IPv6

IPv6 Intro

Refresher

For a quick crash course into IPv6 checkout my IPv6 Quick Explainer guide.

Why Did I Setup IPv6?

Beyond just being good to know because it'll be what we're all using sooner than later there are a few practical advantages of IPv6 over IPv4. Most important to me though is being able to have IP addresses that don't have to be masqueraded by the router. This has several knock-on effects I appreciate.

No Need for Hairpin NAT

I don't have to masquerade IP addresses which means that when I access a device from my LAN I can use the same IP address that is used when people access a device from the WAN. I don't have to setup a hacky Hairpin NAT or necessarily use Split-horizen DNS to just have everything work. The less janky configurations I have to create and maintain to paper over problems of IPv4 the better.

Fine-grained DNS Control

Because each device can have a publically routable address I can setup subdomains to **actually** point to different addresses. As an example I can have `wireguard.swigg.net` point to my router IP address for VPN access while `*.swigg.net` can point to my server IP address I am running in my DMZ. With IPv4 I had to have them both point to my router public IP address and then use some sort of proxy to forward based on hostname plus do something janky like above.

Firewall Setup

Install Shorewall6

Configuring *Shorewall* for IPv6 is nearly identical to how I did it for IPv4. The biggest different is I can skip most things related to *masquerading* since that is less often necessary in the world of IPv6.

The only changes that need to be made is installing and configuring *shorewall6*. I am not going to go over everything again since it is nearly identical to Firewall Setup under IPv4 but pay close attention to the path is now `/etc/shorewall6`

```
# apt install shorewall6
```

```
# /etc/shorewall6/shorewall.conf
- LOG_LEVEL="info"
+ LOG_LEVEL="NFLOG(1,0,1)"
...
- LOGFILE=/var/log/messages
+ LOGFILE=/var/log/firewall.log
...
- IP_FORWARDING=Keep
+ IP_FORWARDING=Yes
```

```
# /etc/shorewall6/zones
+ #----------------------------------------------------------------------
+ # For information about entries in this file, type "man shorewall-zones"
+ #
+ # See http://shorewall.org/manpages/shorewall-zones.html for more information
+
###############################################################################
############
+ #ZONE   TYPE    OPTIONS                 IN                  OUT
+ #                                       OPTIONS             OPTIONS
+ fw      firewall
+ wan     ipv4
```

+ lan   ipv4
+ dmz   ipv4
+ warp   ipv4

/etc/shorewall6/interfaces
+ #--
+ # For information about entries in this file, type "man shorewall6-interfaces"
+ #
+ # See http://shorewall.org/manpages/shorewall-interfaces.html for more information
+
###
############
+ ?FORMAT 2
+
###
############
+ #ZONE□INTERFACE□OPTIONS
+ wan□□WAN_IF□□tcpflags,dhcp,forward=1,accept_ra=2,sourceroute=0,physical=eth0
+ lan□□LAN_IF□□tcpflags,dhcp,forward=1,physical=eth1
+ dmz□□DMZ_IF□□tcpflags,dhcp,forward=1,physical=eth1.8
+ warp□□WARP_IF□□tcpflags,dhcp,forward=1,physical=eth1.9

/etc/shorewall6/policy
+ #--
+ # For information about entries in this file, type "man shorewall-policy"
+ #
+ # See http://shorewall.net/manpages/shorewall-policy.html for more information
+
###
############
+ #SOURCE□DEST□□POLICY□□LOGLEVEL□RATE   CONNLIMIT
+
+ $FW□□all□□□ACCEPT
+ lan□□all□□□ACCEPT
+ dmz□□$FW,wan□□ACCEPT
+ warp□□$FW□□□ACCEPT
+
+ wan□□all□□□DROP□□$LOG_LEVEL
+ # THE FOLLOWING POLICY MUST BE LAST
+ all□□all□□□REJECT□□$LOG_LEVEL

```
# /etc/shorewall6/rules
+ #------------------------------------------------------------------------------------------------------
+ # For information about entries in this file, type "man shorewall-rules"
+ #
+ # See http://shorewall.net/manpages/shorewall-rules.html for more information
+
###############################################################################
###############################################################################
#########################################################################
+ #ACTION□□□SOURCE          DEST           PROTO   DEST    SOURCE          ORIGINAL        RATE         USER/
MARK    CONNLIMIT       TIME           HEADERS         SWITCH          HELPER
+ #                                       PORT    PORT(S)         DEST            LIMIT        GROUP
+ ?SECTION ALL
+ ?SECTION ESTABLISHED
+ ?SECTION RELATED
+ ?SECTION INVALID
+ ?SECTION UNTRACKED
+ ?SECTION NEW
+
+ #       Don't allow connection pickup from the net
+ Invalid(DROP)□□wan             all             tcp
+
+ DNS(ACCEPT)□□all!wan,warp    $FW
+ DNS(ACCEPT)□□$FW,dmz         lan:2001:db8:2fa3:4848::9a57:cec2
+
+ Web(ACCEPT)□□dmz             $FW
+ Web(ACCEPT)□□wan             dmz:2001:db8:2fa3:4848:66:1cb:59a7:bbe1
```

At this point I just have an empty `/etc/shorewall6/snat` configuration because IPv6 doesn't need masqueraded to access the Internet.

```
# /etc/shorewall/snat
+ #------------------------------------------------------------------------------
+ # For information about entries in this file, type "man shorewall-snat"
+ #
+ # See http://shorewall.org/manpages/shorewall-snat.html for more information
+
###############################################################################
###############################################################################
#####
```

```
+ #ACTION          SOURCE          DEST        PROTO   PORT    IPSEC   MARK    USER    SWITCH
ORIGDEST        PROBABILITY
```

Just like before it might be wise to run `shorewall6 check` just to make sure I didn't have any typos.

I already enabled *shorewall-init.service* to secure the system during boot so to hook in *shorewall6* I just needed to edit its configuration and then enable *shorewall6.service* to start at boot like I already did for *shorewall.service* and *shorwall-init.service*.

```
# /etc/default/shorewall-init
- PRODUCTS="shorewall"
+ PRODUCTS="shorewall shorewall6"
```

Then I told it to start at boot.

```
# systemctl enable shorewall6
```

Reboot

It isn't strictly neccessary to reboot but I just prefer to see my system as it would be after it starts up.

```
# reboot
```

Prefix Delegation

I'd recommend reading about Prefix Delegation to get a better understanding of it but the gist is that using DHCPv6 it is possible to request a "prefix" where any IPv6 address starting with that will be routed to the router. Then the router can use that to configure clients on the network to each have a unique address instead of the router only one (as in IPv4) and having to share it using a hack like masquerading.

Install A Client

There are a few different DHCPv6 clients you can use that support *Prefix Delegation* but I decided to go with *wide-dhcpv6-client*. I also tried *dhcpcd* but found the configuration syntax to be a little uglier.

```
# apt install wide-dhcpv6-client
```

The config below is doing a few different things that I'll list but you can read about all the possible dhcp6c.conf configuration.

```
# /etc/wide-dhcpv6/dhcp6c.conf
+
+ interface eth0 {
+ #  send rapid-commit;
+   send ia-na 0;
+   send ia-pd 1;
+ };
+
+ id-assoc na 0 {
+
+ };
+
+ id-assoc pd 1 {
+   prefix ::/60 infinity;
+
+   prefix-interface eth1 {
+     sla-id 0;
```

```
+    sla-len 4;
+    ifid 1;
+  };
+
+  prefix-interface eth1.8 {
+    sla-id 1;
+    sla-len 4;
+    ifid 1;
+  };
+
+  prefix-interface eth1.9 {
+    sla-id 2;
+    sla-len 4;
+    ifid 1;
+  };
+ };
```

- **Lines 3-7** use `eth0` to request a "normal address" with `ia-na` and a delegrated prefix range with `ia-pd`
- **Lines 9-11** are required and correspond to the "normal addresss" I asked for, but there is no extra configuration needed
- **Lines 13 and 14** are the start of the prefix delegation block and I am specifying I want a prefix that gives me a subnet of `/60`. I know *Comcast* will give me a `/60` which is 295,147,905,179,352,825,856 (two hundred ninety five quintillion, one hundred forty seven quadrillion, nine hundred five trillion, one hundred seventy nine billion, three hundred fifty two million, eight hundred twenty five thousand, eight hundred fifty six) so that should be more than enough.
- **Lines 16-20** and the other similar blocks just specify what slice of the prefix I was delegated that I want applied to each interface. The lines `sla-id` (Site-Level Aggregation identifier) is just an index to a what is basically an IPv4 subnet, `sla-len` is the size of that subnet (I requested a /60 so if our SLA is /4 then I end up with /64 which is ideal for an IPv6 subnet), and `ifid` just defines that I want the first address available in our subnet assigned to our interface. So if the IPv6 addresses assigned to this interface was `2601:1833:a3a:100::/64` the interface would have `2601:1833:a3a:100::1/64` assigned to it.

The next step was just to enable and run the service.

```
# systemctl enable --now wide-dhcpv6-client
```

Then I was able to verify that I had publicaly accessible IPv6 addresses.

```
# ip -6 addr
1: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP qlen 1000
```

```
    inet6 2001:6020:ae3:1022:a4d3:f031:fb7e:e629/128 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::2b0:c9ff:fe79:cd77/64 scope link
       valid_lft forever preferred_lft forever
2: eth1@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP qlen 1000
    inet6 2601:1833:a3a:100::1/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::d45a:67ff:fec6:6688/64 scope link
       valid_lft forever preferred_lft forever
3: eth1.8@eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP qlen 1000
    inet6 2601:1833:a3a:101::1/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::d45a:67ff:fec6:6688/64 scope link
4: eth1.9@eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP qlen 1000
    inet6 2601:1833:a3a:102::1/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::d45a:67ff:fec6:6688/64 scope link
       valid_lft forever preferred_lft forever
...
```

Exactly like I hoped, I can see that using `ia-nd` to request a "normal address" for `eth0` resulted in `2001:6020:ae3:1022:a4d3:f031:fb7e:e629/128` being assigned as my routers public IPv6 address. It also looks like the prefix delegation worked since `eth1`, `eth1.8`, `eth1.9` all have the same prefix with incrementing SLA identifiers that you can see represented by `100`, `101`, `102` in their addresses.

DHCP and SLAAC

I already setup *dnsmasq* for IPv4 and so there is very little that needs to be done to add IPv6 support.

I just needed to add `dhcp-range` lines for each subnet. I am tagging them the same as before and using the `::,constructor:<interface>` syntax to tell *dnsmasq* to determine the the prefix the DHCPv6 range should be valid over from the GUAs (Global Unicast Addresse) (publically routable IPs) on each interface. These were assigned in the previous section (Prefix Delegation) by *wide-dhcpv6-client*. Declaring `ra-stateless` configures *dnsmasq* to use SLAAC to automatically configure clients in this prefix.

```
# /etc/dnsmasq.d/dhcp.conf
+ dhcp-range=set:lan,::,constructor:eth1,ra-stateless,12h
+ dhcp-range=set:dmz,::,constructor:eth1.8,ra-stateless,12h
+ dhcp-range=set:warp,::,constructor:eth1.9,ra-stateless,5m
```

Then I enabled router advertisements so *dnsmasq* will broadcast information to any potential clients on the subnet.

```
# /etc/dnsmasq.d/router-advertisements.conf
+ enable-ra
```