Part 4 – DNS server setup for clients

Setup basic DNS server on Linux

So I will eventually have to setup a DNS server on the Wireguard server, but just a simple one. No need to use `bind`, but simply `DNSmasq`.

Install DNSmasq

```
$ sudo apt install dnsmasq


Unpacking dnsmasq-base (2.85-1) ...
Selecting previously unselected package dnsmasq.
Preparing to unpack .../dnsmasq_2.85-1_all.deb ...
Unpacking dnsmasq (2.85-1) ...
Setting up dnsmasq-base (2.85-1) ...
Setting up dns-root-data (2021011101) ...
Setting up dnsmasq (2.85-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/dnsmasq.service →
/lib/systemd/system/dnsmasq.service.
Job for dnsmasq.service failed because the control process exited with error code.
See "systemctl status dnsmasq.service" and "journalctl -xe" for details.
invoke-rc.d: initscript dnsmasq, action "start" failed.
● dnsmasq.service - dnsmasq - A lightweight DHCP and caching DNS server
     Loaded: loaded (/lib/systemd/system/dnsmasq.service; enabled; vendor preset: enabled)
     Active: failed (Result: exit-code) since Mon 2021-09-20 00:43:35 CEST; 18ms ago
    Process: 23908 ExecStartPre=/etc/init.d/dnsmasq checkconfig (code=exited, status=0/SUCCESS)
    Process: 23915 ExecStart=/etc/init.d/dnsmasq systemd-exec (code=exited, status=2)
        CPU: 55ms

Sep 20 00:43:35 hostname systemd[1]: Starting dnsmasq - A lightweight DHCP and caching DNS server...
Sep 20 00:43:35 hostname dnsmasq[23915]: dnsmasq: failed to create listening socket for port 53: Address
already in use
Sep 20 00:43:35 hostname dnsmasq[23915]: failed to create listening socket for port 53: Address already in use
```

```
Sep 20 00:43:35 hostname systemd[1]: dnsmasq.service: Control process exited, code=exited,
status=2/INVALIDARGUMENT
Sep 20 00:43:35 hostname dnsmasq[23915]: FAILED to start up
Sep 20 00:43:35 hostname systemd[1]: dnsmasq.service: Failed with result 'exit-code'.
Sep 20 00:43:35 hostname systemd[1]: Failed to start dnsmasq - A lightweight DHCP and caching DNS server.
Processing triggers for dbus (1.12.20-2) ...
```

Run DNSmasq at startup:

```
$ sudo systemctl enable dnsmasq
```

The problem is that DNSmasq tries to bind to port `53`, which on my system is already occupied by `systemd-resolved`. I need to figure out how to make these two work together or disable systemd-resolved.

> Heads up – Skip to ***Final configuration for server*** to see the final configuration files. The following is mostly me trying to figure out how to do this properly. If you enjoy reading about others suffering, go ahead.

Edit /etc/systemd/resolved.conf

DNS configuration is usually managed by `/etc/resolv.conf` file, however, if you open the file, it states the following – `# This file is managed by man:systemd-resolved(8). Do not edit.`

Therefore use `/etc/systemd/resolved.conf` to perform any configuration regarding DNS. We are interested mainly in the `DNSStubListener` line, which we have to uncomment and set to `no`. That disables systemd-resolved from binding to `127.0.0.53:53`. I would also advise to disable `LLMNR` and `MulticastDNS`, do some research about these two yourself to figure if you need them. The file should now look like this:

```
[Resolve]
DNS=your_DNS_server
#FallbackDNS=
#Domains=
LLMNR=no
MulticastDNS=no
#DNSSEC=allow-downgrade
#DNSOverTLS=no
#Cache=yes
DNSStubListener=no
#ReadEtcHosts=yes
```

Restart systemd-resolved. Be aware that you might lose DNS connectivity until you start DNSmasq. I won't do that right now, because I still need to change a few things:

```
$ sudo systemctl restart systemd-resolved
```

Running `systemd-resolve` to check it's status reveals that some settings aren't still how I wanted. I have managed to change the Global configuration, but not settings for individual interfaces.

```
$ sudo systemd-resolve --status

Global
       Protocols: -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
resolv.conf mode: uplink
     DNS Servers: my_prefered_DNS_server

Link 15 (wg0)
Current Scopes: none
     Protocols: -DefaultRoute +LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported

Link 19 (eth0)
Current Scopes: DNS
     Protocols: +DefaultRoute +LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
   DNS Servers: DNS_server_i_dont_like
```

I haven't find another config file to add these values into, so just change them using the `systemd-resolve` command.

```
$ sudo systemd-resolve --interface eth0 --set-dns my_prefered_DNS_server
$ sudo systemd-resolve --interface eth0 --set-llmnr no
$ sudo systemd-resolve --interface wg0 --set-llmnr no
```

Again, confirm that everything is set to your liking with `systemd-resolve --status`

```
$ systemd-resolve --status
Global
         Protocols: -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
  resolv.conf mode: uplink
Current DNS Server: my_prefered_DNS_server
       DNS Servers: my_prefered_DNS_server

Link 15 (wg0)
```

```
Current Scopes: none
     Protocols: -DefaultRoute -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported


Link 19 (eth0)
    Current Scopes: DNS
         Protocols: +DefaultRoute -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
Current DNS Server: my_prefered_DNS_server
       DNS Servers: my_prefered_DNS_server
```

Edit /etc/dnsmasq.d

By default, the file is full of commented out lines with all kinds of configuration settings. In fact, it's so long that it's better to back it up and write just the options you want into an empty one.

```
$ sudo cp /etc/dnsmasq.conf /etc/dnsmasq.conf.default
```

Delete everything in `/etc/dnsmasq.conf` with `vi` and `1000dd`.

```
$ sudo vi /etc/dnsmasq.conf
```

Add the following lines to the config:

```
interface=lo,wg0
bind-interfaces
```

Restart DNSmasq and check if it binded to the right interfaces (localhost and Wireguard)

```
$ sudo systemctl restart dnsmasq


$ sudo netstat -tulpn | grep 53
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN      25764/dnsmasq
tcp        0      0 10.20.20.1:53           0.0.0.0:*               LISTEN      25764/dnsmasq
udp        0      0 127.0.0.1:53            0.0.0.0:*                           25764/dnsmasq
udp        0      0 10.20.20.1:53           0.0.0.0:*                           25764/dnsmasq
```

All looks great, however, trying multiple pings to common websites reveal random DNS issues:

```
$ ping google.com
ping: google.com: Temporary failure in name resolution
```

> At this point, I have spent a few hours researching how name resolution works on Linux machines. It got a little more complicated with systemd-resolved. I won't explain the whole proccess or what I did, just show you the final configuration files and try to explain how the name resolution works as of now.

Previous plan

I'm currently trying to figure out how to get DNS resolution working on the server itself (so that the server can resolve names with `DNSmasq` and `systemd-resolved` combined). My previous plan was:

- Disable `systemd-resolved` binding to `127.0.0.53` with `DNSStubListener=no` in `/etc/systemd/resolved.conf`
- Set DNS on all interfaces with systemd-resolved to `127.0.0.1` (this also changes nameserver in `/etc/resolv.conf` to `127.0.0.1`)
- Run DNSmasq on `127.0.0.1` (for the server itelf) and on `10.20.20.1` (for the Android Wireguard client).
- Set DNS server in DNSmasq (`/etc/dnsmasq.conf`) to a public DNS of choice for regular name resolution.

Thanks to this, when the server wants to perform a DNS query, this happens *(as far as I understand, don't @ me and kindly let me know if I'm wrong):*

1. It first looks into `/etc/nsswitch.conf` and looks onto the `hosts` line, which says `files dns`.
2. Because of `files`, it looks into `/etc/hosts` first, to see if the address isn't hard-coded this way.
3. Then because of `dns`, it looks into `/etc/resolv.conf`, which is managed by systemd-resolved. This file will only contain `127.0.0.1`, which points to DNSmasq running on that interface on port `53`. The reason why it points to `127.0.0.1` is because we have set DNS to `127.0.0.1` globally in `/etc/systemd/resolved.conf`.
4. DNSmasq will take that query and forward it to our prefered public DNS server configured in `/etc/dnsmasq.conf` with the `server` line.

Unfortunately, this **does not work**, or at least I was unable to make it work this way. The outcome of this is a broken name resolution that sometimes works and sometimes doesn't.

Current plan

To fix my server's DNS resolution, I came up with a reversed setup plan. Instead of everything pointing to DNSmasq for name resolution, I will point DNSmasq to the other side, which is systemd-resolved – **this turned out to produce the same behavior.**

After thorough examination, I decided to completely throw systemd-resolved out the window by stopping the service and disabling it.

```
$ sudo systemctl stop systemd-resolved
$ sudo systemctl disable systemd-resolved
```

Try the good old way of editing `/etc/resolv.conf` and add your DNS server. After adding different upstream DNS servers, I'm slowly starting to realize that the entire issue might have been actually caused by the upstream server and not my local configuration. The DNS I used before is extremely fast (+-2 ms response times), but sometimes just stops responding completely. It might have something to do with reverse lookup and I might somehow fix it later, but for now, I have decided to opt for the slower, but more reliable upstream DNS server.

Final configuration for server

There might actually be a way to do it! Go back to my reversed setup plan, because this time it works.

Enable and start systemd-resolved again:

```
$ sudo systemctl enable systemd-resolved
$ sudo sytemctl start systemd-resolved
```

Look into `/etc/resolv.conf`. Because we have started systemd-resolved, it had already probably overwrote `/etc/resolv.conf` with the following, which is alright:

```
nameserver 127.0.0.53
options edns0 trust-ad
search .
```

Open `/etc/systemd/resolve.conf` and make it look like this:

```
[Resolve]
DNS=my_prefered_DNS_server
FallbackDNS=secondary_DNS_server
#Domains=
LLMNR=no
MulticastDNS=no
#DNSSEC=allow-downgrade
#DNSOverTLS=no
#Cache=yes
DNSStubListener=yes
#ReadEtcHosts=yes
```

Check status of systemd-resolve and edit DNS server for `eth0` if it still has a wrong value

```
$ systemd-resolve --status

Global
          Protocols: -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
   resolv.conf mode: stub
 Current DNS Server: my_prefered_DNS_server
        DNS Servers: my_prefered_DNS_server
Fallback DNS Servers: secondary_DNS_server

Link 15 (wg0)
Current Scopes: none
     Protocols: -DefaultRoute -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported

Link 19 (eth0)
   Current Scopes: DNS
        Protocols: +DefaultRoute -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
Current DNS Server: my_prefered_DNS_server
       DNS Servers: my_prefered_DNS_server
```

Your `/etc/dnsmasq.conf` should look like this. We want to run DNSmasq only on the Wireguard interface, but interface also automatically adds loopback to the list. Use `except-interface` to disable binding to localhost.

```
interface=wg0
except-interface=lo
bind-interfaces
server=127.0.0.53
```

Start DNSmasq

```
# sudo systemctl start dnsmasq
```

You can tell by the output of `systemctl status dnsmasq`, that it's using `127.0.0.53` as it's DNS server.

```
...
using nameserver 127.0.0.53#53
reading /etc/resolv.conf
...
```

Recap what we know about name resolution on the server now. When the server makes a DNS query, this happens (probably):

- Debian consults `/etc/nsswitch.conf`, reads `/etc/hosts` due to `files` option specified at the `hosts:` line, then goes for the DNS server.
- DNS server in `/etc/resolv.conf` points to `127.0.0.53`, which is `systemd-resolv DNSStubListener`.
- `systemd-resolved` is set up to use a prefered public DNS server.

Now onto the main reason why I did this whole thing. Provide a simple DNS server for my Android client.

Setup DNS for client name resolution

Edit /etc/hosts

Add the websites you want to resolve to `/etc/hosts`. That way, when clients queries DNSmasq listening on `10.20.20.1`, it forwards the query to `127.0.0.53`, which also reads `/etc/hosts`, resolving the IP correctly.

```
10.20.20.1      mysite.com
10.20.20.1      mysite2.com
```

Add DNS to Android config

Edit the config by adding `10.20.20.1` to `DNS servers`.

Adjust Firewall for testing

In order for the Android device to be able to use the DNS server on `10.20.20.1`, I need to adjust `iptables` to allow traffic on port 53 on the Wireguard interface. Add this to your iptables configuration file and apply with `iptables-restore`.

```
# Allow DNS queries on the Wireguard interface
-A INPUT -p udp -m udp --dport 53 -i wg0 -j ACCEPT
```

```
$ sudo iptables-restore /etc/iptables/rules.v4
```

> Now access your websites from the Android client and it should resolve correctly to the server's Wireguard IP.

The only "problem" right now is that the Android client will most likely use the Wireguard server IP `10.20.20.1` for all name resolution while it's connected to the VPN. This is not a huge deal since the queries are pretty fast, but might be a deal breaker for some.

Revision #3
Created 22 September 2021 01:52:33 by Marek
Updated 22 September 2021 01:58:52 by Marek