IPv6 Hardening Guide for Linux Servers

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

Version:	1.0
Date:	12/17/2014
Classification:	Public
Author(s):	Antonios Atlasis

Table of Content

List of Figures

1 HANDLING

The present document is classified as PUBLIC.

1.1 Document Status and Owner

As the owner of this report, the document owner has exclusive authority to decide on the dissemination of this document and responsibility for the distribution of the applicable version in each case to the places defined in the respective section.

The possible entries for the status of the document are "Initial Draft", "Draft", "Effective" (currently applicable) and "Obsolete".

Title:	IPv6 – How to Securely Configure Linux 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

1.2 Possible Classifications:

Public:	Everyone
Internal:	All employees and customers
Confidential:	Only employees
Secret:	Only selected employees

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 Linux enterprise servers with high requirements regarding security. The assumptions used for study purposes, are the following:

- The enterprise has enough resources to undergo any type of (manual) configurations potentially required.
- It is vital important to fully protect the servers even from their local link environment. However, the scope is of this study is IPv6-only hardening. Any other type of hardening (e.g. 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 (besides solicited-node multicast addresses).

For reasons of completeness, in many cases several ways to achieve the same goal are given.

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
- Disable:
  - Acceptance of Router Advertisements
  - DAD process
  - Hop-limit, etc.
- Configure the local host firewall to block:
  - IPv6 Extension Headers
  - Unwanted ICMPv6 messages.

2.4 Tested Platforms

As testing platforms, the following Operating Systems were used:

- Red Hat Enterprise Linux 7 x86_64
- Red Hat Enterprise Linux 6.6 x86_64
- SUSE Linux Enterprise Server 12 x86_64

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 *enp0s8* one.

3 Background Information

The sysctl command is used to modify kernel parameters at runtime. */etc/sysctl.conf* is a text file containing *sysctl* values to be read in and set by sysctl at boot time. To view current values, enter:

```
# sysctl -a
```

To display just the ipv6 configuration parameters, enter:

```
# sysctl net.ipv6
```

To display the current value of a very specific parameter:

```
# sysctl net.ipv6.conf.all.accept_ra

net.ipv6.conf.all.accept_ra = 0
```

To load current settings (e.g. after modifying */etc/sysctl.conf*) , enter:

```
# sysctl -p
```

When making changes to networking configuration file(s) – e.g. for a specific network interface – you need to restart the network service so as the changes to take effect:

```
# service network restart
```

In Suse, you can also use:

```
# /sbin/rcnetwork restart
```

4 PREPARATION – CONFIGURE IPV6 ADDRESSES AND GATEWAY STATICALLY

Manually assigning an IP address as well as a gateway is preferable to accepting one from routers or from the network otherwise.

4.1 Manually Assign Global IPv6 Address

To manually assign an IP address for an interface IFACE (for example, enp0s3, enp0s8, etc.)

1. As root, edit the file:

   ```
   /etc/sysconfig/network-scripts/ifcfg-<IFACE>
   ```

   In Suse, the corresponding file is:

   NOTE: The corresponding file ins Suse Linux is */etc/sysconfig/network/ifcfg-<IFACE>*

2. Add or correct the following line (substituting the correct IPv6 address):

   ```
   IPV6ADDR=2001:db8:1:1::abcd/64
   ```

   In Suse, this command should be inserted as following:

   ```
   IPADDR='2001:db8:1:1::1234:5679/64'
   ```

3. As root, restart network service so as changes to take effect:

   ```
   # service network restart
   ```

4.2 Manually Assign IPv6 Router Address (Gateway)

Router addresses should be manually set and not accepted via any autoconfiguration or router advertisement.

1. As root, edit the file:

   ```
   /etc/sysconfig/network-scripts/ifcfg-<IFACE>
   ```

2. Add or correct the following line (substituting your gateway IP as appropriate):

   ```
   IPV6_DEFAULTGW=2001:db8:1:1::3564
   ```

3. As root, restart network service so as changes to take effect:

   ```
   # service network restart
   ```

In Suse:

1. Edit the following file:

   ```
   /etc/sysconfig/network/ifroute-eth1
   ```

2. Add the following entry:

   ```
   default 2001:db8:1:1:800:27ff:fe00:0 - eth1
   ```

NOTE: You might avoid using easily guessable IP addresses like *2001:db8:1:1::1* for routers[1].

[1] *Still it should be noted that once an address has a DNS PTR record (which probably applies to most addresses of network devices in many environments) it will be "found" anyway, see http://7bits.nl/blog/posts/finding-v6-hosts-by-efficiently-mapping-ip6-arpa.*

4.3 Disable Automatic Configuration

After assigning manually our IPv6 address and gateway, it's time to disable the system's acceptance of Router Advertisements and ICMPv6 redirects. For affecting the default configuration:

1. As root, edit the following file:

   ```
   /etc/sysconfig/network
   ```

2. Add or correct the following line:

   ```
   IPV6_AUTOCONF=no
   ```

3. As root, restart network service so as changes to take effect:

   ```
   # service network restart
   ```

This setting results in ensuring that the following kernel (sysctl) parameters are set as following:

1. Make sure (or correct, if needed) that sysctl gives the following output:

   ```
   # sysctl net.ipv6.conf.default.accept_redirects

   net.ipv6.conf.default.accept_redirects = 0

   # sysctl net.ipv6.conf.default.accept_ra

   net.ipv6.conf.default.accept_ra = 0
   ```

If needed and you want to change the sysctl entries manually, please do the following:

1. As root, edit the following file:

   ```
   /etc/sysctl.conf
   ```

2. Add the following files:

   ```
   net.ipv6.conf.default.accept_ra=0

   net.ipv6.conf.default.accept_redirects=0

   net.ipv6.conf.all.accept_ra=0

   net.ipv6.conf.all.accept_redirects=0
   ```

NOTE: The above procedure will affect the default configuration. You may (will probably) need to do the same for the existing interfaces. Assuming that our interface is *enp0s8*:

1. Edit the following file:

   ```
   /etc/sysconfig/network-scripts/ifcfg-enp0s8
   ```

2. Add or correct the following line:

   ```
   IPV6_AUTOCONF=no
   ```

3. As root, restart network service so as changes to take effect:

   ```
   # service network restart
   ```

This setting results in ensuring that the following kernel (sysctl) parameters are set as following:

1. Make sure (or correct, if needed) that sysctl gives the following output:

   ```
   # sysctl net.ipv6.conf.enp0s8.accept_redirects

   net.ipv6.conf.enp0s8.accept_redirects = 0
   ```

```
# sysctl net.ipv6.conf.enp0s8.accept_ra
```

```
net.ipv6.conf.enp0s8.accept_ra = 0
```

If needed and you want to change the *sysctl* entries manually, please enter the following:

1. As root, edit the following file:

   ```
   /etc/sysctl.conf
   ```

2. Add the following lines:

   ```
   net.ipv6.conf.enp0s8.accept_ra=0
   net.ipv6.conf.enp0s8.accept_redirects=0
   ```

NOTE: The above procedure should be repeated for ALL system's interfaces.

4.4 Manually Configure IPv6 DNS Servers

To manually configure your (IPv6) DNS servers:

1. As root, edit the following file:

   ```
   /etc/resolv.conf
   ```

2. Add a line like the following for each DNS server:

   ```
   nameserver 2001:db8:1:1::fedc:abce
   ```

4.5 Manually Assign a Link MTU

You can optionally configure the MTU of the link. To do so:

1. As root, edit the file:

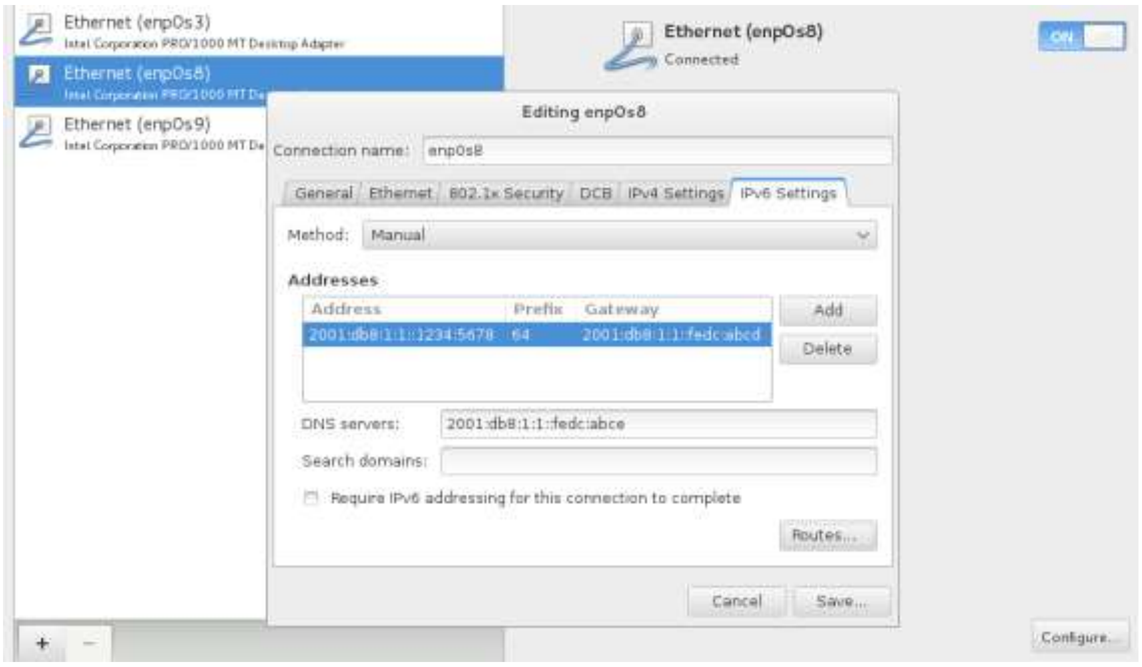
   ```
   /etc/sysconfig/network-scripts/ifcfg-<IFACE>
   ```

2. Add the following line (an IPv6 MTU should not be smaller than 1280 bytes):

   ```
   IPV6_MTU="1280"
   ```

4.6 Manually Assign Global IPv6 Address During System Set-Up

When installing a new Red Hat 7 system, you can directly configure it to use a static IPv6 global address as following (see figure below):


Ethernet (enp0s3)
Intel Corporation PRO/1000 MT Desktop Adapter
Ethernet (enp0s8)
Intel Corporation PRO/1000 MT De
Ethernet (enp0s9)
Intel Corporation PRO/1000 MT De

Ethernet (enp0s8)
Connected

Editing enp0s8

Connection name: enp0s8

IPv6 Settings

Method: Manual

Addresses

Address	Prefix	Gateway
2001:db8:1:1::1234:5678	64	2001:db8:1:1::fedc:abcd

DNS servers: 2001:db8:1:1::fedc:abce

Search domains:

☐ Require IPv6 addressing for this connection to complete

Figure 1: Manual Configuration of an IPv6 address, the default IPv6 gateway and an IPv6 DNS server in a Red Hat 7 system during installation.

After enabling the interface (ON), select the "IPv6 Settings" tab, and from the "Method" scroll-down menu the "Manual" option. Press "Add" and set your desired IPv6 static address, the prefix, the Gateway and, at the bottom, set the DNS server.

Still, please make sure to repeat the steps of section 4.3.

4.7 Defining Manually Static Routes

If you want to define some static IPv6 routes, you can do so using the following two files:

- In Red Hat:

  `/etc/sysconfig/static-routes-ipv6`

  ```
  #Device    IPv6 network to route    IPv6 gateway address
  eth0       fec0:0:0:2::/64          fec0:0:0:1:0:0:0:20
  eth0       2000::/3                 3ffe:ffff:0000:f102:0:0:0:1
  ```

- In Suse:

  `/etc/sysconfig/network-scripts/route6-<interface-name>`

  ```
  #This is a newer support of static routes per interface.
  # Example
  fec0:0:0:2::/64 via fec0:0:0:1:0:0:0:20 dev eth0
  ```

NOTE: Normal default gateway or device shouldn't be specified here, use options *DEFAULTGW* or *DEFAULTDEV* instead.

4.8 Use of IPv6 Privacy Extensions and Temporary Addresses

In this document it is described how to statically configure system's IPv6 addresses, so, normally this configuration should not be needed. However, in case that at some systems autoconfiguration is allowed (although not recommended for our context here), automatically-generated IPv6 addresses are based on the MAC addresses of the underlying hardware (e.g. Ethernet). Thus, it becomes possible to track the specific network interface consequently, the host, over its lifetime. If it is important for a system's IP address to not trivially reveal its hardware address, the use of privacy extensions must be enabled.

To enable IPv6 Privacy Extensions:

1. As root, edit the following file:

   ```
   /etc/sysconfig/network-scripts/ifcfg-<IFACE>:
   ```

2. Add the following line:

   ```
   IPV6_PRIVACY=rfc3041
   ```

Moreover, if for some reason you need to enable temporary addresses (e.g. for interface enp0s8), then the following *sysctl* parameters must be configured:

1. As root, edit the following file and then add the lines described below:

   ```
   /etc/sysctl.conf
   ```

2. Enable Privacy Extensions and prefer temporary addresses over public addresses:

   ```
   net.ipv6.conf.enp0s8.use_tempaddr = 2
   ```

3. Set preferred lifetime (in seconds) for temporary addresses (1 day):

   ```
   net.ipv6.conf.enp0s8.temp_prefered_lft = 43200
   ```

4. Set valid lifetime (in seconds) for temporary addresses (1/2 day).

   ```
   net.ipv6.conf.enp0s8.temp_valid_lft = 86400
   ```

5. Set the maximum number of auto-configured addresses per interface to 1:

   ```
   net.ipv6.conf.enp0s8.max_addresses = 1
   ```

Again, the same procedure should be repeated for all interfaces.

5 Further Hardening of IPv6 Hosts

5.1 Disabling DAD

Since the addresses are statically configured and to avoid any DAD-related attacks, DAD should also be disabled:

1. As root, edit the following file and then add the lines described below:

   ```
   /etc/sysctl.conf
   ```

2. Configure not to accept DAD:

   ```
   net.ipv6.conf.all.accept_dad = 0

   net.ipv6.conf.default.accept_dad = 0

   net.ipv6.conf.enp0s8.accept_dad = 0
   ```

3. Configure not to transmit DAD.

   ```
   net.ipv6.conf.all.dad_transmits = 0

   net.ipv6.conf.default.dad_transmits = 0

   net.ipv6.conf.enp0s8.dad_transmits = 0
   ```

5.2 Configuring Host Behaviour For Routing Advertisement Parameters

Usually the following are not needed, since they should have been taken care by the procedures described in section 4.3. They are presented here for reasons of completeness.

1. As root, edit the following file and then add the lines described below:

   ```
   /etc/sysctl.conf
   ```

2. Configure number of Router Solicitations to send until assuming no routers are present.

   ```
   net.ipv6.conf.default.router_solicitations = 0
   ```

3. Configure not accepting Router Preferences in RAs:

   ```
   net.ipv6.conf.default.accept_ra_rtr_pref = 0
   ```

4. Configure not to learn Prefix Information from Router Advertisement:

   ```
   net.ipv6.conf.default.accept_ra_pinfo = 0
   ```

5. Configure not to learn the default gateway from RAs

   ```
   net.ipv6.conf.default.accept_ra_defrtr = 0
   ```

6. Disable autoconf:

   ```
   net.ipv6.conf.default.autoconf = 0
   ```

5.3 Creating Static Entries for the Neighbor Cache Table

To mitigate the attacks related with the Neighbor Discovery process (man-in-the-middle, DoS, etc.), we can create static entries to the Neighbor Cache table. To this end, we will use the following *ip neigh* options:

- `ip neighbour replace` : add a new entry or change an existing one

  to ADDRESS (default): the protocol address of the neighbour. It is either an IPv4 or IPv6 address.

  dev NAME : the interface to which this neighbour is attached.

  lladdr LLADDRESS : the link layer address of the neighbour. LLADDRESS can also be null.

  nud NUD_STATE : the state of the neighbour entry. *nud* is an abbreviation for 'Neighbour Unreachability Detection'. The state can take one of the following values:

  permanent : the neighbour entry is valid forever and can be only be removed administratively.

For instance, to create an entry for IPv6 address 2001:db8:1:1:800:27ff:fe00:0 having as MAC address 0a:00:27:00:00:00 via device eth1, we can run as root the following command:

```
ip -6 neigh replace 2001:db8:1:1:800:27ff:fe00:0 lladdr 0a:00:27:00:00:00 dev eth1 nud permanent
```

If applied successfully, we should see the following:

```
# ip -6 neigh show

2001:db8:1:1:800:27ff:fe00:0 dev eth1 lladdr 0a:00:27:00:00:00 PERMANENT
```

Of course, we must repeat the above command:

- For all the neighbors for which we want to create static entries/
- To the other local end-nodes too.

If we do so for all the nodes at the local link, Neighbor Solicitation / Advertisement process can be blocked.

In order to load such entries automatically during start-up, we can create a script (e.g. *static_neigh.sh*) and add it in the */etc/rc.local* file in Red Hat or in */etc/init.d/boot.local* in Suse.

5.4 Other Configurations

Again, as root, edit file /etc/sysctl.conf and then add the lines described below:

- Set a common initial hop-limit for sent IPv6 packets:

  ```
  net.ipv6.conf.all.hop_limit=64
  ```

  NOTE: A hop-limit of 64 should be the default value, anyway.

- Reduce the effect of spoofed Packet Too Big messages be reducing the time (in seconds) in which a known path MTU expires:

  ```
  net.ipv6.route.mtu_expires = 60
  ```

- Force MLDv2. Although it is more complicated than MLDv1, it is more difficult to be abused than version 1 and hence, it should be preferred, if MLD is not block. Warning: Make sure that there is no MLDv1-only device in your network.

  ```
  net.ipv6.conf.all.force_mld_version = 2
  ```

- Maximise MLDv2 unsolicited reports interval:

  ```
  net.ipv6.conf.all.mldv2_unsolicited_report_interval=100000000000
  ```

  (not allowed to be changed in Red Hat 6.6).

- Make sure that multicast routing is disabled:

  ```
  net.ipv6.conf.all.mc_forwarding=0
  ```

  (not available in SUSE, not allowed to be changed in Red Hat 6.6).

- Restrict the use of the IPv6 socket to IPv6 communication only (not to use IPv4-mapped address feature):

  ```
  net.ipv6.conf.all.bindv6only = 1
  ```

- Make sure that forwarding is disabled (should be the case by default) by running the following commands:

  ```
  # sysctl net.ipv6.conf.all.forwarding

  net.ipv6.conf.all.forwarding = 0

  # sysctl net.ipv6.conf.default.forwarding

  net.ipv6.conf.default.forwarding = 0

  # sysctl net.ipv6.conf.enp0s8.forwarding

  net.ipv6.conf.enp0s8.forwarding = 0
  ```

  If this is not the case (e.g. the above commands return 1 instead of 0), add these lines to */etc/sysctl.conf* too.

  ```
          net.ipv6.conf.all.forwarding = 0

          net.ipv6.conf.default.forwarding = 0

          net.ipv6.conf.enp0s8.forwarding = 0
  ```

6 CONFIGURING THE HOST FIREWALL

6.1 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):

```
#DROP MULTICAST TRAFFIC GOING TO ALL-NODES LINK-LOCAL

ip6tables -I INPUT 1 -d ff02::1 -j DROP
```

6.2 ICMPv6

6.2.1 Incoming ICMPv6

Allow the following ICMPv6 incoming types of messages:

- Packet Too Big.

```
# PACKET TOO BIG ERROR MESSAGES

# ==============================

# Allow incoming Packet Too Big messages

ip6tables -A INPUT -p icmpv6 --icmpv6-type packet-too-big -j ACCEPT
```

- Destination Unreachable

```
# DESTINATION UNREACHABLE ERROR MESSAGES

# ========================================

# Allow incoming destination unreachable messages only for existing sessions

ip6tables -A INPUT -m state -p icmpv6 --state ESTABLISHED,RELATED --icmpv6-type
destination-unreachable -j ACCEPT
```

- Echo Replies

```
# ECHO REPLIES

# =============

# Allow incoming echo reply messages only for existing sessions

ip6tables -A INPUT -m state -p icmpv6 --state ESTABLISHED,RELATED --icmpv6-type
echo-reply -j ACCEPT
```

- Time Exceeded (Type 3 Code 0)

```
# TIME EXCEEDED ERROR MESSAGES

# ============================

# Allow incoming time exceeded code 0 and 1 messages only for existing sessions

ip6tables -A INPUT -m state -p icmpv6 --state ESTABLISHED,RELATED --icmpv6-type
ttl-zero-during-transit -j ACCEPT

ip6tables -A INPUT -m state -p icmpv6 --state ESTABLISHED,RELATED --icmpv6-type
time-exceeded -j ACCEPT
```

- Parameter Problem (Type 4 Codes 1 and 2)

```
#PARAMETER PROBLEM ERROR MESSAGES

# ================================

# Allow incoming parameter problem code 1 and 2 messages for an existing session

ip6tables -A INPUT -m state -p icmpv6 --state ESTABLISHED,RELATED --icmpv6-type
unknown-header-type -j ACCEPT

ip6tables -A INPUT -m state -p icmpv6 --state ESTABLISHED,RELATED --icmpv6-type
unknown-option -j ACCEPT
```

- For network troubleshooting purposes, you can allow Echo Requests messages from very specific(s) hosts (e.g. admins' hosts):

```
# ECHO REQUESTS

# ===========================

# Allow inbound echo requests only from predetermined hosts and only destined to
specific (our) host address(es)

for trusted_host in ${TRUSTED_HOSTS[@]}

do

        for host_address in ${HOST_ADDRESSES[@]}

        do

                ip6tables -A INPUT -p icmpv6 -s $trusted_host -d $host_address  --
icmpv6-type echo-request -m limit --limit 900/min -j ACCEPT

        done

done
```

6.2.2 Outgoing ICMPv6

Allow the following ICMPv6 outgoing messages

- Packet Too Big

```
# PACKET TOO BIG ERROR MESSAGES

# ==============================

# Allow outgoing Packet Too Big messages

ip6tables -A OUTPUT -p icmpv6 --icmpv6-type packet-too-big -j ACCEPT
```

- Echo Requests

```
# ECHO REQUESTS

# ===========================

# Allow outbound echo requests

for host_address in ${HOST_ADDRESSES[@]}

do

        ip6tables -A OUTPUT -p icmpv6 -s $host_address --icmpv6-type echo-request
        -j ACCEPT
```

```
done
```

- Echo Replies

```
# Allow outgoing echo reply messages
for host_address in ${HOST_ADDRESSES[@]}
do
  ip6tables -A OUTPUT -p icmpv6 -s $host_address --icmpv6-type echo-reply -j ACCEPT
done
```

6.2.3 Neighbor Solicitation / Advertisement Messages

In case we configure the Neighbor Cache entries correctly to ALL our link local hosts (as described in section 5.3), ICMPv6 Neighbor Solicitation / Advertisement messages can be blocked to prevent Niegbor Discovery related attacks (such as Man-in-the-middle, or DoS). In a different case, the following ip6table rules must also be added:

- Neighbor Solicitation and Neighbor Advertisement messages. Required to perform the Neighbor Discovery Process.

```
# NEIGHBOR DISCOVERY MESSAGES
# ===========================
# Allow NS/NA messages both incoming and outgoing
ip6tables -A INPUT -p icmpv6 --icmpv6-type neighbor-solicitation -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type neighbor-advertisement -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type neighbor-solicitation -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type neighbor-advertisement -j ACCEPT
```

6.2.4 Default Policy

All the rest ICMPv6 traffic should be blocked

```
# DROP ANY OTHER ICMPV6 TRAFFIC
# =============================
ip6tables -A OUTPUT -p icmpv6 -j DROP
ip6tables -A INPUT -p icmpv6 -j DROP
```

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).

The corresponding options given by the ip6tables, are the following:

ipv6header: This module matches IPv6 extension headers and/or upper layer header.

--soft Matches if the packet includes any of the headers specified with *--header*.

[!] --header header[,header...]

Matches the packet which EXACTLY includes all specified headers (NOTE: The headers encapsulated with ESP header are out of scope).

Possible header types can be:

hop	Hop-by-Hop Options header
dst	Destination Options header
route	Routing header
frag	Fragment header
auth	Authentication header
esp	Encapsulating Security Payload header
none	No Next header which matches 59 in the 'Next Header field' of IPv6 header or any IPv6 extension headers

The IPv6 Extension Headers blocking rule should be added before any other rule.

Examples:

```
#BLOCK EXTENSION HEADERS

ip6tables -I INPUT 1 -i eth1 -m ipv6header --header dst --soft -j DROP

ip6tables -I INPUT 2 -i eth1 -m ipv6header --header hop --soft -j DROP

ip6tables -I INPUT 3 -i eth1 -m ipv6header --header route --soft -j DROP

ip6tables -I INPUT 4 -i eth1 -m ipv6header --header frag --soft -j DROP

ip6tables -I INPUT 5 -i eth1 -m ipv6header --header auth --soft -j DROP

ip6tables -I INPUT 6 -i eth1 -m ipv6header --header esp --soft -j DROP

ip6tables -I INPUT 7 -i eth1 -m ipv6header --header none --soft -j DROP
```

6.3.1 Blocking Deprecated Extension Headers

In our scenario we block ALL IPv6 Extension Headers. If, on the other hand,, we need to allow Routing Headers, we must block the deprecated Type-0 one. Example:

```
# Drop packets with a type 0 routing header

 ip6tables -A INPUT -m rt --rt-type 0 -j DROP

 ip6tables -A OUTPUT -m rt --rt-type 0 -j DROP

 ip6tables -A FORWARD -m rt --rt-type 0 -j DROP
```

7 Pros and Cons

In a nutshell, in order to perform all the suggested configurations:

1. First of all, we need to modify the following configuration files:

   `/etc/sysctl.conf`

   `/etc/resolv.conf`

   `/etc/sysconfig/static-routes-ipv6`

   `/etc/sysconfig/network-scripts/ifcfg-<IFACE>`

   It is expected that the first three of them will be usually the same per host machine of the same local-link network. For this reason, they can be easily copied using *ssh* to as many machine as we want using an automated script.

   Moreover, even the last one, although will differ from host to host, the required changes will be the same, except from the IPv6 address (as an example, please see the bold fonts in Appendix B). So, even in this case, most of the work can be automated using a script which will append this extra info to the corresponding files. For convenience reasons, we can make sure that each network interface has the same name (e.g. *enp0s8*) in each host.
2. Then, we also need to create the necessary ip6tables to apply and save the corresponding rules. Again, *ssh* and automating the process using a script is your best friend.
3. Finally, the most challenging task is to add permanent entries in the neighbor cache of each system. These entries will be the same in each system except from the IPv6 address of the host system itself which must be excluded. However, in order to automatically reload these permanent entries during each booting process, a suitable script must be loaded by */etc/rc.local* or any equivalent file in other distribution.

Easy? It depends on your scripting skills. Certainly, it is much more complicated than stateless or stateful autoconfiguration, but it is still feasible. But, what are the benefits from all the above semi-manual autoconfiguration? Well, after applying correctly all of the above, tailored in your environment of course, the attacks of section 2.2 can be mitigated. You can control closely your IPv6 environment and prevent any related attacks. Furthermore, you can extend for instance the ip6table rules not only to block unwanted traffic, like Router Advertisement messages, but, to also log them. By doing so, accidental unwanted or malicious actions like spoofed RAs can be easily detected, increasing not only the preventing mechanisms in your environment but also the early-warning capabilities regarding IPv6 attacks.

In summary, certainly there are benefits regarding security from the described semi-manual configurations but some good scripting skills, and some extra effort are required in order to apply them.

8 REFERENCES

[1] https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt

[2] man ip-neighbour

[3] man ip6tables

9 APPENDIX A: EXAMPLE OF A SYSCTL.CONF FILE

```
# System default settings live in /usr/lib/sysctl.d/00-system.conf.

# To override those settings, enter new settings here, or in an
/etc/sysctl.d/<name>.conf file

# For more information, see sysctl.conf(5) and sysctl.d(5).

net.ipv6.bindv6only=1

net.ipv6.route.mtu_expires = 60

net.ipv6.conf.all.forwarding = 0

net.ipv6.conf.default.forwarding = 0

net.ipv6.conf.enp0s8.forwarding = 0

net.ipv6.conf.all.accept_ra=0

net.ipv6.conf.default.accept_ra=0

net.ipv6.conf.enp0s8.accept_ra=0

net.ipv6.conf.all.accept_redirects=0

net.ipv6.conf.default.accept_redirects=0

net.ipv6.conf.enp0s8.accept_redirects=0

net.ipv6.conf.all.autoconf = 0

net.ipv6.conf.default.autoconf = 0

net.ipv6.conf.enp0s8.autoconf = 0

net.ipv6.conf.all.accept_dad = 0

net.ipv6.conf.default.accept_dad = 0

net.ipv6.conf.enp0s8.accept_dad = 0

net.ipv6.conf.all.dad_transmits = 0

net.ipv6.conf.default.dad_transmits = 0

net.ipv6.conf.enp0s8.dad_transmits = 0

net.ipv6.conf.all.hop_limit=64

net.ipv6.conf.default.hop_limit=64

net.ipv6.conf.enp0s8.hop_limit=64

net.ipv6.conf.all.force_mld_version = 2

net.ipv6.conf.default.force_mld_version = 2

net.ipv6.conf.enp0s8.force_mld_version = 2

net.ipv6.conf.all.mldv2_unsolicited_report_interval=100000000000

net.ipv6.conf.default.mldv2_unsolicited_report_interval=100000000000

net.ipv6.conf.enp0s8.mldv2_unsolicited_report_interval=100000000000

net.ipv6.conf.all.mc_forwarding=0

net.ipv6.conf.default.mc_forwarding=0
net.ipv6.conf.enp0s8.mc_forwarding=0
```

10 Appendix B: Example of an IFACE Configuration File

Example file: */etc/sysconfig/network-scripts/ifcfg-enp0s8*

```
TYPE=Ethernet

BOOTPROTO=dhcp

DEFROUTE=yes

IPV4_FAILURE_FATAL=no

IPV6INIT=yes

IPV6_AUTOCONF=no

IPV6_DEFROUTE=yes

IPV6_FAILURE_FATAL=no

NAME=enp0s8

UUID=5a4262f6-adb1-4687-b2f2-30e7042535f3

ONBOOT=yes

HWADDR=08:00:27:2B:F8:BF

PEERDNS=yes

PEERROUTES=yes

IPV6ADDR=2001:db8:1:1::1234:5678/64

IPV6_DEFAULTGW=2001:db8:1:1:7885:9a1b:c3cc:f818

DNS1=2001:db8:1:1::fedc:abce
```

11 Appendix C: ip6tables Example

```
#!/bin/bash


#FLUSH CHAIN RULES

ip6tables -F INPUT

ip6tables -F OUTPUT

ip6tables -F FORWARD


#SET DEFAULT POLICY RULES

ip6tables -P INPUT DROP

ip6tables -P FORWARD DROP

ip6tables -P OUTPUT ACCEPT


# Set of trusted hosts to be allowed for some selective input traffic

export TRUSTED_HOSTS=("2001:db8:1:1:605e:d8e2:2c32:762/128")

# Set of addresses of this host

export HOST_ADDRESSES=("2001:db8:1:1::1234:5678/128")


#DROP MULTICAST TRAFFIC GOING TO ALL-NODES LINK-LOCAL

ip6tables -I INPUT 1 -d ff02::1 -j DROP


#BLOCK EXTENSION HEADERS

ip6tables -I INPUT 2 -i eth1 -m ipv6header --header dst --soft -j DROP

ip6tables -I INPUT 3 -i eth1 -m ipv6header --header hop --soft -j DROP

ip6tables -I INPUT 4 -i eth1 -m ipv6header --header route --soft -j DROP

ip6tables -I INPUT 5 -i eth1 -m ipv6header --header frag --soft -j DROP

ip6tables -I INPUT 6 -i eth1 -m ipv6header --header auth --soft -j DROP

ip6tables -I INPUT 7 -i eth1 -m ipv6header --header esp --soft -j DROP

ip6tables -I INPUT 8 -i eth1 -m ipv6header --header none --soft -j DROP


# ECHO REQUESTS AND REPLIES

# ===========================

# Allow inbound echo requests only from predetermined hosts and only destined to
(our) host address(es)

for trusted_host in ${TRUSTED_HOSTS[@]}

do
```

```
        for host_address in ${HOST_ADDRESSES[@]}
        do
                ip6tables -A INPUT -p icmpv6 -s $trusted_host -d $host_address  --
                icmpv6-type echo-request -m limit --limit 900/min -j ACCEPT
        done
done
# Allow outbound echo requests
for host_address in ${HOST_ADDRESSES[@]}
do
        ip6tables -A OUTPUT -p icmpv6 -s $host_address --icmpv6-type echo-request
        -j ACCEPT
done
# Allow incoming echo reply messages only for existing sessions
ip6tables -A INPUT -m state -p icmpv6 --state ESTABLISHED,RELATED --icmpv6-type
echo-reply -j ACCEPT
# Allow outgoing echo reply messages
for host_address in ${HOST_ADDRESSES[@]}
do
        ip6tables -A OUTPUT -p icmpv6 -s $host_address --icmpv6-type echo-reply -j
        ACCEPT
done

# DESTINATION UNREACHABLE ERROR MESSAGES
# ======================================
# Allow incoming destination unreachable messages only for existing sessions
ip6tables -A INPUT -m state -p icmpv6 --state ESTABLISHED,RELATED --icmpv6-type
destination-unreachable -j ACCEPT

# PACKET TOO BIG ERROR MESSAGES
# =============================
# Allow incoming and outgoing Packet Too Big messages
ip6tables -A INPUT -p icmpv6 --icmpv6-type packet-too-big -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type packet-too-big -j ACCEPT

# TIME EXCEEDED ERROR MESSAGES
# ============================
# Allow incoming time exceeded code  0 and 1 messages only for existing sessions
```

```
ip6tables -A INPUT -m state -p icmpv6 --state ESTABLISHED,RELATED --icmpv6-type
ttl-zero-during-transit -j ACCEPT

ip6tables -A INPUT -m state -p icmpv6 --state ESTABLISHED,RELATED --icmpv6-type
time-exceeded -j ACCEPT


# PARAMETER PROBLEM ERROR MESSAGES

# ================================

# Allow incoming parameter problem code 1 and 2 messages for an existing session

ip6tables -A INPUT -m state -p icmpv6 --state ESTABLISHED,RELATED --icmpv6-type
unknown-header-type -j ACCEPT

ip6tables -A INPUT -m state -p icmpv6 --state ESTABLISHED,RELATED --icmpv6-type
unknown-option -j ACCEPT


# DROP ANY OTHER ICMPV6 TRAFFIC

# =============================

ip6tables -A OUTPUT -p icmpv6 -j DROP

ip6tables -A INPUT -p icmpv6 -j DROP #Not actually required due to the default
deny policy for INPUT chain


#ADD BELOW THIS LINE ALL THE SPECIFIC RULES REQUIRED FOR THE SERVICE DELIVERED BY
EACH SERVER
```