IPv6 Hardening Guide for OS-X

How to Configure Mac OS-X to Prevent IPv6-related Attacks

Version:	1.0
Date:	29/01/2015
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.

1.1 Document Status and Owner

Title:	IPv6 – How to Configure Mac OS-X to Prevent IPv6-related Attacks
Document Owner:	ERNW GmbH
Version:	1.0
Status:	Effective
Classification:	Public
Author(s):	Antonios Atlasis

2 INTRODUCTION

2.1 Goal of this Paper

The goal of this paper 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 extend.

2.2 Tested Platforms

The examined system is an OS X Yosemite, version 10.10.1, fully patched, as of Jan 2015.

Figure 1: The Tested Operating System.

Some more information regarding the kernel:

bash-3.2# ***uname -a***
Darwin TROOPERs-MacBook-Air-2.local 14.0.0 Darwin Kernel Version 14.0.0: Fri Sep 19 00:26:44 PDT 2014; root:xnu-2782.1.97~2/RELEASE_X86_64 x86_64

2.3 Preparation

Let's examine our setup and whether we have an IPv6 Connection or not.

bash-3.2# ***networksetup -listallnetworkservices***
An asterisk (*) denotes that a network service is disabled.
Bluetooth DUN
USB Ethernet
Wi-Fi
Bluetooth PAN

Thunderbolt Bridge

Since I'm using a USB Ethernet adapter to connect to my LAN, I m' further checking this device.

bash-3.2# ***networksetup -getinfo USB\ Ethernet***
DHCP Configuration
IP address: 192.168.1.61
Subnet mask: 255.255.255.128
Router: 192.168.1.99
Client ID:
IPv6: Automatic
IPv6 IP address: none
IPv6 Router: none
Ethernet Address: 58:55:ca:24:93:3d

NOTE: The above command does not display our IPv6 address, although there is one. Let's check it using the ifconfig command.

bash-3.2# ifconfig
... <snipped for brevity> ...
en3: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=4<VLAN_MTU>
ether 58:55:ca:24:93:3d
inet6 fe80::5a55:caff:fe24:933d%en3 prefixlen 64 scopeid 0x9
inet6 2a02:2149:8127:e700:5a55:caff:fe24:933d prefixlen 64 autoconf
inet6 2a02:2149:8127:e700:c03a:cb03:1059:8856 prefixlen 64 autoconf temporary
inet 192.168.1.61 netmask 0xffffff80 broadcast 192.168.1.127
nd6 options=1<PERFORMNUD>
media: autoselect (100baseTX <full-duplex,flow-control>)
status: active

NOTE: As we can observe, IPv6 Privacy Extensions for Stateless Address Autoconfiguration - SLAAC (RFC 4941) are enabled by default, which, from the users' perspective is actually good. In case organizations want to track, e.g. for security reasons, the hosts connected to their networks, they must use solutions like an IPAM system[1].

[1] *See also http://www.insinuator.net/2015/01/evaluation-of-ipv6-capabilities-of-commercial-ipam-solutions/.*

3 LIMITING IPV6 (TO THE LINK-LOCAL)

In the improbable and particular case you do not want a global IPv6 connectivity but, instead, you want to limit IPv6 operation at the link-local, you have, as in most of the cases, two options: To either use the command line (CLI), or the GUI.

Using the CLI, we run the following command:

bash-3.2# ***networksetup -setv6linklocal USB| Ethernet***

Now, let's check our interface again:

bash-3.2# ***ifconfig en3***
en3: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=4<VLAN_MTU>
ether 58:55:ca:24:93:3d
inet6 fe80::5a55:caff:fe24:933d%en3 prefixlen 64 scopeid 0x9
inet 192.168.1.61 netmask 0xffffff80 broadcast 192.168.1.127
nd6 options=1<PERFORMNUD>
media: autoselect (100baseTX <full-duplex,flow-control>)
status: active

Indeed, the global unicast IPv6 address has gone.

Of course, you can disable IPv6 completely (e.g. if IPv6 is not available from your ISP yet).

bash-3.2# ***networksetup -setv6off USB| Ethernet***
bash-3.2# ***ifconfig en3***
en3: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=4<VLAN_MTU>
ether 58:55:ca:24:93:3d
inet 192.168.1.61 netmask 0xffffff80 broadcast 192.168.1.127
nd6 options=1<PERFORMNUD>
media: autoselect (100baseTX <full-duplex,flow-control>)
status: active

Alternatively, you could use the GUI:

Go to *System Preferences → Network*.

For all relevant interfaces click *Advanced*....

Then, choose *TCP/IP* and *Configure IP6: Link-Local only*

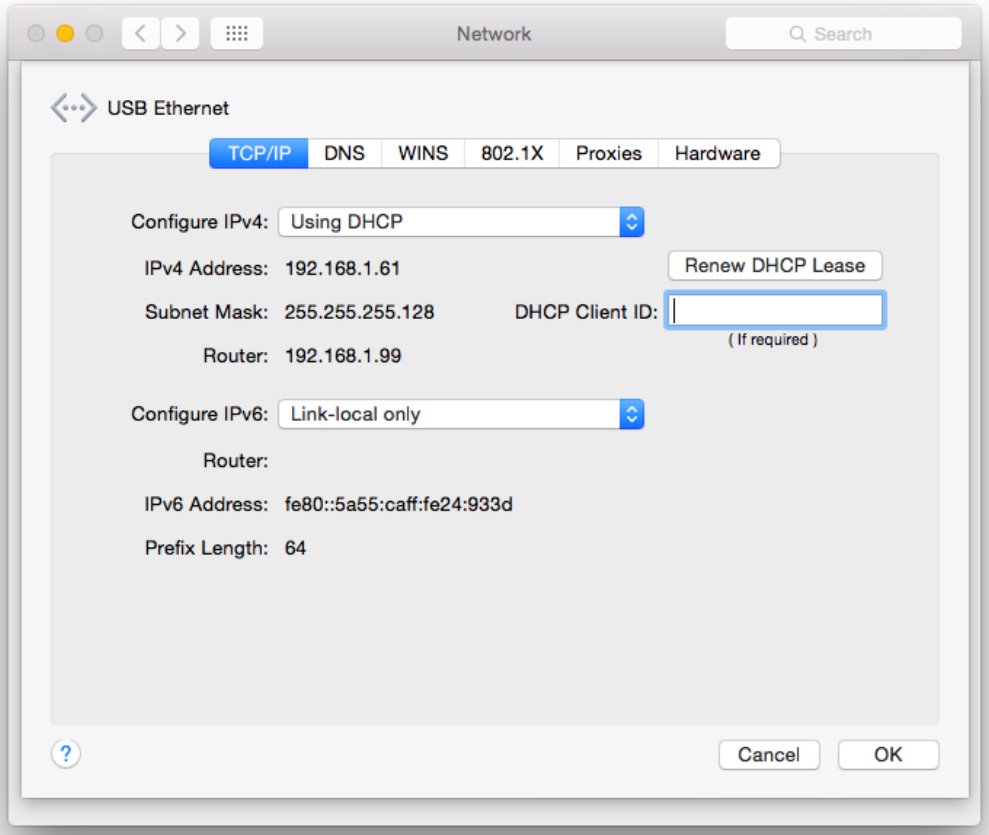

Figure 2: Enabling IPv6 Link-Local Only from the GUI

To enable IPv6 globally again (if you regretted it), run the following command:

bash-3.2# ***networksetup -setv6automatic USB| Ethernet***
bash-3.2# ***ifconfig en3***
en3: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=4<VLAN_MTU>
ether 58:55:ca:24:93:3d
inet6 fe80::5a55:caff:fe24:933d%en3 prefixlen 64 scopeid 0x9
inet 192.168.1.61 netmask 0xffffff80 broadcast 192.168.1.127
inet6 2a02:2149:8127:e700:5a55:caff:fe24:933d prefixlen 64 autoconf
inet6 2a02:2149:8127:e700:ecef:bceb:439:8fc4 prefixlen 64 optimistic autoconf temporary
nd6 options=1<PERFORMNUD>
media: autoselect (100baseTX <full-duplex,flow-control>)
status: active

4 MANUAL CONFIGURATION

A common approach to harden a system is to disable as much automatic configuration as possible and try to configure the corresponding settings manually.

4.1 Disabling Privacy Extensions

As we already said, privacy addresses are enabled by default. Sometimes for security reasons it is needed to track down physical systems, which is rather difficult using IPv6 addresses with Privacy Extensions enabled. To disable them permanently, edit /etc/sysctl.conf (if it doesn't exist, create it) and set:

net.inet6.ip6.use_tempaddr=0

and reboot.

Or, to disable privacy addresses for the current session only, run the command:

bash-3.2# ***sysctl -w net.inet6.ip6.use_tempaddr=0***
net.inet6.ip6.use_tempaddr: 1 -> 0

Of course, you can combine both ways (writing the configuration to sysctl.conf and using the sysctl command) if we both want to make the changes permanent and avoid rebooting our system without a reason. We can apply the same double approach in every case we will configure a kernel (sysctl) variable.

You can also disable the preference to temporary addresses, as following:

sysctl -w net.inet6.ip6.prefer_tempaddr=0

Furthermore, to (optionally) set the preferred lifetime of the temporary address (in seconds), we can use the following command (default value=86400 sec):

sysctl -w net.inet6.ip6.temppltime=XX

We can also set the valid lifetime of the temporary address by setting appropriately the following parameter (default value: 604800):

net.inet6.ip6.tempvltime=XX

4.2 Configure Host's IPv6 address and the Default Gateway (Router) Statically

From the GUI, you can go to *System Preferences → Network*. For all relevant interfaces click *Advanced*

Then, choose *TCP/IP* and *Configure IP6: Manually.*

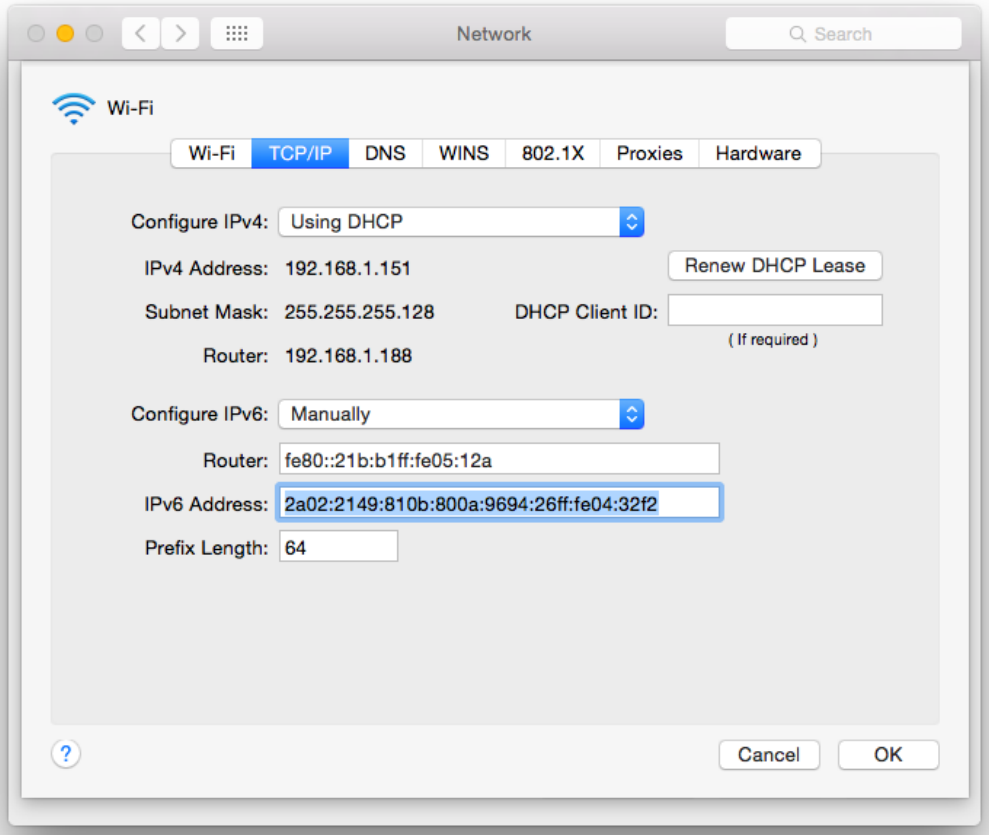

Figure 3: Configuring the IPv6 host address and the default router manually.

You can also do the same by using the CLI:

bash-3.2# ***networksetup -setv6manual USB\ Ethernet 2a02:2149:8127:e700:5a55:caff:fe24:933d 64***
fe80::20d:b9ff:fe28:c214

where fe80::20d:b9ff:fe28:c214 is the router IPv6 link-local address

To verify it:

bash-3.2# ***networksetup -getinfo USB\ Ethernet***
DHCP Configuration
IP address: 192.168.1.61
Subnet mask: 255.255.255.128
Router: 192.168.1.99
Client ID:
IPv6: Manual
IPv6 IP address: 2a02:2149:8127:e700:5a55:caff:fe24:933d
IPv6 Router: fe80::20d:b9ff:fe28:c214
IPv6 Prefix Length: 64
Ethernet Address: 58:55:ca:24:93:3d

As we can see, when using static configuration for IPv6 addresses, the networksetup -getinfo command displays the IPv6 address of the host and of the default gateway.

4.3 Configure Additional Static IPv6 Routes

Before we set-up a new static route, let's examine first the already available ones by using the following command:

```
bash-3.2# netstat -rnf inet6
Routing tables
Internet6:
Destination                    Gateway                    Flags      Netif Expire
default                        fe80::20d:b9ff:fe28:c214%en3  UGc         en3
::1                            ::1                        UHL         lo0
2a02:2149:8614:f700::/64       link#4                     UC          en3
2a02:2149:8614:f700:20d:b9ff:fe28:c214  0:d:b9:28:c2:14           UHLWI       en3
2a02:2149:8614:f700:5a55:caff:fe24:933d 58:55:ca:24:93:3d          UHL         lo0
2a02:2149:8614:f700:e12e:a5d2:bc22:57c0 58:55:ca:24:93:3d          UHL         lo0
fe80::%lo0/64                  fe80::1%lo0                UcI         lo0
fe80::1%lo0                    link#1                     UHLI        lo0
fe80::%en3/64                  link#4                     UCI         en3
fe80::20d:b9ff:fe28:c214%en3   0:d:b9:28:c2:14            UHLWIir     en3
fe80::5a55:caff:fe24:933d%en3  58:55:ca:24:93:3d          UHLI        lo0
ff01::%lo0/32                  ::1                        UmCI        lo0
ff01::%en3/32                  link#4                     UmCI        en3
ff01::%en0/32                  link#5                     UmCI        en0
ff02::%lo0/32                  ::1                        UmCI        lo0
ff02::%en3/32                  link#4                     UmCI        en3
ff02::%en0/32                  link#5                     UmCI        en0
```

To setup an additional static route, we will use the following command:

*networksetup -setv6additionalroutes <networkservice> [<dest> <prefixlength> <gateway>]**

Example:

*bash-3.2# **networksetup -setv6additionalroutes USB\ Ethernet 2001:db8:1:1:: 64 2001:db8:1:1::1***

Let's see again the available routes:

```
bash-3.2# netstat -rnf inet6
Routing tables
Internet6:
Destination                    Gateway                    Flags      Netif Expire
default                        fe80::20d:b9ff:fe28:c214%en3  UGc         en3
::1                            ::1                        UHL         lo0
2001:db8:1:1::/64              2001:db8:1:1::1            UGSc        en3
...<snipped for brevity> ...
```

As we can see, the routed that we wanted has been added.

4.4 Configure Manually the MTU

Sometimes, we need to manually configure the MTU of the local link.

To do so, first check the already used MTU size:

bash-3.2# networksetup -getMTU USB| Ethernet
Active MTU: 1500 (Current Setting: 1500)

Now, let's define our desired MTU size (e.g. 1280 in our example).

*bash-3.2# **networksetup -setMTU USB| Ethernet 1280***
*bash-3.2# **networksetup -getMTU USB| Ethernet***
Active MTU: 1280 (Current Setting: 1280)

As we can see, the MTU for our used interface have been successfully defined.

5 Configuring Some IPv6 sysctl Parameters

Although not clearly documented, there are several sysctl parameters that can be used to harden IPv6. To find them, you can simply run:

sysctl -A | grep inet6

There should be eighty-four (84) inet6 parameters in total.

However, to find even the smallest description about them, we had to download and check the corresponding files of the kernel source code.

Moreover, in all the following examples, the *sysctl* command is used. To make the changes permanent, as already described in a previous section, you must add the described configuration to */etc/sysctl.conf*

A complete list of the available Ipv6-related sysctl parameters can be found in the Appendix.

5.1 Verify That Forwarding is Disabled

This should be the case by default, but it is never a bad idea to check that IPv6 forwarding is disabled and hence, it does not act as a router without knowing it. To do so, run:

bash-3.2# sysctl net.inet6.ip6.forwarding
net.inet6.ip6.forwarding: 0

Of course, this value should be set to zero (0).

5.2 (Trying to) Disable the Acceptance of Router Advertisements

To disable the acceptance of IPv6 Router Advertisements - RAs (and consequently, to mitigate the related attacks), it seems that we can use the *inet6.ip6.accept_rtadv* parameters.

bash-3.2# sysctl -w net.inet6.ip6.accept_rtadv=0
sysctl: oid 'net.inet6.ip6.accept_rtadv' is read only

So, it seems that, although there is the corresponding parameter, this cannot be altered because it is a read-only one. By checking again the kernel source code, this parameter is marked at the comments as "deprecated". We also added it to systl.conf to check if it works and rebooted the system, but Router Advertisements (Ras) were still accepted by OS-X. So, it seems that we cannot avoid RAs and consequently all the related attacks in this OS.

5.3 Configuring Other IPv6 Router Advertisement Parameters

Since we are not able to stop the acceptance of IPv6 RAs, let' see what other IPv6 RA-related parameters can be configured:

- Set the maximum number of acceptable prefixes via RAs per interface to one (1) – the default value is sixteen (16):

  bash-3.2# sysctl -w net.inet6.ip6.maxifprefixes=1
  net.inet6.ip6.maxifprefixes: 16 -> 1

- Set the maximum number of acceptable default routers via RAs to one (1) – the default value is sixteen (16).

bash-3.2# sysctl -w net.inet6.ip6.maxifdefrouters=1
net.inet6.ip6.maxifdefrouters: 16 -> 1

5.4 Disable the Acceptance of IPv6 Deprecated Addresses

To disable the acceptance of IPv6 deprecated addresses, you can run the following command:

bash-3.2# ***sysctl net.inet6.ip6.use_deprecated=0***
net.inet6.ip6.use_deprecated: 1 -> 0

For more info, please check RFC 4862, paragraph 5.5.4.

5.5 Eliminating the Acceptance of Header Options

Another interesting parameters is the one that can define how many header options can be processed. Of course, to eliminate them we need to set them to zero (0), which by default is set to 15:

bash-3.2# ***sysctl net.inet6.ip6.hdrnestlimit=0***
net.inet6.ip6.hdrnestlimit: 15 -> 0s

5.6 Enabling the Acceptance of ULAs Only

If you are in a lab environment and you want to allow only the acceptance of Unique Local IPv6 Unicast Addresses (ULAs) – RFC 4193 (for instance, to make sure that there is no chance to communicate with the real word out there, not even by …accident), you can run the following command:

bash-3.2# *sysctl -w net.inet6.ip6.only_allow_rfc4193_prefixes=1*
net.inet6.ip6.only_allow_rfc4193_prefixes: 0 -> 1

5.7 Disable DAD

You can also disable the Dupplicate Address Detection (DAD transmits by setting them to zero (0):

bash-3.2# ***sysctl -w net.inet6.ip6.dad_count=0***
net.inet6.ip6.dad_count: 1 -> 0

5.8 Disabling the Acceptance of ICMPv6 Node Info

Another good approach would be to also disable the acceptance of Node Information requests (RFC 4620). To do so, run:

bash-3.2# ***sysctl -w net.inet6.icmp6.nodeinfo=0***
net.inet6.icmp6.nodeinfo: 3 -> 0

5.9 Disable the Acceptance of ICMPv6 Redirects

To disable the acceptance of ICMPv6 Redirects (enable by default) run the following command:

bash-3.2# sysctl -w net.inet6.icmp6.rediraccept
net.inet6.icmp6.rediraccept: 1

You can also set the maximum number of routes created via redirect to zero (0) – default 1024.

bash-3.2# sysctl -w net.inet6.ip6.maxdynroutes=0
net.inet6.ip6.maxdynroutes: 1024 -> 0

5.10 Disable the Acceptance of Fragmented Packets

You can implicitly forbid the acceptance of fragmentation by setting the number of accepted fragments, as well as the number of accepted fragmented packets to zero (0):

bash-3.2# ***sysctl -w net.inet6.ip6.maxfrags=0***
net.inet6.ip6.maxfrags: 2048 -> 0
bash-3.2# ***sysctl -w net.inet6.ip6.maxfragpackets=0***
net.inet6.ip6.maxfragpackets: 1024 -> 0

5.11 Configuring MLD

There are some MLD-related parameters that:

a) First, they allow us to enable the use of MLDv1, MLDv2 or both (default). These aer the following.

net.inet6.mld.v1enable: 1
net.inet6.mld.v2enable: 1

b) Secondly, give the impression that can be used to disable MLD totally:

net.inet6.mld.use_allow: 1

Unfortunately, even when we unset all the above parameters, MLD was still used (and this is MLDv1). So, it seems that OS-X is yet another OS where you cannot disable MLD completely

Finally there is also another MLD-related sysctl parameter, the following one:

net.inet6.mld.debug: 0

As you can see, each default value is zero. Please make sure to remain so. When MLD is used in debugging mode, according to the related RFCs it allows the acceptance of MLD messages to the system's unicast (link-local) address, which is the cause of MLD-related attacks. So, please leave this parameter unset.

5.12 Setting the Neighbor Cache Threshold

You can also set the threshold of the Neighbor Cache entries by setting accordingly the neighborgcthresh parameter, the default value of which is 1024.

net.inet6.ip6.neighborgcthresh: 1024

There are known some attacks related with Neighbor Cache Exhaustion, and, depending on your environment (that is, how many hosts are around at the local link) and your system's memory, you can find a good compromise between the memory used and the entries required to fill your cache.

6 LIMITATIONS OF THE SUGGESTED APPROACH

The described approach has some limitations in comparison with the corresponding hardening guides of Linux and Windows servers, which can be summarized as following:

- The acceptance of Router Advertisements cannot be disabled using a sysctl parameter because, as explained, it has been deprecated.
- In this guide the neighbor cache was not configured manually (as in our corresponding Linux guide).
- The GUI of the OS-X firewall does not provide options for configuring the acceptance or not of IPv6 Extension Headers. Given the Desktop nature of OS-X, blocking all incoming connections (which is possible) is highly recommended.
- The application of the suggested configuration to a group of such machines has not been examined.

7 Appendix: List of IPv6-Related sysctl parameters (and their default values)

net.inet6.ip6.forwarding: 0
net.inet6.ip6.redirect: 1
net.inet6.ip6.hlim: 64
net.inet6.ip6.maxfragpackets: 1024
net.inet6.ip6.accept_rtadv: 1
net.inet6.ip6.keepfaith: 0
net.inet6.ip6.log_interval: 5
net.inet6.ip6.hdrnestlimit: 15
net.inet6.ip6.dad_count: 1
net.inet6.ip6.auto_flowlabel: 1
net.inet6.ip6.defmcasthlim: 1
net.inet6.ip6.gifhlim: 0
net.inet6.ip6.kame_version: 2009/apple-darwin
net.inet6.ip6.use_deprecated: 1
net.inet6.ip6.rr_prune: 5
net.inet6.ip6.v6only: 0
net.inet6.ip6.rtexpire: 3600
net.inet6.ip6.rtminexpire: 10
net.inet6.ip6.rtmaxcache: 128
net.inet6.ip6.use_tempaddr: 0
net.inet6.ip6.tempppltime: 86400
net.inet6.ip6.tempvltime: 604800
net.inet6.ip6.auto_linklocal: 1
net.inet6.ip6.prefer_tempaddr: 1
net.inet6.ip6.use_defaultzone: 0
net.inet6.ip6.maxfrags: 2048
net.inet6.ip6.mcast_pmtu: 0
net.inet6.ip6.neighborgcthresh: 1024
net.inet6.ip6.maxifprefixes: 16
net.inet6.ip6.maxifdefrouters: 16
net.inet6.ip6.maxdynroutes: 1024
net.inet6.ip6.fragpackets: 0
net.inet6.ip6.fw.enable: 1
net.inet6.ip6.fw.debug: 0
net.inet6.ip6.fw.verbose: 0
net.inet6.ip6.fw.verbose_limit: 0
net.inet6.ip6.scopedroute: 1
net.inet6.ip6.adj_clear_hwcksum: 0
net.inet6.ip6.maxchainsent: 0
net.inet6.ip6.select_srcif_debug: 0
net.inet6.ip6.mcast.maxgrpsrc: 512
net.inet6.ip6.mcast.maxsocksrc: 128
net.inet6.ip6.mcast.loop: 1
net.inet6.ip6.only_allow_rfc4193_prefixes: 0
net.inet6.ipsec6.def_policy: 1

net.inet6.ipsec6.esp_trans_deflev: 1
net.inet6.ipsec6.esp_net_deflev: 1
net.inet6.ipsec6.ah_trans_deflev: 1
net.inet6.ipsec6.ah_net_deflev: 1
net.inet6.ipsec6.ecn: 0
net.inet6.ipsec6.debug: 0
net.inet6.ipsec6.esp_randpad: -1
net.inet6.icmp6.rediraccept: 1
net.inet6.icmp6.redirtimeout: 600
net.inet6.icmp6.nd6_prune: 1
net.inet6.icmp6.nd6_delay: 5
net.inet6.icmp6.nd6_umaxtries: 3
net.inet6.icmp6.nd6_mmaxtries: 3
net.inet6.icmp6.nd6_useloopback: 1
net.inet6.icmp6.nodeinfo: 3
net.inet6.icmp6.errppslimit: 500
net.inet6.icmp6.nd6_debug: 0
net.inet6.icmp6.nd6_accept_6to4: 1
net.inet6.icmp6.nd6_optimistic_dad: 63
net.inet6.icmp6.nd6_onlink_ns_rfc4861: 0
net.inet6.icmp6.nd6_prune_lazy: 5
net.inet6.icmp6.rappslimit: 10
net.inet6.icmp6.nd6_llreach_base: 30
net.inet6.icmp6.nd6_maxsolstgt: 8
net.inet6.icmp6.nd6_maxproxiedsol: 4
net.inet6.icmp6.prproxy_cnt: 0
net.inet6.mld.gsrdelay: 10
net.inet6.mld.v1enable: 1
net.inet6.mld.v2enable: 1
net.inet6.mld.use_allow: 1
net.inet6.mld.debug: 0
net.inet6.send.opstate: 0
net.inet6.send.opmode: 0