Author(s):	Antonios Atlasis

ERNW
providing security.

Table of Content

ERNW
providing security.

List of Figures

1 HANDLING

The present document is classified as PUBLIC. Any distribution or disclosure of this document SHOULD REQUIRE the permission of the document owner as referred in section "Document Status and Owner".

1.1 Document Status and Owner

Title:	IPv6 – How to Securely Configure Windows Servers to Prevent IPv6-related Attacks
Document Owner:	ERNW GmbH
Version:	1.0
Status:	Effective
Classification:	Public
Author(s):	Antonios Atlasis
Quality Assurance:	Enno Rey

2 INTRODUCTION

2.1 Goal, Scope and Assumptions of this Study

The goal of this study is to propose proactive configuration measures so as to prevent most of the known IPv6-related attacks, while, on the other hand, keeping the configuration "manageable" to the best possible extent.

This study is about IPv6-capable Windows servers with high requirements regarding security. The assumptions used for study purposes, are the following:

- The organization has enough resources to undergo any type of (manual) configuration that may be required.
- It is important to fully protect the servers even from their local link environment. However, the scope of this study is IPv6-only hardening. Any other type of hardening (e.g. DC hardening, web server hardening, database hardening, etc.) are beyond the scope of this study.
- The services provided by the IPv6-capable servers do not rely on any IPv6 Extension header, or on any multicast traffic.

It should be noted that all the accompanying notes[1] we gave when publishing the similar document for Linux apply even more for Windows and that every step (in particular stuff like disabling MLD) must be carefully tested in your environment. The present paper is intended mostly to serve as a source of inspiration ("what could be done") and for documentation purposes ("how to do it").

2.2 IPv6-related Attacks to Mitigate

The hardening guidelines provided in this study aim at mitigating the following IPv6-related attacks:

- Router Advertisement related attacks (MiTM, router redirection, DoS, etc).
- MiTM / DoS attacks during the Neighbor Discovery process.
- DoS during the DAD process.
- IPv6 Extension Headers related attacks.
- Smurf-like attacks at the local link.
- Packet Too Big Attacks
- Reconnaissance by exploiting various ICMPv6 messages.

2.3 Methodology

Based on the attacks described in subsection 2.2, in order to prevent them in a nutshell the following configurations are suggested:

- Configure manually:
  - IPv6 host address
  - IPv6 gateway
  - IPv6 DNS server
  - MTU (≥1280 bytes)
  - Neighbor Cache

[1] *See http://www.insinuator.net/2014/12/ipv6-hardening-guide-for-linux-servers/.*

  - Current Hop Limit
- Disable:
  - Acceptance/Processing of Router Advertisements
  - DAD process
  - MLD process. etc
- Configure the local host firewall to block:
  - IPv6 Extension Headers
  - Unwanted ICMPv6 messages.

2.4 Tested Platforms

As testing platform, Windows 2012 R2 was used. The reason for choosing this is that since this is the newest Windows enterprise OS, it offers the best possible support of IPv6 among the Windows OS family and, obviously, it is expected to be the most long-term one nowadays.

However, the same procedures / guidelines are expected to work at other latest Windows OS versions.

The example addresses used in this document is an IPv6 address scope reserved for documentation purposes, as defined by RFC3849, that is, 2001:DB8::/32.

The Network Interface that we use at our examples is the *„Ethernet 3"* one.

3 Background Information

Windows offers a plethora of configuration options using the *netsh* command and specifically, the *netsh interface ipv6* option. This command, in a nutshell provides the following options (only the most related ones are included, while in some of them the available suboptions are not provided for brevity reasons):

```
netsh interface ipv6 set/show
```

- *address* (sets the IP address or default gateway to an interface)
- *dnsservers* (sets DNS server mode and addresses)
- *global*
  - *defaultcurhoplimit* (default hop limit of packets sent)
  - *neighborcachelimit* (maximum number of neighbor cache entries)
  - *routecachelimit* (maximum number of route cache entries)
  - *reassemblylimit* (maximum size of reassembly buffer)
  - *icmpredirects* (whether the path cache is updated in response to ICMP redirecets)
  - *sourceroutingbehavior* (determines the behaviour for source routed packets)
  - *mldlevel* (Level of multicast support)
  - *multicastforwarding* (whether multicast packets can be forwarded)
  - *randomizeidentifiers* (whether interface identifiers are randomised)
- *interface*
  - *forwarding* (whether packets can be forwarded)
  - *advertise* (whether Router Advertisement – RAs – are to be sent)
  - *mtu* (the MTU interface)
  - *siteprefixlength* (default length of global prefix for the site)
  - *nud* (whether neighbor unreachability detection is enabled)
  - *retransmittime* (retransmit time – in ms)
  - *dadtransmits* (numner of duplicate address detetion tramsits)
  - *routerdiscovery* (can be enabled, disabled, or controlled by DHCP)
  - *managedaddress* (whether managed address configuration is enabled)
  - *advertisedrouterlifetime* (router lifetime – in seconds)
  - *advertisedefaultroute* (whether interface will be advertised as default router)
  - *currenthoplimit* (hop limit in outbound traffic)
- *neighbors* (sets a neighbor address)
- *prefixpolicy* (modifies prefix policy information)

privacy (modifies privacy configuration parameters)

route (modifies route parameters)

interface

nexthop

siteprefixlength

metric

publish

validlifetime

preferredlifetime

teredo (sets a teredo state)

In most of the above options, two more suboptions are also provided which can be used accordingly:

store

active (set only lasts until next boot)

persistent (set in persistent)

For instance, to display the current global configuration settings, we can use the following command:

```
C:\> netsh interface ipv6 show global
```

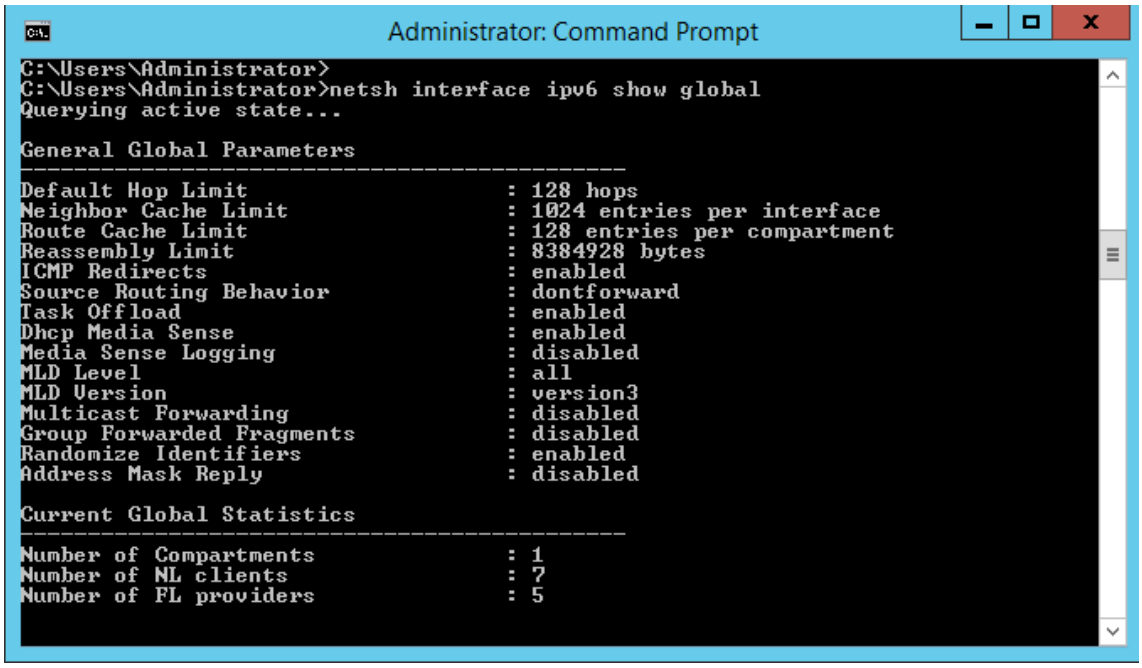

Figure 1: Default global IPv6 configuration parameters in a Windows 2012R2 server.

4 Preparation – Configure IPv6 Addresses, Default Gateway and DNS Server Statically

Windows servers, especially Domain Controllers, DNS servers, etc., should have configured their IPv6 address and gateway.

4.1 Manually Assign Global IPv6 Address, Default Gateway and Preferred DNS Server Through a GUI

To manually assign an IPv6 address, a gateway and a DNS server, you can use the usual Windows GUI. Pls note the below is just for sample purposes; in most networks the default gateway(s) and the DNS server(s) are different systems ;-)

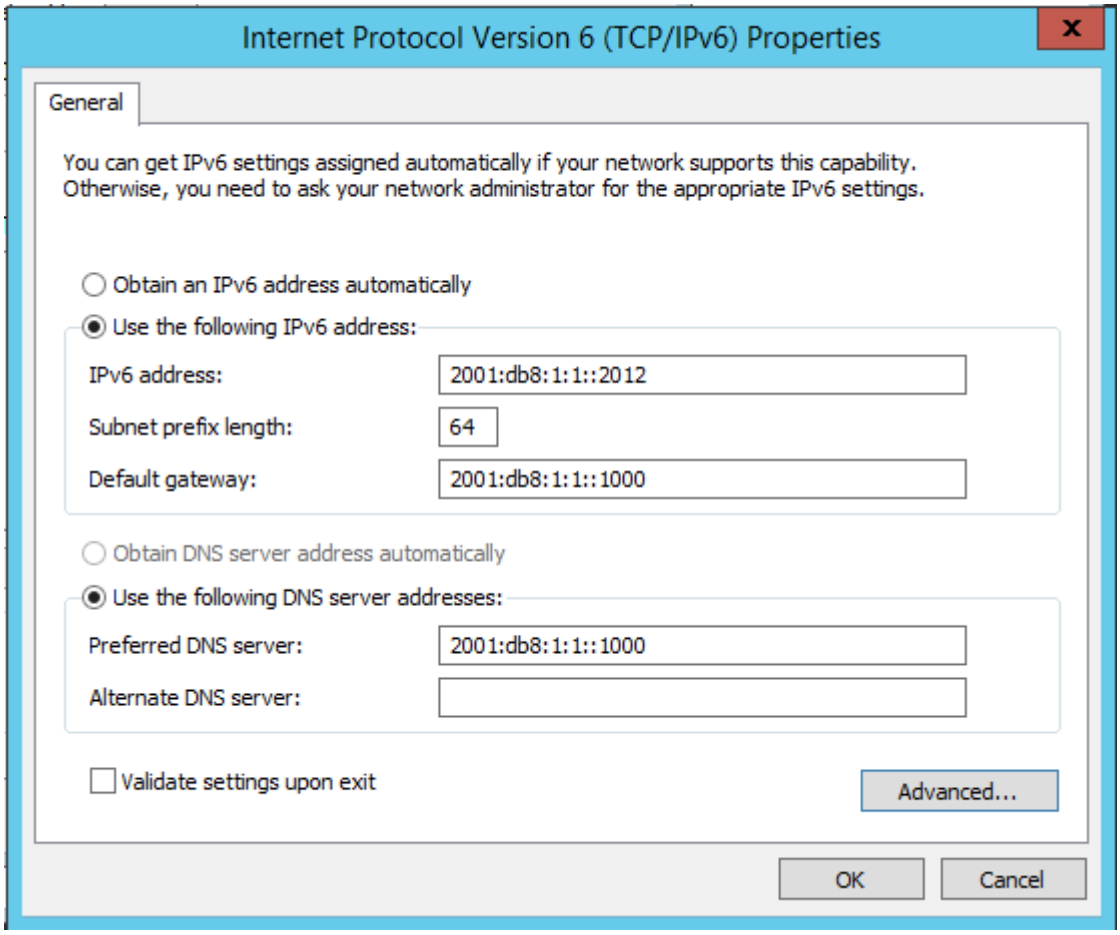

Figure 2: Configure statically the IPv6 address, the default gateway and the preferred DNS server in a Windows 2012R2 server.

4.2 Manually Assign Global IPv6 Address, Default Gateway and Preferred DNS Server Using *netsh*

Alternatively, you can use the *netsh* command line options. Example:

```
C:\> netsh interface ipv6 set address "Ethernet 3" "2001:db8:1:1::2012"
unicast infinite infinite
```

Where infinite is the preferred and the valid lifetime.

For configuring the default gateway, you can use:

```
C:\> netsh interface ipv6 set route ::/0 "Ethernet 3" 2001:db8:1:1::1000
```

And for configuring the preferred DNS server, you can use:

```
C:\> netsh interface ipv6 set dnsservers name="Ethernet 3" static
"2001:db8:1:1::1000" primary
```

4.3 Disable Automatic Configuration, Neighbor Discovery Process and MLD Operation

After assigning manually our IPv6 address and gateway, it's time to disable the system's stateless autoconfiguration after accepting RAs. Here, we shall use a kind of trick. We shall disable MLD, which will also disable the Neighbor Discovery Process, DAD, etc. To this end, we shall use the following command:

```
C:\netsh interface ipv6 set global mldlevel=none
```

It should be noted that this exact step is only to be performed in very strict environments as this somewhat breaks a lot of core IPv6 functionality[2].

4.4 Adding Static Entries Into Neighbor Cache

Since we have implicitly disabled the Neighbor Discovery process, we need to add permanent entries in the Neighbor Cache in order to make the layer-2 communication at the local link possible. To do so, we can use the following command:

```
C:\> netsh interface ipv6 set neighbors "Ethernet 3" 2001:db8:1:1::1000"
"12-34-56-78-9a-bc"
```

Where *"12-34-56-78-9a-bc"* is the MAC address of the host with the IPv6 address " 2001:db8:1:1::1000". Of course, we need to:

a. Add similar entries for all neighbours of the host
b. Repeat the whole process in EACH host of the local link.

[2] *On the relationship of MLD and Neighbor Discovery see also* http://www.insinuator.net/2014/09/mld-and-neighbor-discovery-are-they-related/.

5 Further Hardening of IPv6 Servers

5.1 Disabling ICMPv6 Redirects

To disable ICMPv6 Redirection, use the following command:

```
C:\netsh interface ipv6 set global icmpredirects=disabled
```

5.2 Configuring Manually the Default Current Hop Limit

To configure the default current hop limit for outgoing packets, use the following:

```
C:\netsh interface ipv6 set global defaultcurhoplimit=64
```

By doing so, we set the default value of the Current Hop Limit to 64, the one used by Linux systems.

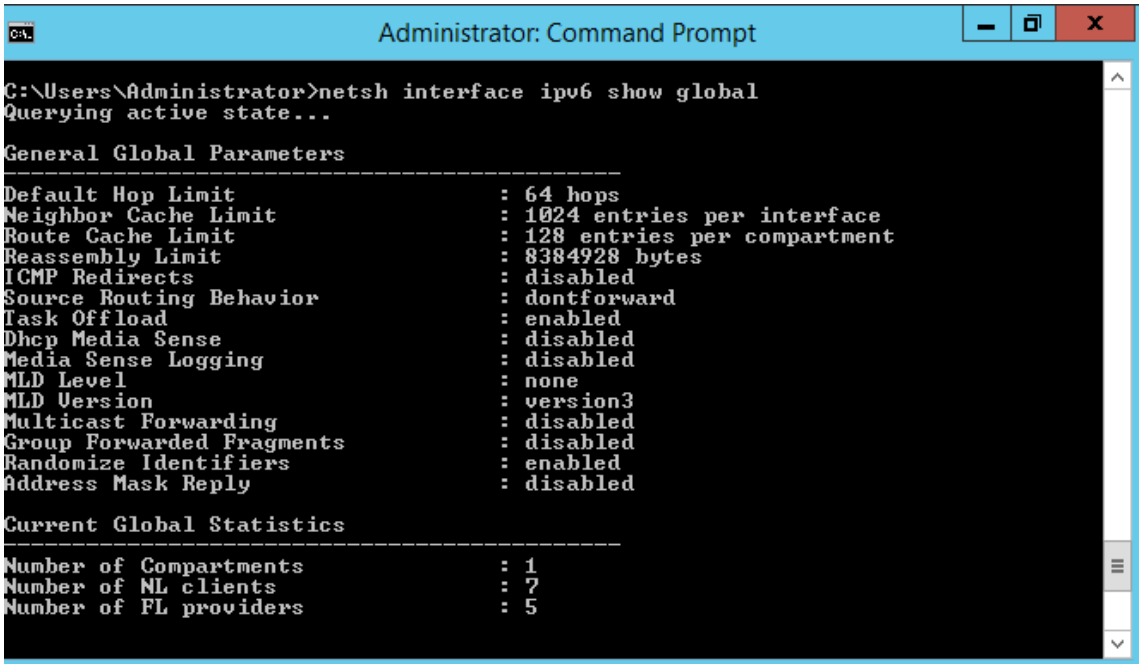

Figure 3: netsh IPv6 global configuration parameters after hardening.

5.3 Disabling ISATAP and Teredo (if enabled)

By default, in a Windows 2012 R2 host ISATAP tunnel adapters are enabled (see figure below):

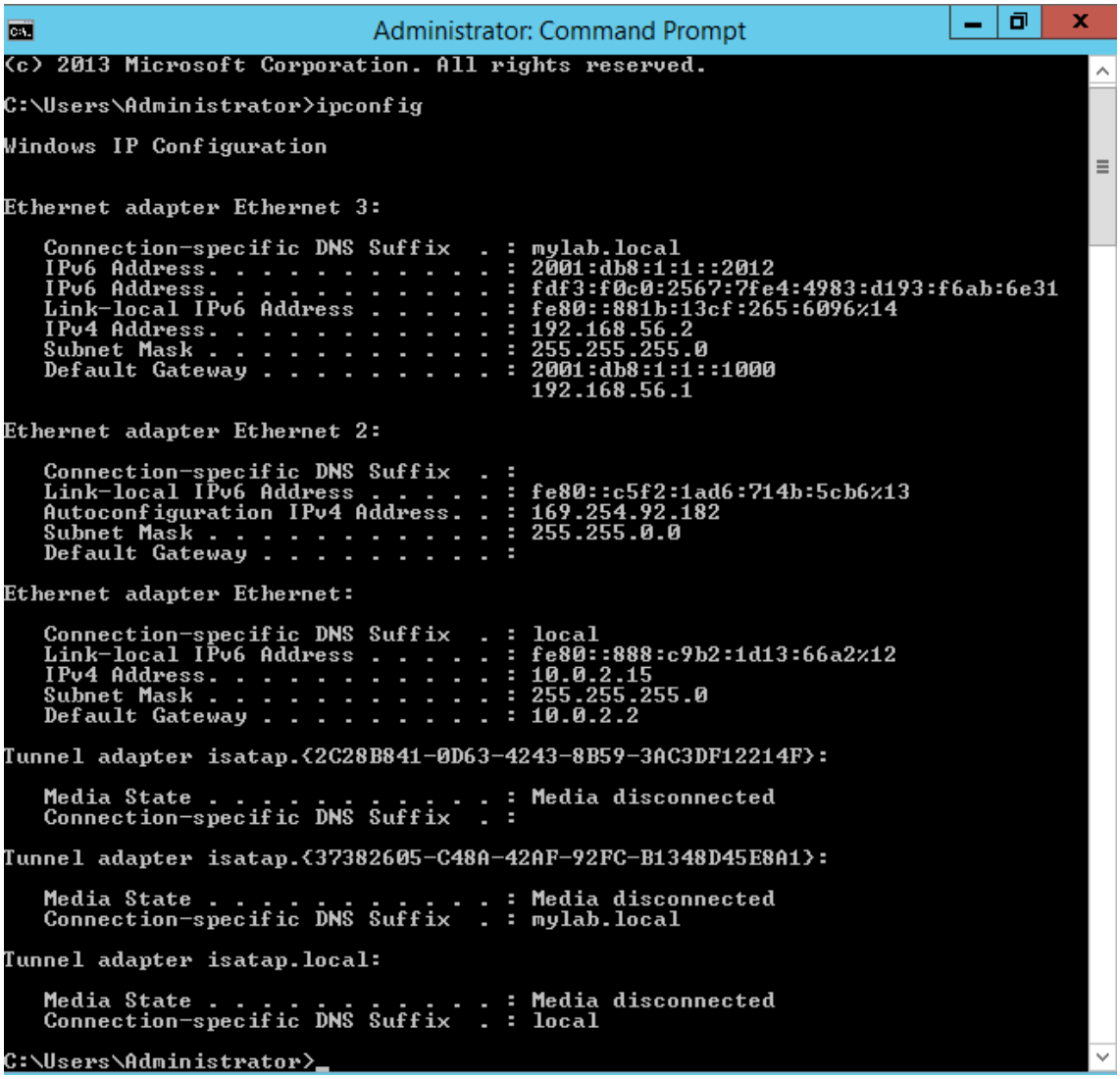

Figure 4: Tunnel adapters enabled by default in a Windows 2012 R2 host.

To disable ISATAP, run the following command:

```
C:\> netsh interface ipv6 isatap set state disabled
```

Similarly, if Teredo is enabled:

```
C:\> netsh interface ipv6 set teredo type=disabled
```

For 6to4:

```
C:\> netsh interface ipv6 6to4 set state disabled
```

Make sure that all these interfaces are or have been disabled by running an *ipconfig* command.

Teredo, 6to4 and isatap can also be disabled for a group of computers using Group Policy[3], by going to „Administrative Templates" → „Network" →"TCPIP Settings" → "Ipv6 Transition Technologies" (see figure below):

[3] *Which then modifies the well-known* DisabledComponents *registry parameter, see also KB article KB929852.*

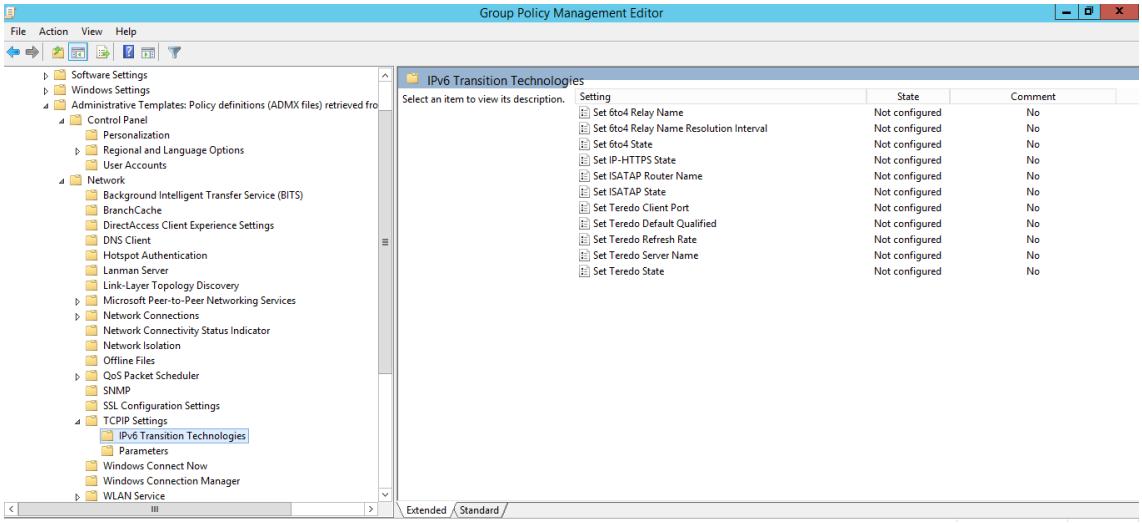

Figure 5: Configuration of Tunnel Adapters via Group Policy

5.4 Setting the MTU, Disabling Router Discovery and Minimising DAD Transmits Per Interface

The netsh command gives us a lot of options, as we can see in section 3, which allow us to configure them even per interface. For instance, if we want to set the MTU, disable the Router Discovery and minimising the DAD Transmits for interface „1", we can use the following command:

```
C:\> netsh interface ipv6 set interface "1" mtu=1280 dadtransmits=0
routerdiscovery=disabled
```

5.5 Defining Manually Static Routes

We can also add some static routes to our systems, using the set route command. Example:

```
C:\> netsh interface ipv6 set route 2001:db8:1:2::/64 „Ethernet 3"
2001:db8:1:1::1000 0 2 no 5000 6000
```

Where 2 is the metric, 5000 is the valid lifetime, 6000 is the preferred lifetime, etc.

For more information, please run:

```
C:\> netsh interface ipv6 set route
```

6 CONFIGURING THE HOST FIREWALL

6.1 ICMPv6

6.1.1 Incoming ICMPv6

Allow the following ICMPv6 incoming types of messages:

- Packet Too Big.
- Destination Unreachable
- Echo Replies
- Time Exceeded (Type 3 Code 0)
- Parameter Problem (Type 4 Codes 1 and 2)
- For network troubleshooting purposes, you can allow Echo Requests messages from very specific(s) hosts (e.g. admins' hosts).

6.1.2 Outgoing ICMPv6

Allow the following ICMPv6 outgoing messages.

- Packet Too Big
- Echo Requests
- Echo Replies

6.1.3 Default Policy

All the rest ICMPv6 traffic should be blocked

Based on the above, the incoming and outgoing ICMPv6 configurations are shown below:

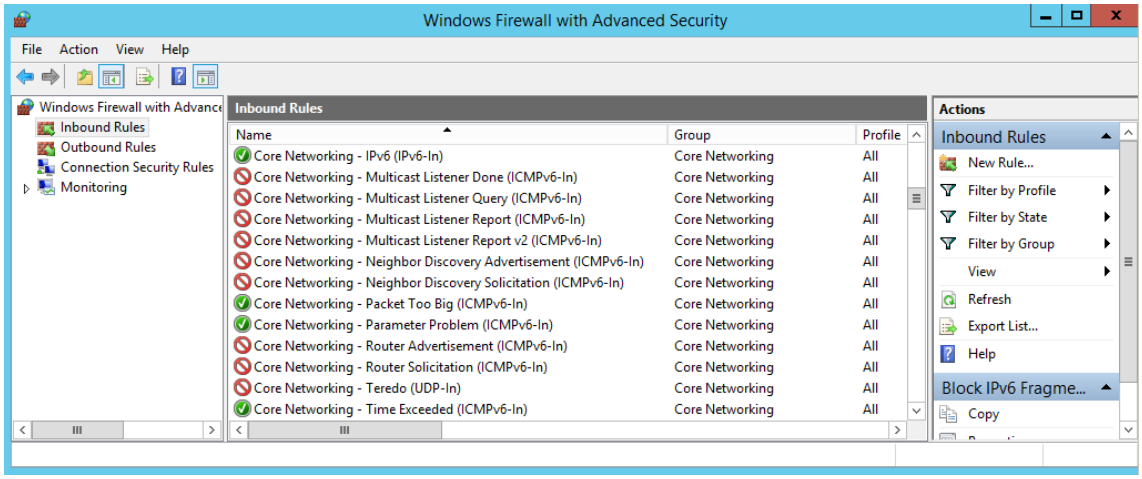

Figure 6: Inbound ICMPv6 Rules at Windows 2012 R2 server after hardening

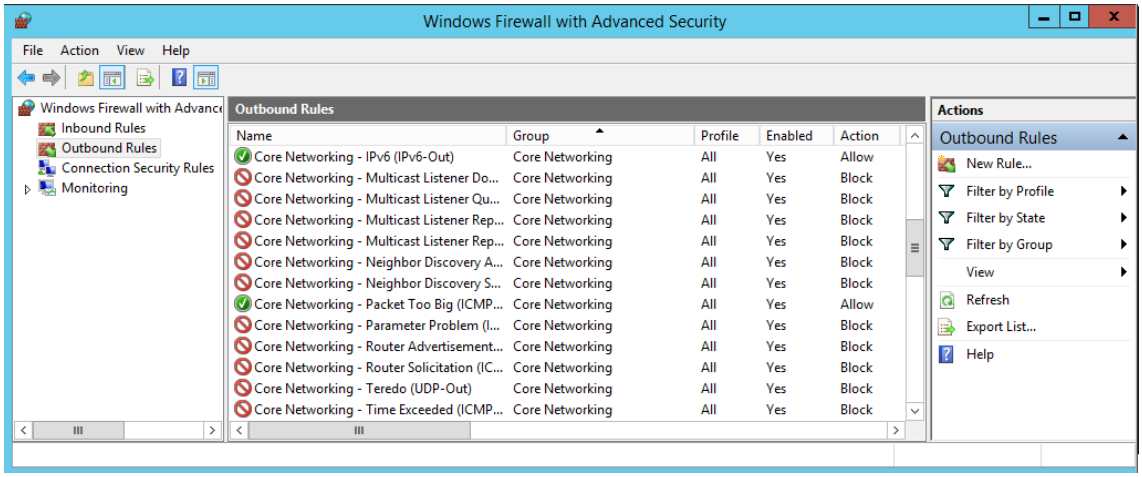

Figure 7: Outbound ICMPv6 Rules at Windows 2012 R2 server after hardening

6.2 Prevent „Smurf"-like Attacks at the Local Link

First, let's mitigate any kind of potential "smurf" attack at the local link by dropping traffic to all-nodes link-local multicast address (even for otherwise allowed messages). To do so, you need to block all IPv6 traffic destined to ff02::1 using "Windows Firewall" → "Advanced Settings"→ "Inbound Rules" → "New Rule" → "Custom" and then as protocol choose the IPv6 and finally set "Local IP Address" at "Scope" to ff02::1.

6.3 IPv6 Extension Headers

Normally, no IPv6 Extension headers should be allowed. To this end, any packets containing any IPv6 extension header should be dropped (unless, it has been shown that it is required for very specific reasons).

Windows 2012 R2 support the most well-known IPv6 Extension headers (defined in RFC 2460) and specifically, the Hop-by-Hop, the Destination Options, the Fragment Extension header, the Routing header and the No Next Header. To do so you can go to:

"Windows Firewall" → "Advanced Settings"→ "Inbound Rules" → "New Rule" → "Custom" → "All Programs" and at protocol and ports you get the following drop-down menu:

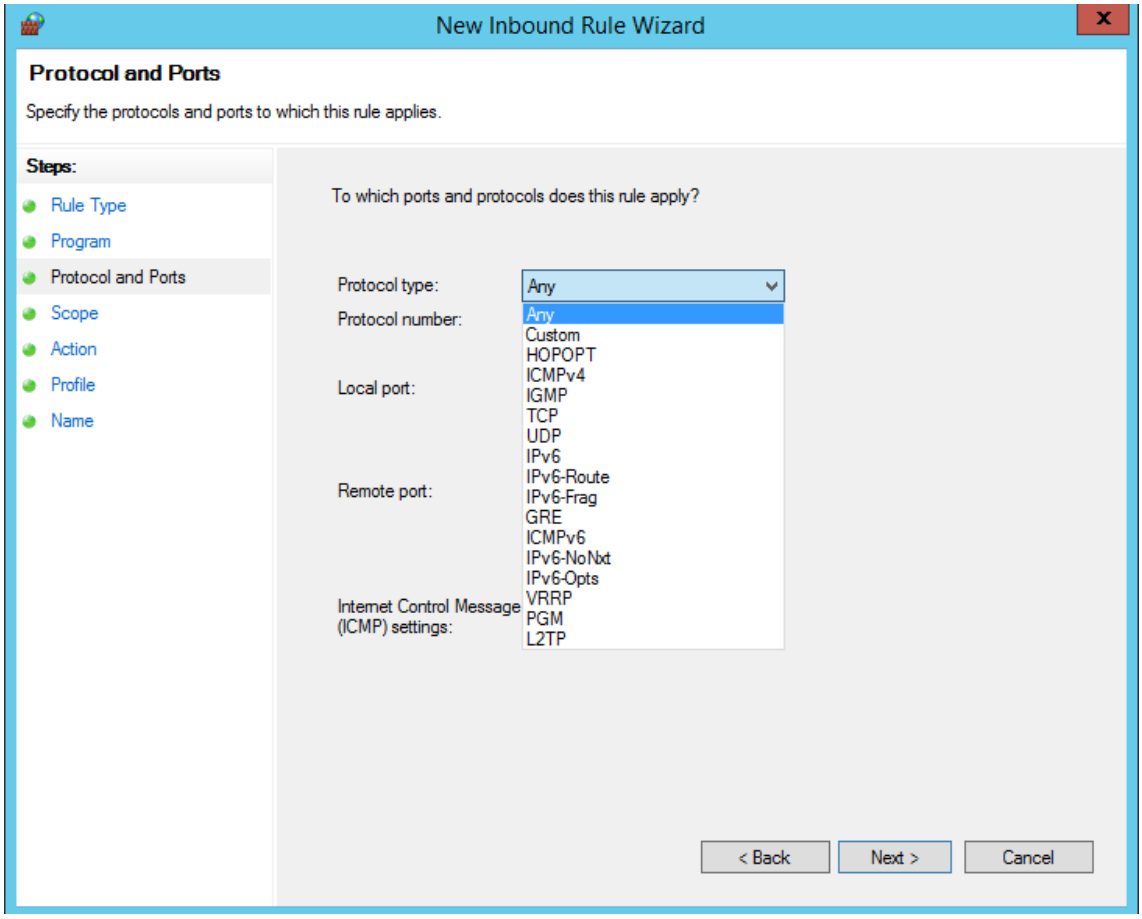

Figure 8: Supported protocols at Windows Firewall

As you can see, you can choose, among else, the HOPOPT (Hop-by-Hop), IPv6-Route, IPv6-Frag, IPv6-NoNxt (No Next Header) and IPv6-Opt (Destination Options) Extension headers. You can also choose "Custom" and then define the protocol number on your own.

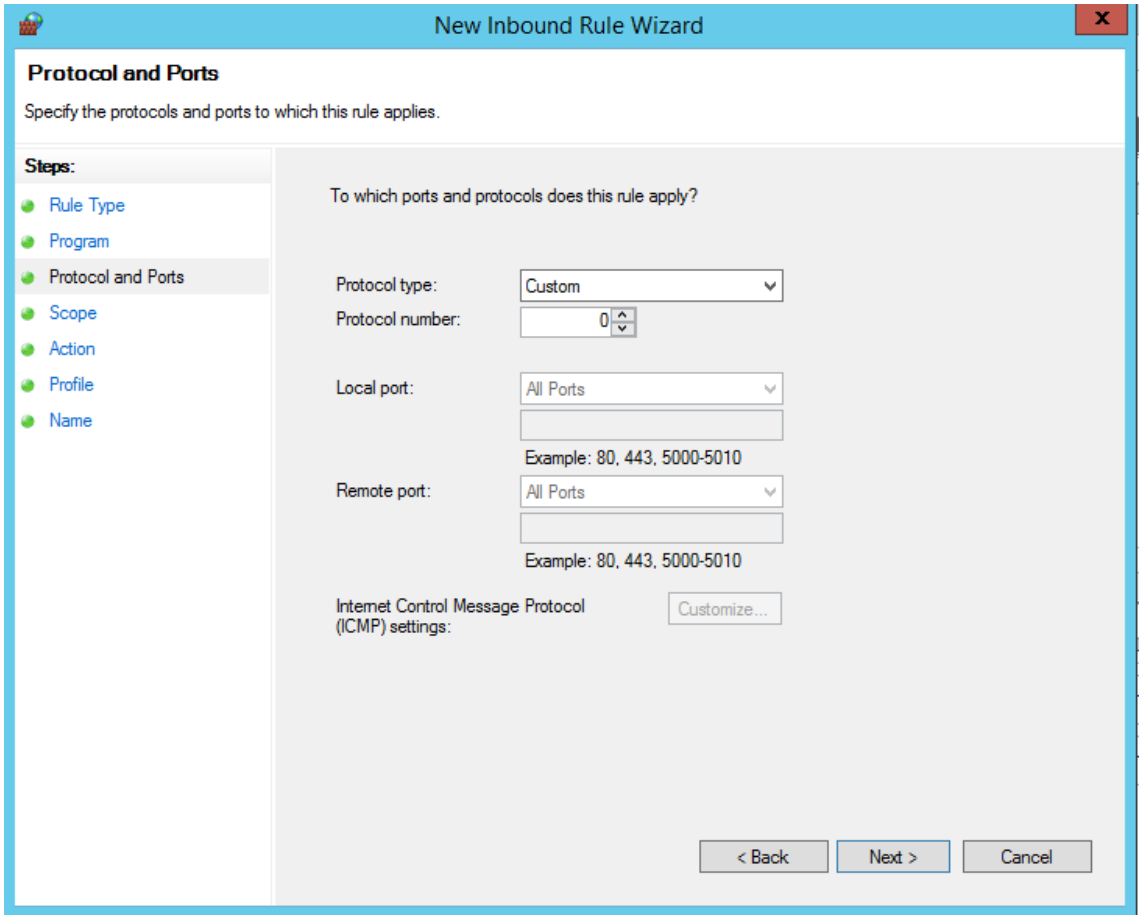

Figure 9: Defining a custom protocol at Windows Firewall

However, while using a specific Extension header (e.g. Routing header), you do not have any options to block for instance Type-0 and allow Type-2.

Note: during our tests it seemed that after creating such rules, packets that incorporate such Extension Headers are NOT blocked (e.g. TCP SYN packets to port 445 or ICMPv6 Echo Requests). We will further test this and follow up through "appropriate channels" in case here's an unexpected behaviour.

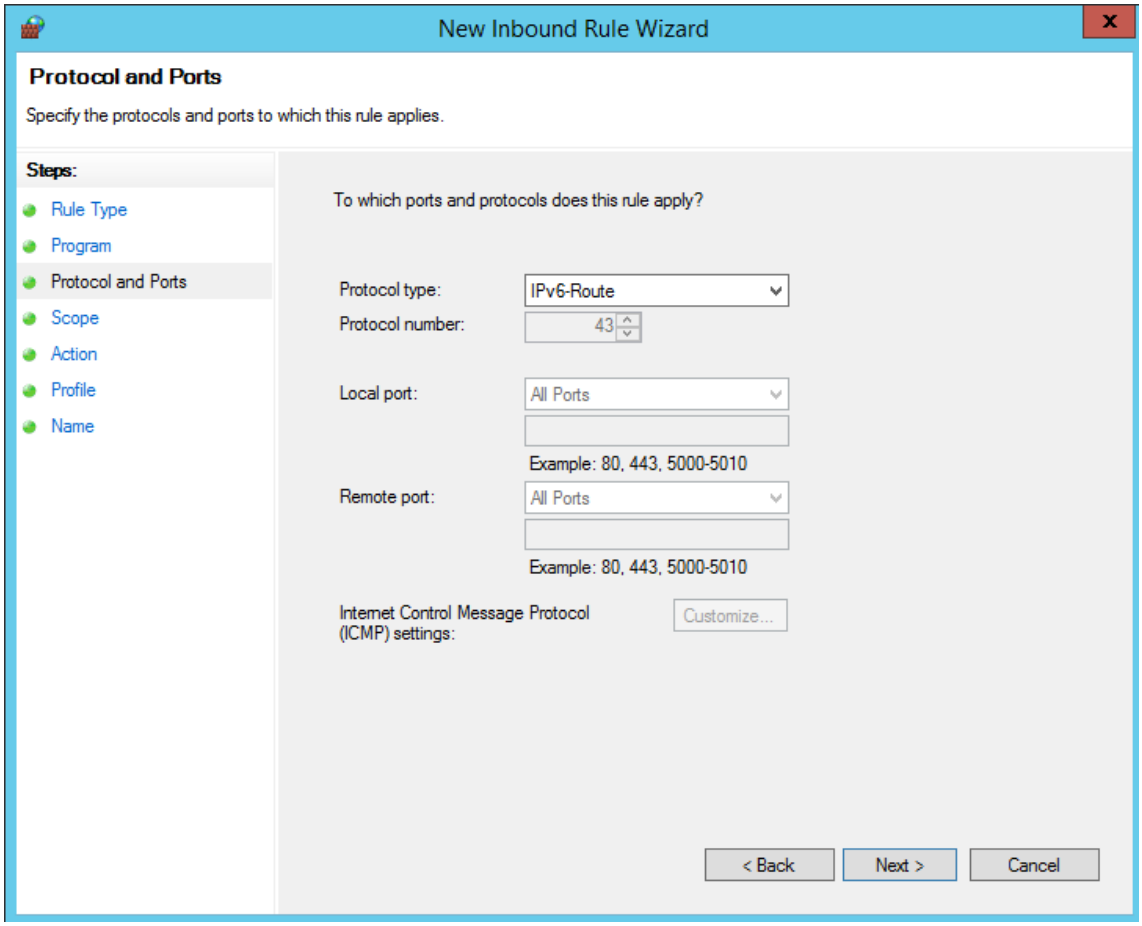

Figure 10: Lack of defining explicit options at IPv6 Extension Headers.

Based on the above discussion, the inbound Windows Firewall rules regarding IPv6 Extension Headers should look like as following:

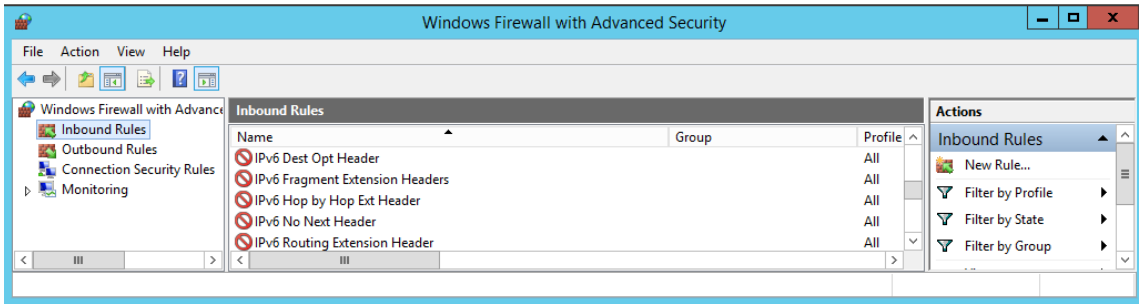

Figure 11: Blocking IPv6 Extension Headers at Windows Firewall Inbound Rules.

6.4 Using Group Policy to Deploy a Windows Firewall Policy for a Group of Computers

Thankfully we do not have to repeat the above procedure regarding firewall rules for all the hosts in our network one-by-one, but we can use Grouo Policy instead. By going to „Windows Settings“ → „Security Settings“ → „Windows Firewall with Advanced Security“ we get the following screen:

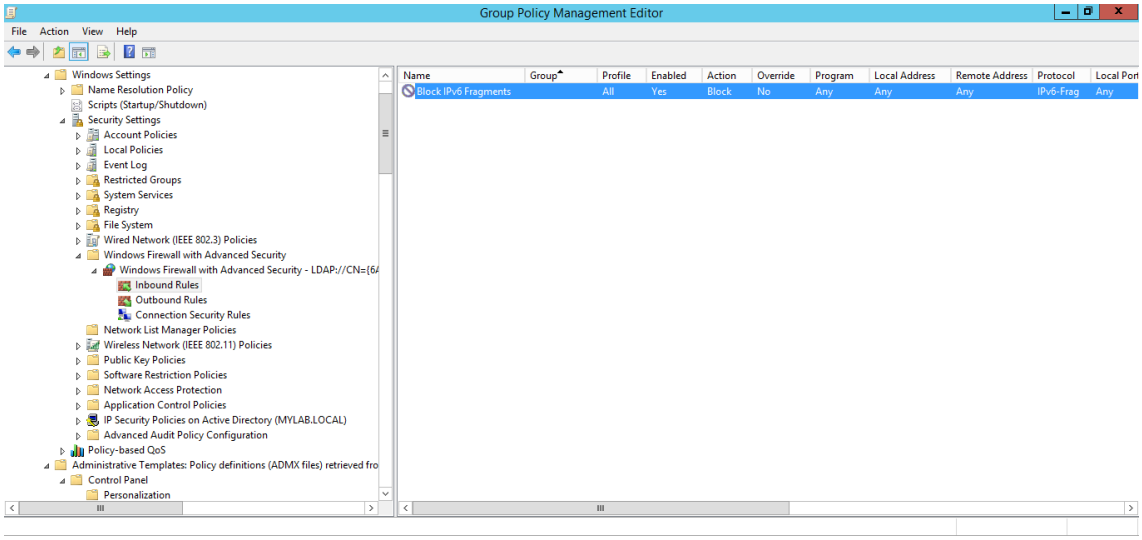

Figure 12: Adjusting Windows Firewall settings using Group Policy.

In the above environment you can use the same procedures as described above to define your rules (e.g right-click Inbound Rules, then "New Rule", etc.).

7 Applying The Configuration to a Group of Machines

The configuration described above can be divided in two separate ways of applying them: a) Using the *netsh* command and b) using Group Policy.

Defining specific policies by Group Policy and then applying them to a group of machines is probably the easiest way to configure them with the least possible effort.

On the other hand, *netsh* configurations can be achieved by making suitable scripts and applying them even remotely. Netsh provides the *–r <remote machine>* option to apply the rules remotely (for more information and options, please use *netsh /?*). So, applying global configuration policies to a group of machines like the ones described in this document using a script and *netsh* is feasible. The only challenge is probably the addition of permanent entries to the Neighbor Cache of each macine, since these will differ in each one of them. Still, by combining good scripting skills and *netsh* can be achieved.

To sum up, although it may take some additional effort, Windows server hardening against the typical IPv6 attacks is feasible and can be achieved in a Windows enterprise environment with high security demands.

8 REFERENCES

[1] RFC 2460

[2] *http://technet.microsoft.com/en-us/library/cc740203%28v=ws.10%29.aspx*

[3] *http://technet.microsoft.com/en-us/library/bb490939.aspx*